The University of Nottingham

Virtual Reality for Fixture Design and Assembly

by Qiang Li, M.A

Thesis submitted to the University of Nottingham for
the degree of Doctor of Philosophy

October 2008

Abstract

Due to today's heavy, growing competition environment, manufacturing companies have to develop and employ new emerging technologies to increase productivity, reduce production costs, improve product quality, and shorten lead time. The domain of Virtual Reality (VR) has gained great attention during the past few years and is currently explored for practical uses in various industrial areas e.g. CAD, CAM, CAE, CIM, CAPP and computer simulation etc.

Owing to the trend towards reducing lead time and human effort devoted to fixture planning, the computerization of fixture design is required. Consequently, computer aided fixture design (CAFD) has become an important role of computer aided design/manufacture (CAD/CAM) integration. However, there is very little ongoing research specially focused on using the VR technology as a promising solution to enhance CAFD systems' capability and functionality.

This thesis reviews the possibility of using interactive Virtual Reality (VR) technology to support the conventional fixture design and assembly process. The trend that the use of VR benefits to fulfil the optimization of fixture design and assembly in VE has been identified and investigated.

The primary objectives were to develop an interactive VR system entitled Virtual Reality Fixture Design & Assembly System (VFDAS), which will allow fixture designers to complete the entire design process for modular fixtures within the Virtual Environment (VE) for instance: Fixture element selection, fixture layout design, assembly, analysis and so on. The main advantage of VFDAS is that the VR system has the capability of simulating the various physical behaviours for virtual fixture elements according to Newtonian physical laws, which will be taken into account throughout the fixture design and evaluation process. For example: gravity, friction, collision detection, mass, applied force, reaction force and elasticity. Almost the whole fixture design and assembly process is achieved as if in the real physics world, and this provides a promise for computer aided fixture design (CAFD) in the future.

The VFDAS system was validated in terms of the collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results are presented and quantified by a series of simple examples to show what the system can achieve and what the limitations are.

The research concluded VR is a useful technology and VFDAS has potential to support education and application for fixture design. There is scope for further development to add more useful functionality to the VFDAS system.

Acknowledgements

I would first like to thank my supervisors Dr Sue Cobb, Dr Richard Eastgate and Dr Xun Chen without whom this thesis would probably not exist. Their constant support and guidance have been immeasurably helpful for this research. I am truly grateful.

I would like to thank everyone from the Human Factors Research Team without exception for always being ready and willing to help be it with technical difficulties or a cuppa and a supportive word.

Finally I would like to thank my family and my girl friend Lin Lin Qiu for their unwavering support throughout my studies and never doubting that I would make it in the end, thank you.

Table of Contents

List of Figures

GLOSSARY

2D Two Dimensional, a ‘flat’ drawing, scene or object that exhibits no sense of depth

3D Three Dimensional, a drawing, scene or object with appearance of sense of depth

BOOM Binocular Omni-Orientation Monitor, virtual reality display system

CAD Computer Aided Design, the use of computer technology to aid in the design

CAE Computer Aided Engineering, the use of information technology for supporting engineers in tasks such as analysis, simulation, design, manufacture, planning, diagnosis and repair.

CAFD Computer Aided Fixture Design, the use of computer technology to aid in the fixture design

CAM Computer Aided Manufacturing, the use of computer-based software tools that assist engineers and machinists in manufacturing or prototyping product components

CAMFD Computer Aided Modular Fixture Design, the use of computer technology to aid in the modular fixture design

CAPP Computer Aided Process Planning

CAVE Cave Automatic Virtual Environment, fully immersive virtual reality system

CIM Computer Integrated Manufacturing, a method of manufacturing in which the entire production process is controlled by computer

CNC Computer Numerical Controlled, a computer “controller” that reads G-code instructions and drives a machine tool

FMS Flexible Manufacturing System, a manufacturing system in which there is some amount of flexibility that allows the system to react in the case of changes, whether predicted or unpredicted.

GUI Graphical User Interface, something you can see or interact with, such as a button you could click on

HMD Head Mounted Display, a display mounted on the participant’ head. The HMD is usually tracked, meaning that a participant’s head movement results in an appropriate change in the visual scene.

Immersion A feeling of psychological involvement in a VE. Also the term used to describe the period of VR use.

Interaction Any action made by the user that results in a change in the VE, selection and navigation

Interactivity The potential for a VE to react to the participant’s movements and behaviour

MFS Modular Fixturing System

Navigation The process of the user affecting their field of view within the VE, in effect moving around the VE

Objects In a VE context objects are visual representations of an entity (could be made up of many shapes).

Presence The users sense of actually ‘being there’ within the VE.

Realism How closely the VE visually or audibly resembles the real world environment on which it is based

Rendering A process undertaken by the computer which continually draws the VE

VE Virtual Environment, the visual representation of the “virtual world” viewed by the participant, and produced by the VR system.

VE Usability A series of factors that attempt to enhance the VE for the participant’s virtual experience

VFDAS Virtual Reality Fixture Design and Assembly System

Virtual World The environment created and displayed to the user

VR Virtual Reality, the technology or system on which the virtual environment is displayed

Chapter 1. Introduction

1.1 Background

Virtual Reality (VR) is emerging as a potential application in relation to numerous fields of interest, and has been adopted in diverse research fields including 3d graphics, video games development, mechanical engineering, construction engineering, manufacturing engineering, ergonomic analysis, 3d scientific visualization, medical surgical research, education and cognitive mapping study etc.

Due to today's heavy competition environment, manufacturing companies have to employ new emerging technologies to increase productivity, reduce production costs, improve product quality, and shorten lead time [1]. The domain of VR has gained great attention during the past few years and is currently explored for practical uses in various industrial areas e.g. CAD, CAM, CAE, CIM, CAPP and computer simulation etc.

One area in which VR has rarely been applied is fixture design. A fixture is a device that locates, holds and supports workpieces in position and orientation during machining processes. Fixtures play a significant role to assure production quality, shorten production cycle time and decrease production cost. Owing to the trend towards reducing human effort and lead time devoted to fixture planning, the computerization of fixture design is required. Computer aided fixture design (CAFD) has been investigated and has become an important role of computer aided design/manufacture (CAD/CAM) integration [2]. Current systems do not use the features offered by VR technology and so this thesis explains the use of VR in fixture design.

1.2 The Fixturing Work at University of Nottingham

The University of Nottingham Responsive Manufacturing Group has been involved in fixture design research in collaboration with its industrial partner, Rolls-Royce. The University of Nottingham fixture design process adopts a systematic approach to the analysis of interactions between the fixture and other factors. These factors normally refer to component, machine, tool, process planning and inspection.

The University of Nottingham concurrent fixture development methodology supports a range of virtual simulations to model the fixture assembly, fixturing processes, tooling, machining and their interactions at an early stage of fixture design in order to determine their impacts on the component quality. The fixture development procedure is shown in **Figure 1-1**. The virtual simulation tools include CAD/CAM (Computer Aided Design/Computer Aided Manufacturing) modelling, VR (Virtual Reality) simulation, FEA (Finite Element Analysis), and kinematic numerical analysis.

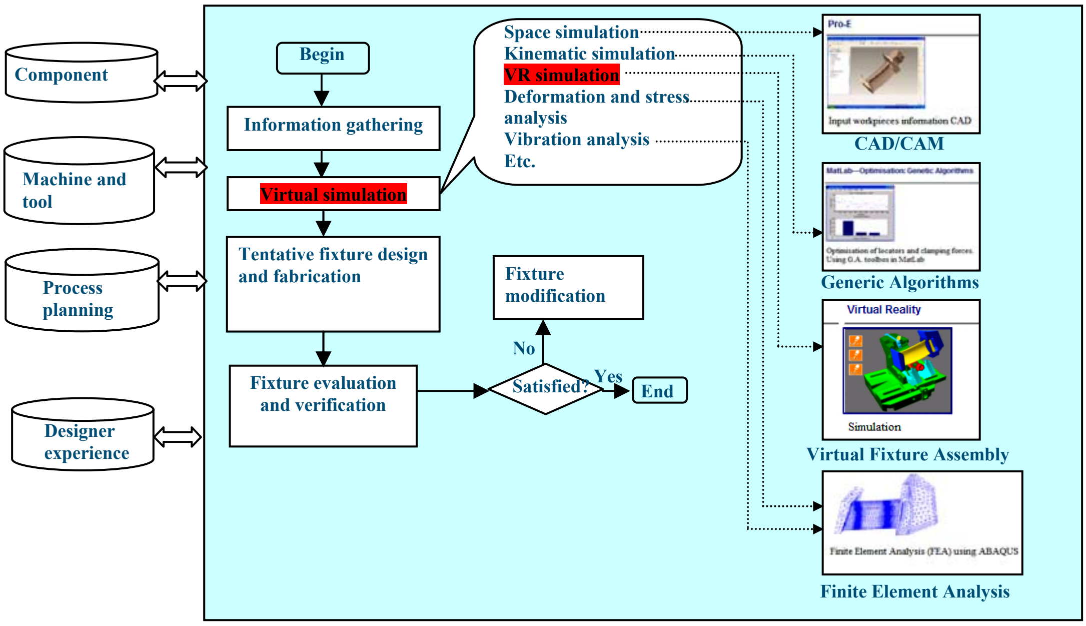


Fig.1-1. The Procedure of Fixture development

The research in this thesis was initially carried out as a part of this research framework and focused on the research aspect of VR simulation for fixture design and assembly which is highlighted in **Fig.1-1**. This research theme was able to contribute the whole research framework of fixture development at University of Nottingham.

1.3 Motivations of VR Application Research for Fixture Design

Through an overview of the Virtual Reality (VR) applications in diverse domains, the author was able to comprehend the potential of VR regarding innovation applications and practical uses in industry. A vast amount of research for VR applications relating to various industry aspects has been conducted over the last thirty years in areas including product design, rapid virtual prototyping, virtual assembly (VA), manufacturing simulation and plant layout etc. However there is very little ongoing research specially focused on "Virtual Reality for Fixture Design and Assembly". This was identified as a research gap that may be useful for industrial development.

1.4 Aims & Objectives

The research theme in this thesis is addressed as "Virtual Reality for Fixture Design and Assembly". The primary objectives and aims were to develop an interactive VR system entitled Virtual Reality Fixture Design & Assembly System (VFDAS), which would allow fixture designers and assembly engineers to complete the design process for modular fixtures within the virtual environment (VE). This required the identification of what types of VR scenario and interactivity in VFDAS are needed to achieve the functionality in terms of computer aided fixture design (CAFD) and solve the actual problems arising from the fixture design process, such as: fixture location determination and assembly interference check. The VFDAS system would need to comprise the functionality of fixture element selection, fixture layout design, assembly planning, essential analysis and so on. More specifically, it would be useful for VFDAS to be used to select modular fixtures from VR libraries, spatially manipulate each element to assemble on the arbitrary workpiece users supply in preference, detect the assembly collision in advance and implement design optimization and verification etc. Furthermore, the feasibility of using interactive VR technology to facilitate the fixture design and assembly process has to be evaluated. The advantages of VR that could be used to support the conventional fixture design have to be identified.

The main objectives of VFDAS development are briefly summarized as follows:

A. to provide fixture designers a platform to virtually implement interactive VR fixture design and assembly so as to select and position the elements one by one around the workpiece to generate the fixture assembly as if in the real physics world

B. to realize rapid evaluation and iteration of multiple fixture design ideas to support the problem definition in a virtual environment

C. to provide fixture designers the intuitive manipulation and natural manner to fulfil the fixture design and assembly process in VFDAS system

D. to build a user-friendly library and contain a number of modular fixtures within this VR system in order to provide fixture designers with more alternatives for the element selection and simplify the fixture design process

E. to generate the appropriate visualization and 3D graphical representation for fixture elements during the process of fixture design and assembly

F. to realize the assembly process planning for fixtures using the interactive VR simulation and provide the function of recording the animation for the process of virtual fixture design and assembly

G. to provide the function of cost analysis and generate bill of materials (BOM)

H. to establish the capability of simulating various physical properties and real-time 3D collision detection in accuracy in VFDAS according to Newtonian laws e.g. mass, gravity and friction etc

I. to analyze the machining strategy and cutting path envelope before execute fixture layout design so as to avoid design interference

J. to act as an effective role of communication between fixture designers and people from different fields

K. to improve fixture productivity and economy

L. to perform the kinematics analysis for the verification of a fixture design

M. to reversely transfer the final fixture assembly model back to the CAD system so that it is ready for the subsequent manufacturing process

N. to develop a comprehensive fixture design system by incrementally combining many micro focused research activities and functions into the VFDAS system

1.5 The State of the Art of CAFDs

A great amount of research has been conducted in the computer-aided fixture design (CAFD) area. In order to demonstrate the state of the art of CAFD research, three latest typical CAFD systems are selected and reported as follows. More literature and introduction related to CAFD research can be found in section 2.2.4 in the literature review chapter.

Shokri et al [96] recently reported a new software in 2008 that can be used to plan fixture configuration and assembly procedure for modular CMM measuring fixtures. The system is implemented applying VC++ in a Solidworks platform. An assembly algorithm is developed to design a suitable fixture configuration and assembly procedure by using 'minimum fixture elements' and 'simple fixture configuration' criteria. Using Set Theory, a fixture structure is considered as an assembly unit divided into several sub-assemblies and elements. Each sub-assembly is defined as a functional unit that comprises one contacting element, one connecting element and some other assembly elements.

Kang et al [97] addressed another fixture design system for networked manufacturing in 2007, which includes characteristics of rapid configuration designing, three-dimensional modeling, a standardized elements database. This CAFD system may also transfer information with various other systems. For the requirement of rapid configuration design, based on the analysis of the contents and characteristics of fixture design, a hybrid CBR/KBR (case-based reasoning /knowledge-based reasoning) fixture design method was applied to develop this fixture design system.

In comparison with the two CAFD systems, the interactive VR system may have the advantages of making the fixture design in a natural and instructive manner, providing better match to the working conditions, reducing lead-time, and improving fixture productivity and economy. VR system can also support the visualization or planning of a whole assembly process but the conventional CAFD systems can not achieve this.

Peng et al [98] reported a novel modular fixture design and assembly system based on VR in 2006, which comprises the Graphic Interface (GUI) module, Virtual Environment (VE) module and Database module. The GUI is basically a graphic interface that is used to integrate the virtual environment and modular fixture design actions. The VE provides the users with a 3D display for navigating and manipulating the models of modular fixture system and its components in the virtual environment. The database deposits all the models of environment and modular fixture elements, as well as the domain knowledge and useful cases.

Peng et al [103] addressed a precise manipulation approach to support the interactive design and assembly of modular fixture configuration in a virtual environment in 2008 in order to develop a VR-based modular fixture assembly design system.

Although Peng et al [98] [103] claim that their modular fixture design systems are based on VR technology, the essential physical features of VR actually can not be found in these two systems such as: mass, gravity, friction, applied force, elasticity and toppling. To some extent, these two CAFD systems should not be regarded as the sound VR simulations for fixture design. Moreover, all these four CAFD systems presented above lack the capability of the real-time 3D collision detection which is required to conduct the design interference check between fixturing elements, workpiece and machine tools.

To fill the research gap, the intention of this thesis is to develop the VFDAS system that can simulate these physical features of VR and the real-time 3D collision detection in accuracy so that the fixture design and assembly process is realistically carried out as if in the real physics world. The VFDAS system can also demonstrate the fixture physical behaviour in use and restore the real fixture design conditions in realism.

1.6 Overview of the Research

1.6.1 Brief Introduction

The research in this thesis was engaged with multidisciplinary knowledge comprising different expertise in interactive VR system development, fixture design, ergonomics and computer aided design. The major research deliverable in this thesis would be the novel VFDAS system that is developed to support the CAFD process.

Fixture designers were the target end users of the proposed VFDAS system. For the purpose of this research, fixture designers were classified into two categories of people: amateur fixture designers e.g. engineering students and academic technicians; expert users from industry.

To support the development of VFDAS, a couple of interactive VR simulations were developed to provide pilot case studies that could be used to determine the appropriate methodology and estimate the possible difficulties and requirements. In addition, these pilot studies were used as demonstrators for considering the additional value of VFDAS towards the engineering education in terms of fixture design.

Finally, the VFDAS system will be validated in terms of collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results of VFDAS will be demonstrated and quantified by a series of simple examples so as to show what the system can achieve and what the limitations are.

1.6.2 Physics Simulation and Accurate Collision Detection in VFDAS

As a lot of research has shown, the realization of accurate collision detection is always a critical aspect of VR simulation system during a virtual manufacturing process. Although a great research progress has been made in the development of accurate, efficient and real-time collision detection algorithms for convex objects, the research in developing the collision detection algorithm of non-convex objects i.e. concave objects still remains limitations and difficulties [3].

The methodology that is used to perform the simulation of real-time 3D collision detection in accuracy will be explored and identified in this research in order to shorten the research gap of accurate collision detection with regard to the concave 3D objects. Based on this potential methodology, the framework of the VFDAS system could be established. The development of VFDAS will not only provide users the real-time 3D collision detection in accuracy but also achieve the good quality of simulation results of physical properties that demonstrate an interactive fixture design platform within a physics VR environment. These physical properties in VFDAS refer to gravity, friction, collision detection, mass, applied force and elasticity associated with fixture elements. These physical properties normally will be taken into account during the fixture design and evaluation process.

1.7 Structure of Thesis

The structure of this thesis is shown in **Fig.1-2**.

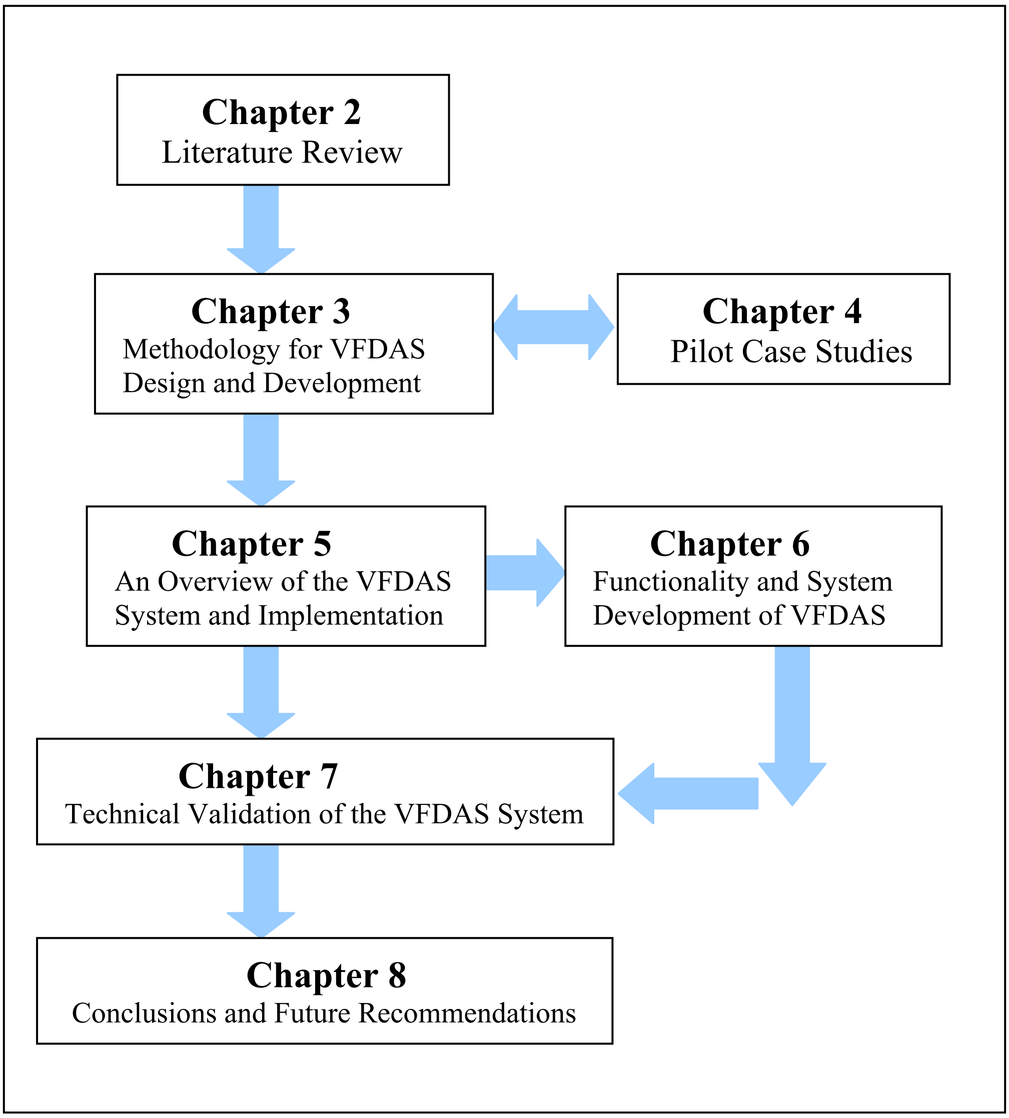

Chapter 2
Literature Review
Chapter 3
Methodology for VFDAS Design and Development
Chapter 4
Pilot Case Studies
Chapter 5
An Overview of the VFDAS System and Implementation
Chapter 6
Functionality and System Development of VFDAS
Chapter 7
Technical Validation of the VFDAS System
Chapter 8
Conclusions and Future Recommendations

Fig.1-2. The Thesis Structure

Following the introduction chapter two looks at the literature concerning VR and fixture design. The potential applications VR technology could be undertaken will also be presented and made a comparison by a description of reading literatures. In particular, the review resource related to VR for industrial uses would be described in detail. All of them may contribute to the resulting effectiveness of the proposed VFDAS system under the development.

Chapter three aims to introduce the comprehensive methodology which has been appropriately used to achieve the VFDAS design and development. A series of affiliated methods comprised within it relating to various research aspects e.g. fixture development, VE behaviour programming, computer aided design, the integration of VR and CAD and usability design would be explained one by one in this chapter.

Chapter four describes the two pilot case studies. Through conducting them, the appropriate methodology, essential difficulties and requirements in relation to carry out the research in this thesis can be estimated and determined in advance before reach the final goal of developing the VFDAS system.

Chapter five demonstrates the specific functionality and system architecture relating to the VFDAS. The primary 14 functioning modules associated with the VFDAS are demonstrated according to their functionality and interaction activities. Moreover, the application mechanism and industrial role related to the VFDAS system are explained. Finally, the approach how to identify the requirements and functionality of the VFDAS may also be presented in this chapter.

Chapter six introduces the development process of the VFDAS system which was theoretically divided into three incremental development phases. Thus, the relevant behaviour programming details in relation to the completion of the main functionality and interactions of the VFDAS are explained.

Chapter seven technically discusses and validates the simulation results of the VFDAS system with regard to collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results of VFDAS are presented and quantified by a series of simple examples to show what the system can achieve and what the limitations are. Furthermore, the essential computer platform used for the VFDAS system will also be discussed.

Chapter eight summarizes the conclusions in relation to the research in this thesis as well as addresses the contributions to the aims and objectives of the research. Finally, recommendations for future research based on a continuation of the research in this thesis are stated.

Chapter 2. Literature Review

2.1 Introduction

This chapter presents a review of CAFD literature and an overview of Virtual Reality (VR) technology. Section 2.2 reviews the research aspect related to fixture design. Section 2.3 explains what attributes of VR could be from various aspects. Section 2.4 presents typical VR applications in relation to education and training. Finally, section 2.5 focuses on a literature review of VR for industrial applications e.g. industrial design and assembly.

Figure 2-1 illustrates the structure of the literature review and how the research became centred on VR application for fixture design and assembly.

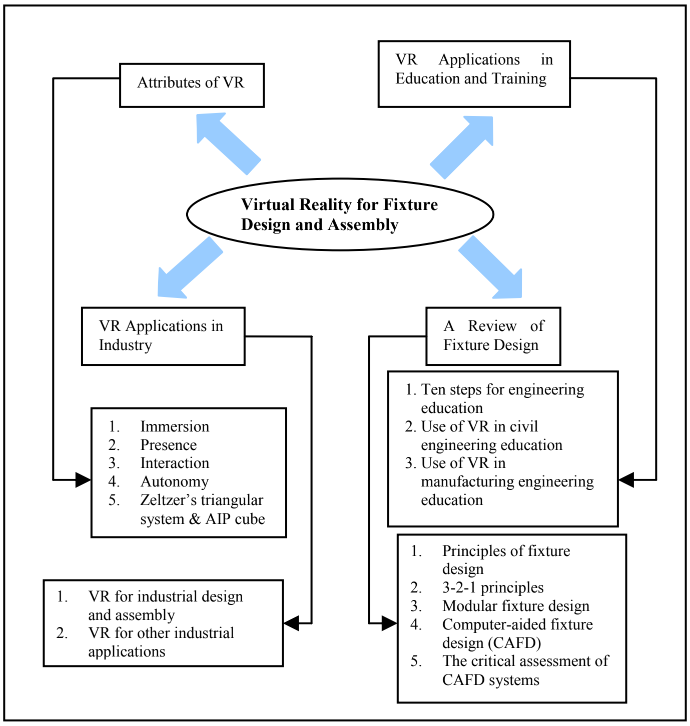

Attributes of VR
VR Applications in Education and Training
Virtual Reality for Fixture Design and Assembly
VR Applications in Industry
A Review of Fixture Design
1. Immersion
2. Presence
3. Interaction
4. Autonomy
5. Zeltzer's triangular system & AIP cube
1. Ten steps for engineering education
2. Use of VR in civil engineering education
3. Use of VR in manufacturing engineering education
1. VR for industrial design and assembly
2. VR for other industrial applications
1. Principles of fixture design
2. 3-2-1 principles
3. Modular fixture design
4. Computer-aided fixture design (CAFD)
5. The critical assessment of CAFD systems

Fig.2-1. Literature Review Structure

2.2 A Review of Fixture Design

A fixture is a mechanism used in manufacturing to hold a workpiece, position it correctly with respect to a machine tool and support it during machining [4]. The requirement of a fixture is to locate and secure the workpiece in the correct orientation and relationship so that the manufacturing process can be carried out according to design specifications [5]. Fixture design is regarded as a concurrent activity of process planning. Fixturing development is an extremely complex procedure, as it has been involved in many aspects of product development cycle: design, process planning, manufacturing, inspection. The majority of fixtures designs are performed for a particular workpiece, as 'dedicated fixtures'. Due to the trend in manufacturing promoting a larger product diversity, flexibility and quality, many companies require the fixturing systems to be more flexible. They can allow a variety of parts to be held during machining and assembly, thus minimizing cost for dedicated fixtures and reducing the inventory of a multiplicity of fixtures [5]. Consequently, the concept of 'modular fixture' has emerged and popularized in term of the flexibility requirements.

During the past two decades, the manufacturing research community has focused on developing up to date technologies e.g. computer-aided design and manufacturing (CAD/CAE), computer-aided process planning (CAPP). Along with the technological evolution, the new method of computer-aided fixture design (CAFD) has been used to reduce lead time and human effort devoted to fixture planning [6].

2.2.1 Principles of Fixture Design

A typical fixture design for prismatic parts consists of three essential elements: locators, clamps and supporters. Locators are employed to position the workpiece in static equilibrium so as to remove all degrees of freedom. Clamps are used to hold the workpiece firmly against the locators during the machining process. The primary design factors of fixture clamps comprising external cutting force and tool direction etc have to be taken into account during a fixture design process. Supporters are added to improve the stability of the workpiece. The use of these fixturing elements can be determined manually or analytically [5].

As machining features are subjected to external cutting forces, the three fixture elements have to guarantee that the workpiece is rigidly located and assure its repeatability. Repeatability refers to the workpiece and the subsequent workpiece can be precisely located in the same position by the fixture. Furthermore, the external cutting forces have to be counteracted by the fixture so that the workpiece remains in equilibrium to ensure the specification and tolerance [5].

An unconstrained workpiece will have 12 degrees of freedom in three dimensional space, because its movements can follow along the positive and negative directions of the X, Y and Z axes as well as the clockwise and counter-clockwise rotations around the three axes **(Fig.2-2)**. During the machining process, the degrees of freedom of movement of the workpiece must be constrained by the locators and clamps.

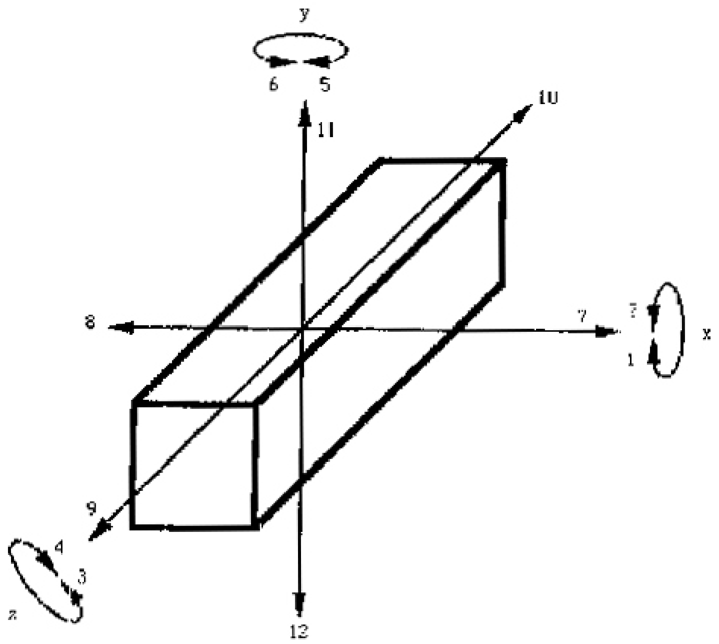


Fig.2-2. Twelve Degrees of freedom for prismatic workpiece

2.2.2 3-2-1 Principles

The 3-2-1 principle is usually applied to the prismatic workpiece while modular fixture elements are employed for the fixture design. The key concept of the principle is to define the first, second and third reference datum according to three vertical locators, two horizontal locators and one horizontal locator respectively. Afterwards, one or more clamps are used to perform clamping on the opposite facet of each reference datum so that the workpiece can be securely held and positioned during the manufacturing process. The main guidelines with respect to the determination of locating and clamping locations have been summarized by Hou and Trappey [6] as follows.

(1) The first datum is often defined as the facet with the largest area and parallel with the base plate. The second datum is often the facet with the largest projective edge (onto the base plate). The third datum is the facet close and perpendicular to the second datum.

(2) The area extended by the three vertical locators should be maximized to achieve a higher stability.

(3) It is better that the three vertical locators are placed on the interior points of the first datum to reduce deflection

(4) The centre of gravity of the workpiece needs being projected into the triangle shape formed by the three vertical locators in order to prevent instability.

(5) The two horizontal locators on the second datum should be positioned apart from each other.

(6) As the workpiece is sheet-like in structure, the revised N-2-1 (N > 3) principle can be employed to prevent blending.

(7) Directions of the clamping forces are required to against the corresponding horizontal and vertical locators to prevent excessive torque.

(8) The locator layout should be arranged to against the main cutting force exerted by machining.

(9) The angle between the second and third datum should be $>90^{\circ}$ and $< 120^{\circ}$ to avoid being over-located.

(10) The horizontal and vertical clamping facets should be placed opposite to the corresponding locating datum (i.e. the angle between their out normals has to be $> 150^{\circ}$)

(11) If any through hole exists in the workpiece, locators with the appropriate dimension can be used to provide a higher stability.

(12) Supports with floating mechanism (not fixed locator) are able to prevent over-located and severe deflections.

2.2.3 Modular Fixture Design

The concept of modular fixture dates back from the Second World War. Modular fixturing systems firstly came into prominence in the late 1960s and were primarily employed along with NC machine tools. However, the range of their applications was not widespread until the appearance of multiple axis computer numerical controlled (CNC) machine tools and flexible manufacturing systems (FMS) [7].

With the needs for flexibility and increasing design complexity of products, modular fixtures have finally emerged. A modular fixture achieves flexibility via multi-purpose fixturing elements. A sound modular fixturing system normally consists of a large number of standard fixturing elements such as baseplates, locators, clamps and support elements etc. These fixturing elements can be selected to construct the fixture configuration so as to hold the workpiece for machining. The majority of modular fixtures are classified into kits with T-slotted, dowel pin and grid holes baseplate fixtures. Generally, T-slotted, dowel pin holes and grid holes are disseminated on all faces of the base plates [5]. The supporting, locating and clamping elements are fastened together with bolts held in T-nuts or with capped screws. The multiple types of commercial Modular Fixturing Systems (MFS) are presented as follows.

(1). T-Slot or Tenon-Slot system:

(a) Erwin Halder Modular Jig and Fixture System, USA
(b) Warlton Unitool, UK
(c) CATIC (China National Aeronautical Technology Import and Export Corporation) System, China
(d) Gridmaster System, UK (Tenon Slot)

(2). Dowel pin system

(a) Bluco Technik, Germany
(b) SAFE (Self Adapting Fixture Element) System, USA
(c) Write Alufix System, Germany.

(3). Grid hole system

a) Venlic Block Jig System, IMAO Corporation, Japan

b) Yuasa Modular Flex System, USA

c) Kipp Modular Flexible Fixturing System, Germany

Additionally, there are many reasons for using modular fixtures. The major advantages of using modular fixtures are listed below:

(1). Modular fixturing elements enable re-usage for other products once they are dissembled.

(2). The requirements of rapid design, changes and modification can be realized by using modular fixtures to allow a faster response to customer's demands.

(3). The need for conducting smaller batch sizes in production can be easily satisfied.

(4). The use of multiple axis CNC machine tools can be greatly facilitated because the interchangeability of modular fixture elements supports their applications.

(5). Modular fixtures reduce the requirement for storage compared to dedicated fixtures and the time and labour cost spent on designing dedicated fixtures.

(6). Fixture construction can be simply executed without the need for engineering drawings.

(7). Modular fixtures are suitable for a variety of workpieces and adaptable to changes in design, process plan and machine tool.

2.2.4 Computer-aided Fixture Design (CAFD)

Computer-aided design (CAD) and computer-aided manufacturing (CAM) systems are referred as standard engineering tools to reduce the time and cost of product design and manufacturing. Despite the well-known fact that there is an executive gap between CAD and CAM, Computer-aided process planning (CAPP) has been suggested as the link to achieve complete CAD/CAM integration [8].

The function of a fixture is to hold a workpiece firmly in position during a manufacturing process. Due to the trend toward high-precision production and automation, computerization of fixture design is required to reduce the lead time and cost of product development. Therefore, computer-aided fixture design (CAFD) has been developed and used as a part of computer aided design and manufacture (CAD/CAM) integration [2].

According to the classification of fixture design, Wang [9] suggests that the computer aided fixture design (CAFD) can be divided into four types (1). Feature based fixture design (2). Rule based fixture design (3). Sensor based fixture design and (4). Internet based fixture design. Furthermore, Rong et al [10] proposed that CAFD is made up of three design procedures: set-up planning, fixture planning and fixture configuration design **(Fig.2-3)**. Set-up planning defines the number of set-ups, the position and orientation of workpiece in each set-up and the facet of workpiece needed to be machined in each set-up. Fixture planning is to determine the locating, supporting and clamping points on different workpiece surfaces. Fixture configuration design is the important procedure of CAFD, in which the standard fixture elements are chosen and placed to the corresponding locations.

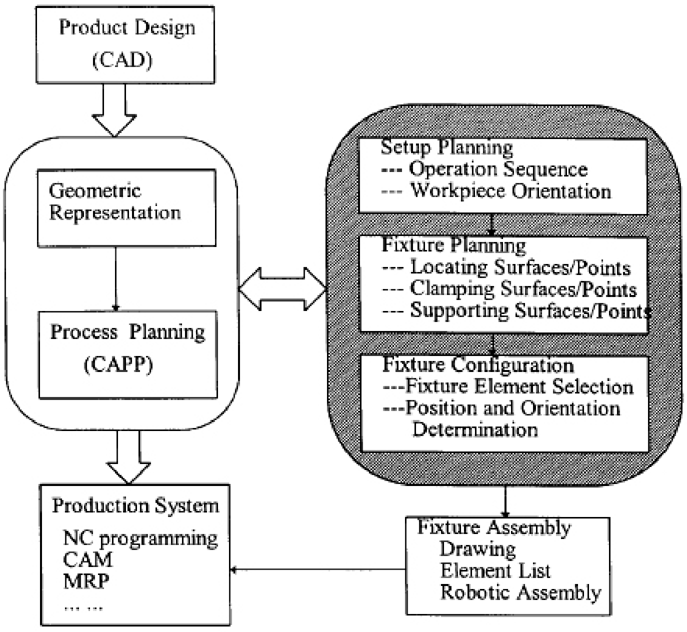


Fig.2-3. Fixture design in manufacturing systems (Rong et al, 1997)

During the development of a modular fixture design system, apart from the considerations such as: fixture configuration, interference check, etc., it is essential to build a modular fixture element database that can be incorporated into a CAD system.

Dai et al [11] addressed a new method to establish the fixture element database and construct the fixturing towers (i.e. subassemblies), which facilitates the database creation process and simplifies the use of a database in fixture assembly. The computer aided modular fixture design system (CAMFD) **(Fig.2-4)** has been developed using the knowledge-based system, Intelligent CAD (ICAD) and interfacing with UG-II for modelling the workpiece.

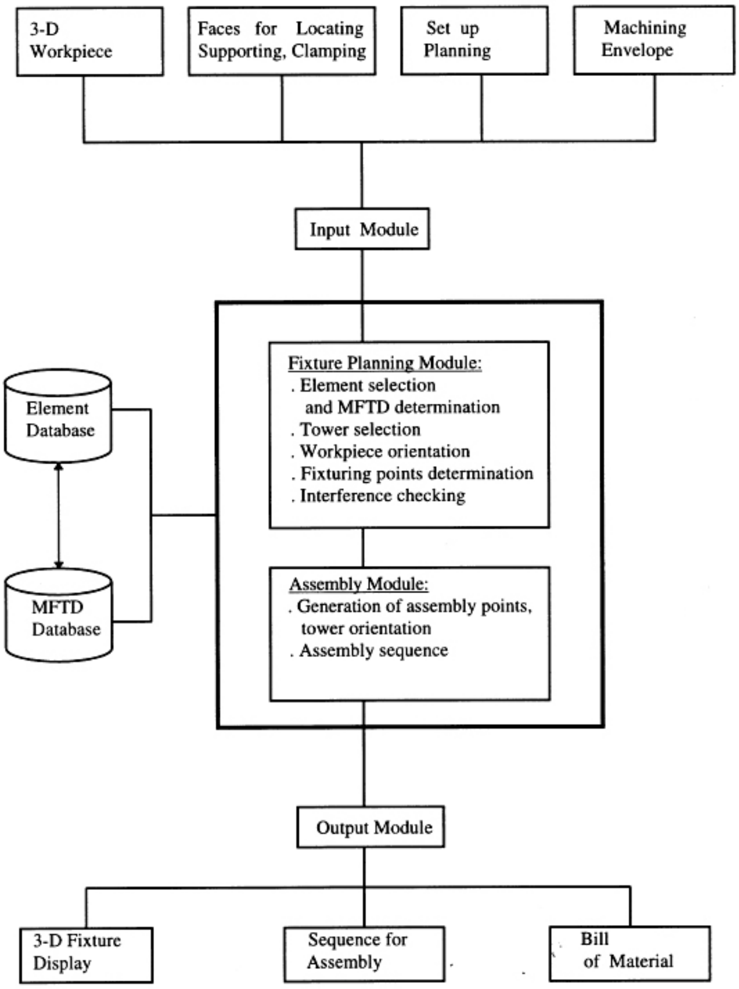


Fig.2-4. A framework of the proposed CAMFD system (Dai et al, 1997)

Along with much work being carried out to develop an efficient CAMFD system, fixture design systems can be classified into three types depending on their extent of automation; i.e. interactive class, semi-automated class and fully automated class [12]. The preliminary research was commonly conducted on the development of interactive computer-aided fixture system. Kow et al [13] in 2000 describe a CAD-based approach to develop a modular fixture design system using the Unigraphics (UG) solid modeller, integrated with a modular fixture element database, based on a hole-based IMAO modular fixturing system. A parametric modelling technique was applied to construct the database. The CAMFD system **(Fig.2-5)** adopted an integrated approach to fulfil the functions of interactive, semi-automated and fully automated fixture design combined collaboratively in a single environment. Furthermore, the interactive fixture design module allows designers to choose the fixturing faces, points and elements for fixture construction. Semi-automated and automated module fixture design modules are developed through incorporating the experience and knowledge of designers into rules and algorithms in order to automate the selection of fixturing elements and locating points.

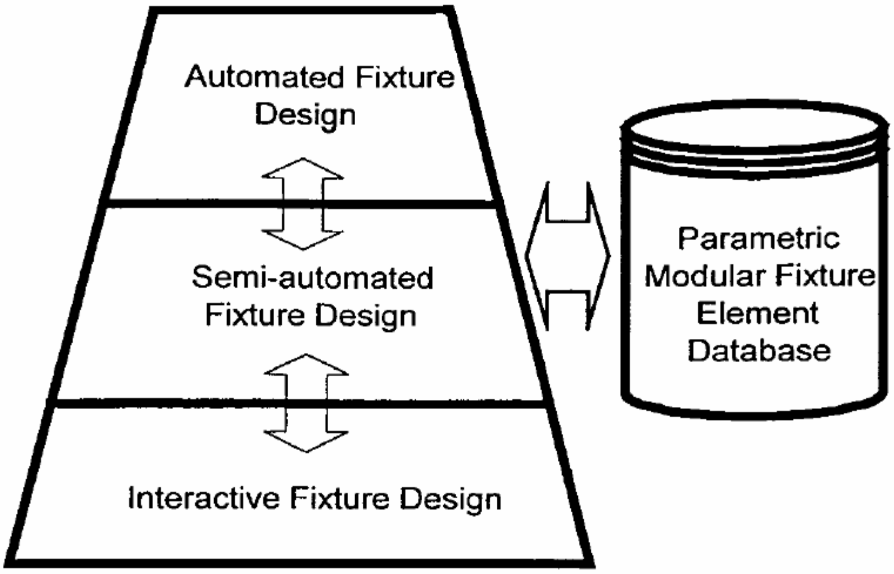

Automated Fixture Design
Semi-automated Fixture Design
Interactive Fixture Design
Parametric Modular Fixture Element Database

Fig.2-5. The structure of integrated CAMFD system (Kow et al, 2000)

More recently, research related to fixture design has centred on the micro aspects of fixture design that are referred to singular problems e.g. deformation analysis (Finite Element Analysis), stability evaluation, fixture repeatability, tolerance analysis, geometric reasoning and so on. Many fixture design systems are capable of solving singular design problems, but much effort now needs to target at integrating these systems into one [5]. Along with the research maturity, a considerable CAFD research has been conducted to develop more comprehensive functioning systems in order to support the total fixture development process.

Therefore, a comprehensive CAMFD system **(Fig.2-6)** was presented by Hou and Trappey [6], which differs from other systems. This CAMFD system consists of three modules, i.e. Fixture Data management, Fixture Element Selection and Fixture Layout Design. According to fixture feature recognition and classification, a fixture database in the Fixture Data Management module is created to maintain the fixture related data. After expertise extraction, appropriate fixture elements can be automatically chosen with respect to machining conditions in the Fixture Element Selection module. Subsequently, fixture layouts that meet manufacturing specifications are generated in the Fixture Layout Design module because fixturing locations and orientations can be identified by incorporating relevant reasoning algorithms. This CAMFD system is well constructed under AutoCAD toolkit and database management system (DBMS) software. The research mainly aims to achieve a comprehensive CAFD framework to provide efficient decision supports for fixture planning.

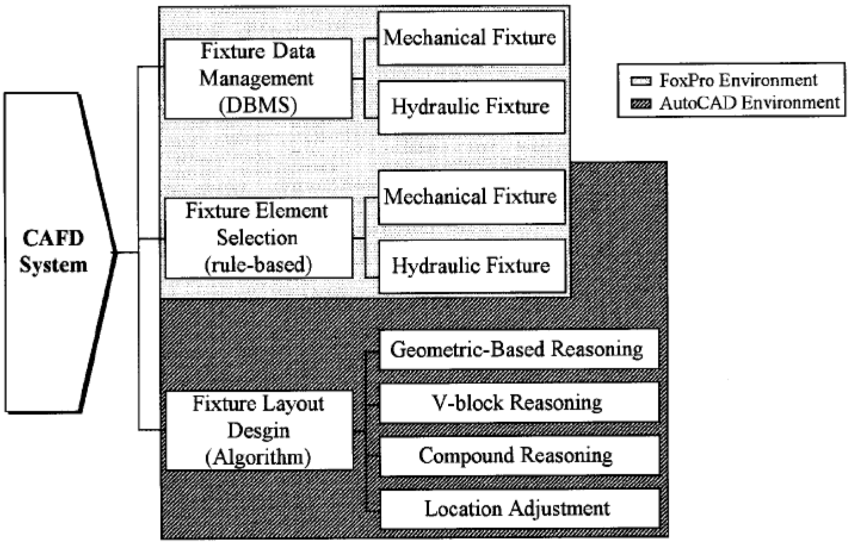


Fig.2-6. The structure of integrated CAMFD system (Hou et al, 2001)

The 3D Modular Fixture Design Expert (MOFDEX) is a comprehensive integrated CAMFD system developed by Lim et al in 1992 [7], which uses the development methodology of integrating 3D solid CAD modelling facilities into knowledge-based modular fixturing systems in order to achieve modular fixture design, pricing and inventory control. The MOFDEX system was designed to provide the functions addressed in the following headings [7].

- Definition of 3D solid workpiece using a parametric library of standardized machining feature
- Definition of machining parameters
- Extraction of workpiece and machining features
- Solicitation of additional engineering attributes
- Conversion of workpiece and machining features into object representation
- Specification of cutting tools needed
- Design (i.e. selection and positioning of element to form assembly) of fixture configuration
- Representation of design using a pre-determined CAD library of fixturing elements
- Graphical simulation and potential interference detection
- Converting modular fixturing elements into a bill material added with price list

In accordance with these functions, 17 major modules have been established within the MOFDEX system. These fixture design modules are shown below:

- Workpiece definition module
- Feature extraction module
- Macro converter module
- Object-oriented module
- Engineering attributes solicitation module
- Rule-base module
- Output formatting module
- LISP and CATIA interface module
- Fixture representation and library module
- Collision and obstruction detection module
- LISP and DBASE interface module
- Pricing module
- Pricing and inventory interface module
- Pricing and rule-based interface module
- Fixture inventory module
- Inventory and rule-based interface module
- Output from the pricing module

An overview of system layout in relation to the MOFDEX system is illustrated in the following picture **(Fig.2-7)**.

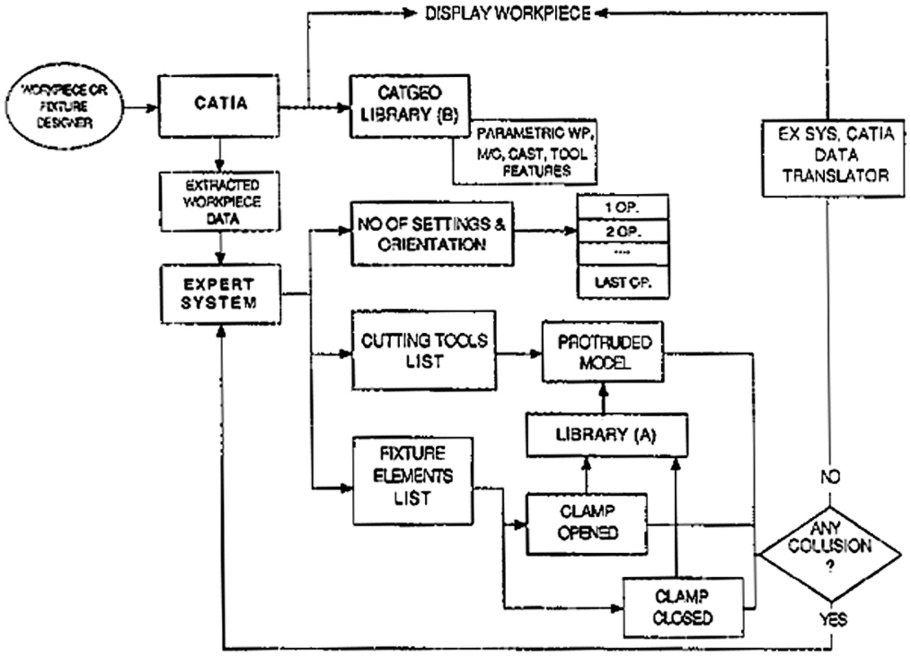


Fig.2-7. An Overview of the MOFDEX System (Lim et al, 1992)

The fixture design process is a highly intuitive process that is based on heuristic knowledge of tool designers. Since Artificial Intelligent (AI) has obtained great acceptance as a technology for manufacturing, expert systems can provide a logical tool for automating the fixture design by using the advantages of representation and management of experience-based knowledge [5].

There are a number of intelligent knowledge-based systems (IKBS) reported in the literature. Developed in late 1980s and early 1990s [14] [15] [16] [17], these systems are gradually obtaining practical acceptance for the design of modular fixtures. However, the main deficiency of these IKBS for modular fixture design is that they were developed for academic research and could not efficiently support both users and manufacturers for modular fixture design in practice.

On the contrary, a computer-aided modular fixture configuration design system, named FIX-DES [18] that was presented by Ma et al in 1998 had been developed and applied in industry. This system can be transferred into other CAD systems if necessary. As soon as fixturing requirements e.g. locating and clamping surfaces/points are determined, the fixture configuration will be automatically generated by executing the two following procedures.

A. Selecting fixture elements from a fixture element database to construct fixture units based on fixture element assembly relationships.
B. Positioning the fixture units and elements into location on a baseplate while the fixturing requirements and assembly relationships are maintained.

In addition, the fixture element assembly relationships can be formed automatically as long as the geometric models and fixturing functions of fixture elements are defined so that the FIX-DES system can be used to manage different fixture design cases. This system was developed using core programs in C and C++ and user interface programs based on a CAD environment. In FIX-DES system, the functions related to the interactive design and design modification for human involvement are also provided so as to perform modular fixture configuration design using both automated and interactive design processes [18].

2.2.5 Critical Assessment of CAFD Systems

Apart from the older CAFD research presented in section 2.2.4, this section reviews the latest CAFD systems to update the state of the art in CAFD research. A critical assessment of overall CAFD systems will also be given.

Ji et al [99] recently proposed a new approach to the computer-aided generation of fixture configuration design in manufacturing in 2007. The concept of polychromatic sets and the polychromatic set approach relating to the generation of fixture configuration design was introduced and explained. Firstly this approach identifies the relationship between fixture elements and then derives a contour matrix and its corresponding Boolean matrix. Additionally, the feasible paths and final alternatives regarding the fixture configuration design are determined using the theory of polychromatic sets. This approach was also demonstrated using a typical fixture configuration design example.

Boyle et al [100] reported a case-based reasoning (CBR) CAFD methodology called 'CAFixD' that adopts a rigorous approach to define indexing attributes based upon axiomatic design functional requirement decomposition. A design requirement is decomposed in terms of functional requirements, and the design is then reconstituted to generate a complete fixture design. The CAFixD framework and operation were presented and the indexing mechanisms were also discussed in detail.

In order to respond to questions such as: how to exchange the information and data among element drawings, element stocks and assembling modular fixture drawings which are not clear and etc, Guo et al [101] reported a virtual modular fixture management and aid-design system in 2006. This system unites the manufacture, users, owner and designers of the modular fixture on network. It not only accomplishes the scientific management of the information of the elements and fixture but also improves the efficiency of the design and shortens the period of the manufacture.

In order to satisfy the need for information exchange between the fixture design domain and other manufacturing domains, the development of appropriate information models for computer-aided fixture design systems was carried out to support integrated design and manufacturing. Mervyn et al [102] addressed a fixture design activity model in 2006 that correlates fixture design to other design and manufacturing activities. The implementation of the information models in XML and the exchange of the information models using an XML messaging approach were also introduced.

A summary of recent literature describing the development of computer-aided fixture design (CAFD) systems is given in **Fig.2-8** and reviewed below:

1. Most of the CAFD systems developed were based on the use of modular fixture design. This suggests that modular fixtures are the most suitable type of fixture design in terms of a CAFD process, which can be applied to produce a greater computerization impact in the research.

2. Although most of CAFD systems allow fixture configuration evaluation by letting designers visualize the final assembly, these systems can not adequately support the visualization or planning of the fixture assembly process [44].

3. Most of CAFD systems lack the functionality of the real-time 3D collision detection which is important to the design interference check between fixture elements, workpiece and machine tool during a fixture design process [98].

4. A great deal of CAFD research has focused on the micro aspects of fixture design that are related to the singular problems e.g. case-based reasoning, deformation analysis (Finite Element Analysis), kinematic analysis (Generic Algorithms), fixture configuration design, the information management of the elements and fixture, collision-free design and automation design etc. The shortage of research in the development of comprehensive CAFD systems is identified, which can be used to complete the entire process of fixture design, assembly process planning and essential functional analysis.

5. In order to determine the locating, supporting and claming points for fixture planning, the 3-2-1 fixturing principle has been used to deal with the prismatic shape of workpiece in a great amount of CAFD research as the essential rules [6].

6. Simulating physical properties for fixtures in use is still the research bottleneck for existing CAFD systems. They should be taken into account throughout the process of fixture design and assembly evaluation. This is particularly true for modular fixture design process.

7. Very few CAFD research adopts interactive VR technology to support the conventional fixture design and assembly process, although the advantageous features of VR have a greater potential of making the fixture design in a natural and instructive manner, providing better match to the working conditions, reducing lead-time, and improving fixture productivity and economy. These VR features refer to realistic visualization, high-level of interactivity, and physical property simulation etc.

8. Although much CAFD research has been implemented, fixture design still continues to be a major bottleneck in the integration of CAD and CAM activities. Future fixture design systems will focus on methods which not only serve as a bridge between CAD and CAM but also seek to support cross-functional participation and concurrent engineering approaches.

Authors	Research purpose	System features	Fixture type	Developing Method	Process	Workpiece	Year
Shokri, M., Arezoo, B. [96]	Planning of fixture configuration and assembly procedure	The assembly algorithm was developed	Modular fixtures	VC++ / Solidworks platform	Inspection /CMM measuring	Complex geometry	2008
Kang, Y.G., Wang, Z., Li, R., Jiang, C. [97]	Fixture configuration design / networked manufacturing	Rapid configuration designing, 3D modeling, a elements database	Modular fixture	A hybrid CBR/KBR fixture design method	Machining	Complex geometry	2007
Ji et al [99]	The generation of fixture configuration design	Identify the relationship between fixture elements/ Derive a contour matrix and its Boolean matrix	Modular fixture/ dedicated fixture	The polychromatic set approach	Machining	Prismatic shape	2007
Peng, G.L., Liu, W.J. [98]	Developing a desktop VR system/ support the decision making for fixture design	Hierarchical data model for fixture assembly / machining simulation	Modular fixture	Interactive VR technology	Machining	Complex geometry	2006
Boyle, I.M., Rong, K., Brown, D.C. [100]	To develop a CAFD methodology/ CAFixD	Indexing attributes based upon axiomatic design functional requirement decomposition	Modular fixture	A case-based reasoning (CBR) method	Machining	Prismatic shape	2006
Guo, H., Yan, X.G., Li, J.W. [101]	To manage the information of the elements and fixture/ improve the efficiency of the design	This system unties the manufacture, users, owner and designers on network	Modular fixture	No information	Machining	Complex geometry	2006

Mervyn, F., Kumar, A.S., Nee, A.Y.C. [102]	Information exchange between the fixture design domain and other manufacturing domains	A fixture design activity model that correlates fixture design to other design and manufacturing activities	Modular fixture/ dedicated fixture	XML/XML messaging approach	Machining	Complex geometry/ prismatic shape	2006
Hou, J.L., Trappey, A.J.C. [6]	To introduce a comprehensive CAMFD system	Fixture Data management / Fixture Element Selection / Fixture Layout Design	Modular fixture	AutoCAD/ database management system (DBMS)	Machining	Complex geometry/ prismatic shape	2001
Kow, T.S., Kumar, A.S., Fuh, J.Y.H. [13]	A tool-collision-free fixture design/ a fixture design system	Interactive/ semi-automated / fully automated fixture design	Modular fixture	Unigraphics (UG)/ a fixture element database	Machining	Prismatic Shape	2000
Dai, J.R., Nee, A.Y.C., et al. [11]	Automating modular fixture design	A new method to establish the fixture element database/ the fixturing towers	Modular fixture	Knowledge-based system/ Intelligent CAD (ICAD)/ UG-II	Machining	Prismatic Shape	1997
Lim, B.S., Imao, T., Yoshida, H., et al. [7]	To develop a comprehensive integrated CAMFD system	Modular fixture design/pricing/ inventory control	Modular fixture	3D solid CAD modelling/knowledge-based modular fixturing systems	Machining	Prismatic Shape	1992

Fig.2-8. A Summary of Literature in the Development of CAFD Systems

In order to narrow CAFD research gap of using interactive VR technology to support the fixture design and assembly, the research in this thesis aims to develop a useful VFDAS system. It was considered that the VFDAS system would provide the function of real-time 3D collision detection in accuracy to improve the design interference check for a fixture design process. The VFDAS would also be able to simulate essential physical properties, e.g. mass, gravity, friction, applied force, elasticity and toppling, so that the fixture design process could be realistically carried out as if in the real physics world. In addition, the VFDAS system would also improve the visualization and planning of the fixture assembly process.

2.3 Attributes of VR

2.3.1 Immersion

Immersion is a key feature of VR technology. This terminology describes the feeling of a VR user, that his virtual environment is real. A high degree of immersion is equivalent to a realistic or “believable” virtual environment [19].

The term “Immersion” could be also defined as the sensation of being immersed within a virtual environment (VE). [20]

There are several factors which detract from the experience of immersion reported by Gabriel Zachmann. They are indicated as: Feedback Lag, Narrow field-of-view, a monoscopic view and low display resolution in order of significance [19]. In addition, the factor of feedback lag is considered as the most important effect with regard to immersion effectiveness, which is represented in related psychological experiments.

On the other hand, some VR systems provide each user with a personal view of the virtual environment using an HMD (Head Mounted displays) in **Fig.2-9** which visually isolates them from the real world. The user can acquire a positive sense of being immersed in the VE, which is further enhanced when touch and sound are introduced.

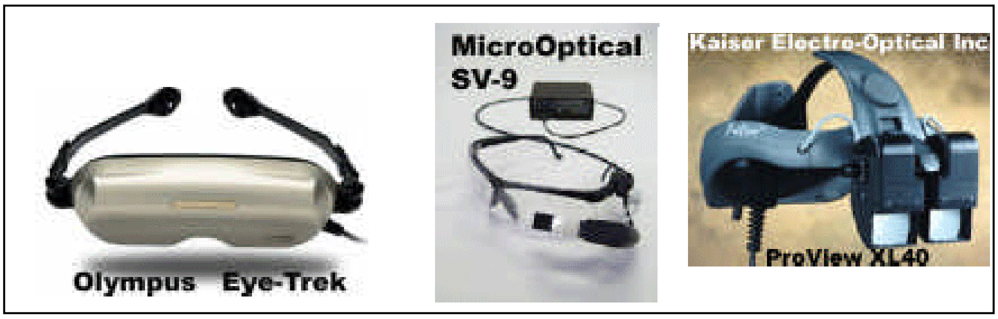


Fig.2-9. Typical Examples of the VR HMD

Much of the discussion emphasized the significance of fidelity of a virtual environment and visual ‘immersion’ of the user (e.g. by using a head mounted display or CAVE display system) and assessment of how well these support user performance in VR/VE tasks [21][22]. Uses of immersive VR for training workers in industrial processes have been suggested in the literature. Chryssolouris et al reported an immersive VE for training in manual assembly operations [23] and also presented a virtual machine shop as a planning and training tool for machining processes [24] such as: machine tool setup and NC part program execution. In addition, Mavrikios et al reported in 2006 that a VE was developed to support training in manual welding processes using immersive VR [25].

However, due to various aspects of VR environments that need further research and development, e.g. representation speed, presence, real-time interaction and response, realistic visualization and expensive costs, full immersive VR simulation systems are rarely employed in real industry [26][27]. The lowest level of VR systems are desktop systems which provide only a monitor-based viewing of virtual objects. The features of desktop, non-immersive VR systems are far from the possibility of immersive VR technologies, but the main advantage of desktop systems is that standardised computer techniques can be applied [28]. As a result, desktop, non-immersive VR systems have greater applicability for industrial uses compared to immersive VR technologies. The advantage of desktop-based VR systems is to provide a standardised method to support most of common computers even including the lowest level of monitor-based computers and this may help to spread the VR application in industry.

To develop the framework of the VFDAS system, the author started from exploring the lowest level of desktop-based VR system in this thesis to reduce costs and maximise benefits at the preliminary research stage of the proposed VFDAS.

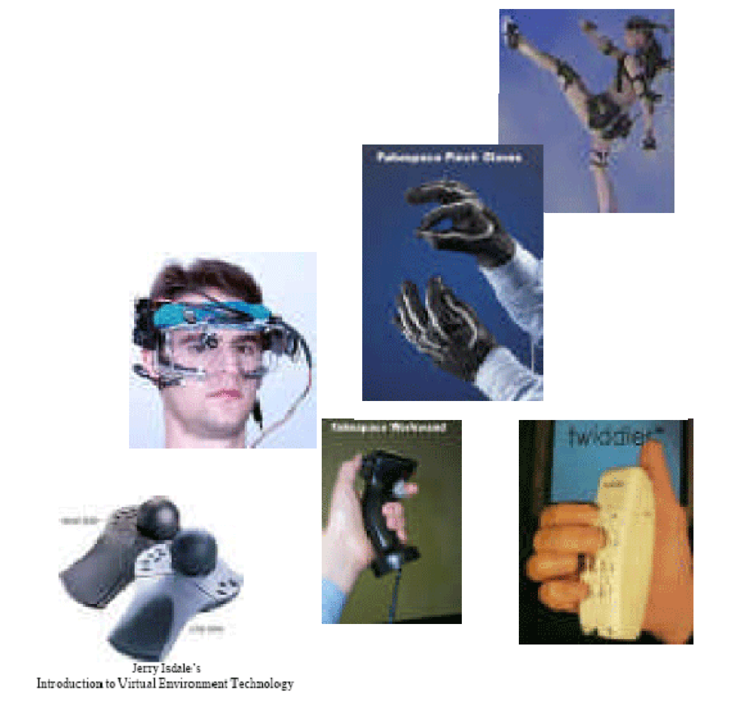
2.3.2 Presence

While immersion is an objective measure, presence is the subjective sense of being in the virtual environment (VE). Presence requires a self-representation in the VE. For instance: a virtual body (often, only the hand is represented in the VE, since the whole body is not tracked). Presence also requires that participants can identify the movements of a virtual body represent his/her movements [19].

In addition, Sheridan defined presence as the sense of being physically present with the visual, auditory, or force displays generated by the computer [29]. However, Steuer addressed a simpler definition in 1992 to the whole concept by introducing that presence is the "feeling of" being in an environment. Steuer also proposed that if a feeling of presence is not high, then users become detached from the environment, thus resulting in a decrease of performance [30]. This may be true but other factors can also affect performance, e.g. frame-rate, the screen resolution or the VE layout itself [31].

2.3.3 Interaction

Interaction is described as any action of the user aiming to modify or probe the virtual environment [19]. In order to achieve a good degree of immersion, it is necessary to develop interaction techniques that are as intuitive as possible. Conventional interaction devices such as: key board, mouse, tablet, etc. that are employed with most VR applications are not adequate enough for natural interaction. The main drawback of these is their low number of input dimensions, maximally 2. However, new advanced devices e.g. SpaceMouse, DataGlove, tracking systems, Boom, Wands, etc **(Fig.2-10)** provide 6 and even more dimensions that allow efficient, natural interaction techniques.



Jerry Isdale's
Introduction to Virtual Environment Technology

Fig.2-10. SpaceMouse, DataGlove, Tracking Systems, Wands

2.3.4 Autonomy

Autonomy reflects the extent to which the environments function on their own, without (and sometimes in spite of) user input. Systems with low autonomy, like many tutorials and practice programs, sit doing nothing until students enter an answer to a question or click on a navigation icon. Autonomous environments, on the other hand, follow their own goals, evolve and develop whether the user does anything or not. Real-time simulations and many games fall into this category [32].

2.3.5 Zeltzer's Triangular System and AIP Cube

Zeltzer suggested three definitions in the following to demonstrate the general attributes of VR. Zeltzer [33] proposed a framework for describing the attributes of VR. He proposes that computer-based environments can be characterized along three dimensions which he calls autonomy, presence and interaction.

Autonomy: Objects must react to external stimuli, have collision boundaries and exhibit real-world effects (e.g. coefficients of restitution, gravity and friction).
Interaction: One must be able to manipulate the parameters of each object in real time.
Presence: A crude measure of the fidelity of the viewing system [33]

To create a reliable representation of our world in VR, three important attributes of VR are raised and their correlation **(Fig.2-11)** is perceived, which must be considered while constructing a virtual environment.

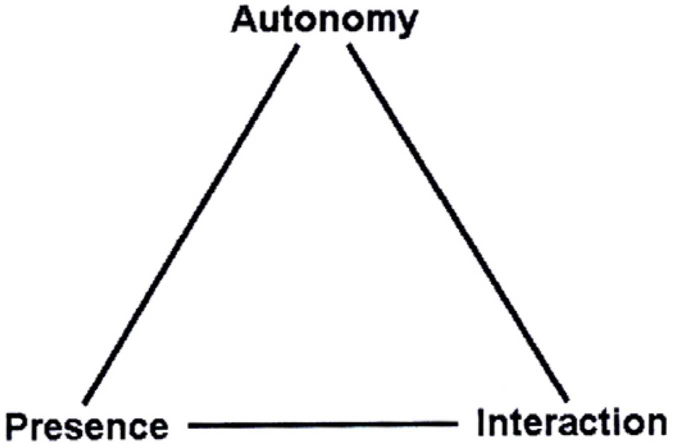


Fig.2-11. VR Triangular system

The correlation of VR attributes is further explored with the development of Zeltzer's AIP cube in **Fig.2-12.** Here there is the situation where each corner refers to a different level of VR.

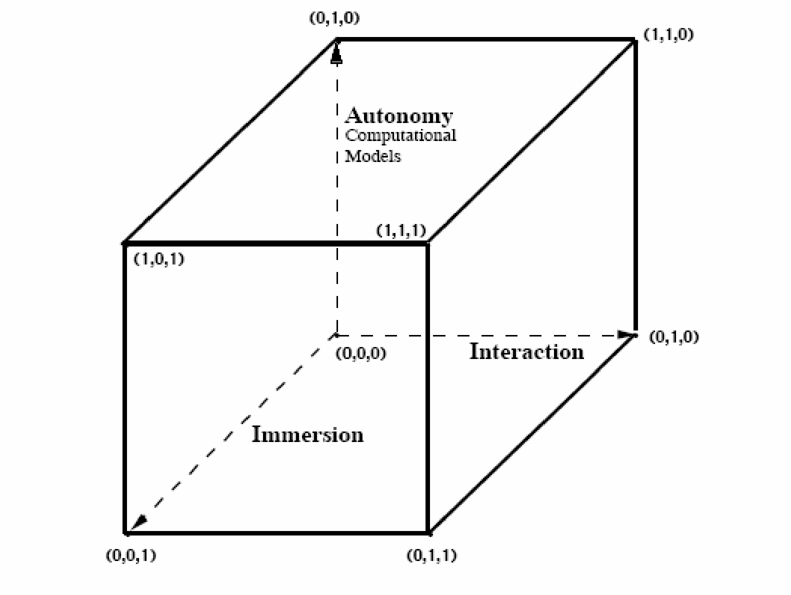


Fig.2-12. Autonomy, Interaction, Presence in VR-Zeltzer's Cube

2.4 VR Applications in Education and Training

Virtual Reality (VR) applications have great potential for use in education at all levels. VE learning tools for education have the potential and additional values to complement existing approaches in education. Learners can be offered an opportunity for active learning with three-dimensional representations, multiple perspectives and frames of reference, simultaneous visual and auditory feedbacks [34].

Despite all the advantages of VR educational tools, they should not be used to replace lecturers and classes in their entirety. Instead they should be determined as a supplement rather than a replacement.

2.4.1 Ten Steps to Developing Virtual Reality Applications for Engineering Education

A number of VR projects have been conducted in the department of chemical engineering at the University of Michigan for several years to develop a series of VR based computer modules for use in undergraduate engineering education [35]. Three primary goals were stated:

1. To produce modules with practical use to as many students as possible.
2. To determine the applicability of virtual reality to engineering education.
3. To develop a knowledge base of techniques for display of, and interaction with, scientific and technological information and concepts in virtual world, that can later be applied to practical engineering problems.

This research also outlines their findings regarding the second goal, which includes not only how to produce effective VR applications, but also the identification of which topics most benefit from VR, and how best to incorporate educational VR into the engineering curriculum.

Along with their research work, ten key steps [35] regarding the development of VR-based education modules are summarized by Bell et al in 1997 which can be considered as the useful guidance to carry out VR application research in education:

1. Understand the strengths and weakness of educational VR.
2. Identify the intended audience and the end user's probable equipment.
3. Identify an application that is suitable for VR.
4. Choose an appropriate development platform.
5. Consider carefully the trade-off of simulation realism versus performance, and plan out the simulation.
6. Start with a simple framework, and then gradually add details.
7. Provide for student evaluation early and often; Develop the simulation based upon user feedback.
8. Prepare instructions suitable for students, faculty, and systems administrators.
9. Incorporate the simulation into the curriculum.
10. Share your results far and wide.

2.4.2 Use of VR in Civil Engineering Education

Although a great deal of research is carried out in relation to Computer Aided learning (CAL) and the enthusiasm is shown for VR as a potential educational tool, there are few actual VR applications developed for use in college [36].

Messner et al mentioned a concept called 4D Computer Aided Design (CAD) modeling **(Fig.2-13)**, in which 3D CAD with schedule time as the 4th dimension is taken into account for a discussion [37]. To improve construction education by using VR, the 4D CAD modeling of construction processes is applied to the undergraduate architectural engineering program. A tool that allows construction engineering students to interactively create the construction sequence for a project in an immersive environment was developed to evaluate the use of immersive VR. During the process of experiments, a comparison between VR tools and conventional project planning methods e.g. use of 2D drawing and Critical Path Method (CPM), was conducted to explore the capability of VR for supporting students' conceptual understanding with regard to design and planning on construction projects. The conclusions from these experiments demonstrated positive results in the use of VR for architectural engineering education. The results indicate the students can understand construction plans much better when the visualization tools are used. The results also indicate the students can gain rich experience by developing and critiquing construction schedules in a full-scale virtual environment [37].

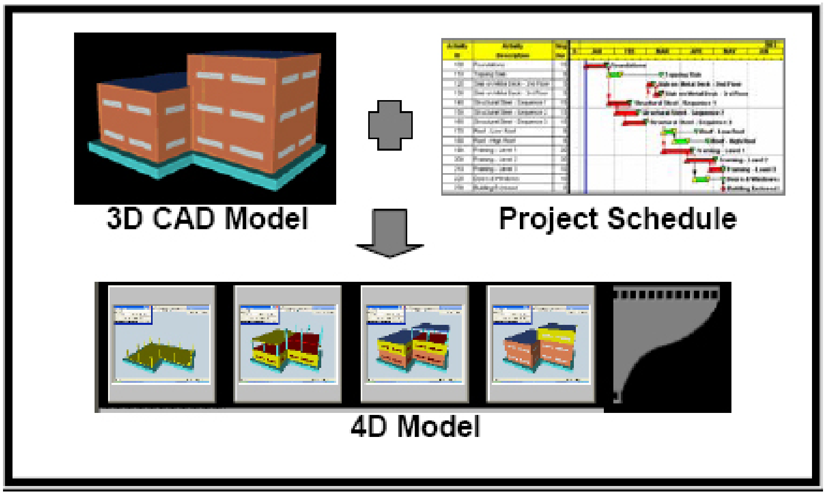


Fig.2-13. 4D CAD modeling in VE used for architectural engineering education

Moreover, Kalisperis et al (2002) introduced another similar project via which architecture students' perception and the usefulness of various systems for different tasks were evaluated using a usability study. The final conclusion indicates VR techniques can improve the students' perception and convey relevant information to students more efficiently with less misrepresentation comparing with traditional techniques [38].

2.4.3 Use of VR in Manufacturing Engineering Education

Due to the low level of necessity with regard to spatial perception and visualization quality, VR applications in Manufacturing Engineering education are relatively rare compared to Construction Engineering education. In addition, HTML is the standard for writing webpages [39]. Most of web-based interactive teaching packages developed with the combination of Java applet and HTML attempt to simulate the virtual manufacturing process for supporting manufacturing education. The VE educational tools that are established using Java technology based on web pages can not perform genuine VR attributes during students' learning process. In general, these types of VEs are referred to "2½ D representations" which only provide a part of characteristics of real virtual environments.

A VR research project carried out in the Manufacturing Division at the National University of Singapore aimed to explore and develop a web-based interactive VR teaching package that provides interactive virtual environment for a module on automated machine tools. Virtual reality simulations were developed and used in the teaching materials to enhance the student understanding of complex concepts during automated machining process. It was helpful in teaching automated machine tools which operate with the numerical control (NC) of motions. These VR simulations provide great capability of training students in NC programming and operations without the need to work on real NC machines in the laboratory [34].

The essential solutions for developing this web-based VR teaching system were to produce integration amongst Java, VRML (Virtual reality modeling language) and HTML so as to achieve interactive VR simulations for the purposes of manufacturing education. The VR simulations of the NC machining operations were created using VRML and Java. VRML was used to build the virtual NC machining environment. Java applets are used to provide preset animations. Interaction between Java and VRML can be generated in VR simulation through applying user inputs into the external authoring interface (EAI).

Sackett et al commented that a new engineering college graduate must develop a holistic view of total business process from design to production to delivery in order to succeed in this dynamic environment [40]. To address the manufacturing educational need of new engineers, a collaborative learning network called the Virtual Factory Teaching System (VFTS) was developed to provide a workspace that illustrates the concepts of factory management and design for complex manufacturing systems. The VFTS is unique in its integration of four domains: web-based simulation, engineering education, the Internet, and virtual factories [41].

2.5 VR Applications in Industry

Due to the today's growing competition environment, manufacturing companies are subject to employ new computer technologies to reduce production costs, increase manufacturing productivity, improve product quality and shorten design-to-manufacture cycle for new product development. During the last two decades, Virtual Reality (VR) has proved its potential for the visualization, interactive simulation, manipulation of complex industrial design and production data. VR is beginning to be applied to cope with the realistic issues of industrial design and manufacturing planning in order to create the manufacturing flow and product data management through the whole production process. As a result, VR simulation technology for human-computer interfaces is integrated into computer aided design (CAD) and computer aided planning (CAP) to enhance their deficiency in application.

2.5.1 VR for Industrial Design and Assembly

Virtual Prototyping (VP) is a challenging class of applications for VR. A VR system suitable for VP must be able to handle geometric complexities, which can not be reduced by common rendering approaches. In order to ensure a feeling of presence and immersion, the VP system must be able to achieve a frame rate of no less than 15 frames per second at any time [19].

The visualization system [28] used for product design and planning presents a new idea for the integration of CAD and VR in industrial design and manufacturing processes. Although it is such a unique approach for integration, VR visualization techniques were properly used in product development. However, interaction was not employed in this project. Finally, the system for VRML model visualisation that enables changes in the configuration file, which is written in XML, was established.

Steffan et al [42] described a process for integrating an interactive VR-based assembly simulation of a digital mock-up into an existing CAD/CAM infrastructure. The potential for VR assembly simulation to improve the CAD/CAM procedures is outlined. The satellite assembly simulation **(Fig.2-14)** was created within the REALAX^{TM} VR authoring tool to verify whether a mirror could be dismantled without removing the optical monitor tube or not.

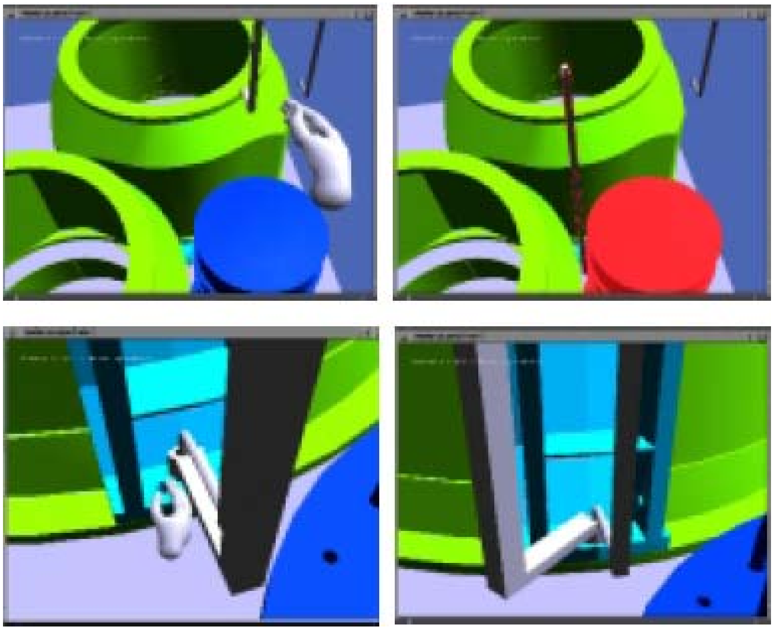


Fig.2-14. The interactive satellite assembly simulation (Steffan et al, 1998)

The virtual design and assembly system (VDAS) [1] is a VR-based engineering application which allows engineers to design, modify and assemble mechanical products and the research was carried out by the Hong Kong Polytechnic University. The VDAS comprises with the functionality of assembly planning and interactive manipulations within the immersive virtual environment e.g. part modification, assembly plan verification and modification. In this VDAS, several interactions with virtual objects that a user can perform are presented below:

(1) Modify the geometry of virtual objects by changing one or more variables.
(2) Add new features to virtual objects.
(3) Execute a Boolean operation on two virtual objects.
(4) Simulate the assembly with the assembly planning result.
(5) Edit assembly trajectories and part sequence.

The prototype of VDAS has only fundamental functions that are required during a virtual design and assembly process, which have not yet focused on the investigation of a specific application related to the industrial assembly such as: fixture design & assembly, cable harness design & assembly or automobile design & assembly. This research to date is still a framework of VDAS without any specific application in mind.

As Dewar et al explained, traditional automatic assembly planning related to disassembly process has to depend on the assumption that if you can disassemble a part, you can assemble it, and vice versa. It may not illustrate the true situation in a real physical world. The plan for disassembly might not represent the best one for assembly [43]. The development of VR applications in engineering provides many solutions to solve these problems. An engineer can now perform the assembly intuitively in the virtual environment (VE) using both VR hardware and software. The information gathered via this process is applied for the assembly planning and verification. The Virtual Assembly Design Environment (VADE) **(Fig.2-15)** is a VR-based engineering application that allows engineers to plan, evaluate, and verify the assembly of mechanical systems, which was developed at Washington State University in collaboration with NIST (National Institute of Standards and Technology) [44]. The system enables exporting necessary data to VADE through a user-selected option as soon as mechanical assembly models are created in a parametric CAD package such as: Pro/ENGINEER. Using VADE, VR users can then carry out the assembly process with their hands and virtual assembly tools e.g. screw driver, wrench, and so on in order to allow users to make decisions, design changes and perform other engineering tasks. Furthermore, the virtual hand that is connected with an instrumented glove device i.e.

CyberGlove was employed. Several interactive VR functions in the VADE related to the design and assembly simulation are accomplished, which are shown below [44]:

- Real-time collision detection
- Simulation of dynamic behaviours of the parts held in the user's hand
- Dynamic interactions between the user, the parts, the base part, tools, and the environment objects
- Simulation of ballistic motion of the objects in space
- Simulation of dynamic behaviours of the parts while constrained on the base part

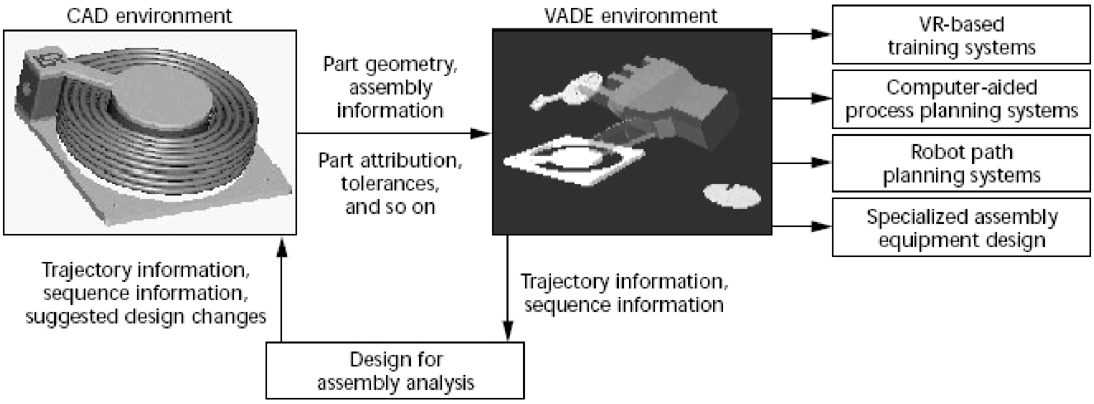

CAD environment
VADE environment
Part geometry, assembly information
Part attribution, tolerances, and so on
VR-based training systems
Computer-aided process planning systems
Robot path planning systems
Specialized assembly equipment design
Trajectory information, sequence information, suggested design changes
Trajectory information, sequence information
Design for assembly analysis

Fig.2-15. A VADE application scenario (Jayaram et al, 1999)

Likewise, another typical research that addresses the similar issues with these arising from the VADE project was conducted by Kaufman et al in 1996 [45]. In general, the primary research direction related to the issues of VR assembly refer to automatic assembly planning, collision detection, physically based modelling, swept volumes and dynamic simulation etc [45, 46]. For example: the Archimedes system is an interactive assembly planning system that automates several procedures of the planning process, which provides a platform for users to create an assembly plan, view a simulation, add constraints, and iterate design until acceptable solutions are identified [45].

Virtual assembly (VA) based on VR technology is a key technique of virtual manufacturing. As a key application of VR in virtual manufacturing, VA provides not only an interactive interface for the virtual product design, but also an efficient method to verify assembly performance of these products [47]. Interactive VR assembly simulation in an immersive or a semi-immersive VE is able to carry out the evaluation of products to detect the potential design interference before the actual assembly process. To develop a good VA system, the interdisciplinary expertise would be required such as: CAD modelling, information handling, real-time collision detection in accuracy, human-machine interaction etc.

An applicable Virtual Assembly (VA) system must include several advantages such as: high efficiency, real time and good interactivity so that it can provide a platform to carry out assembly operations e.g. path planning, process planning, sequence planning and assembly simulation, etc. To develop a useful VA system for industrial use, three essential techniques are identified and introduced by Liu et al [48], which are shown below:

- **Data transformation from CAD system to VA system**
  A. So far the main modeling method for the VA system is to use 3D CAD software e.g. SolidWorks, Pro/Engineer or Unigraphics etc.
  B. There does not exist a data exchange standard between CAD systems and VA systems.
  C. The approach to transform necessary data from CAD system to VA system is still a problem in VA construction.

- **Collision detection**
  A. Virtual assembly operations in VE including path selection, sequence planning and virtual hand gripping are fulfilled due to the collision detection simulation provided by of VR.

- **Virtual manipulation with a virtual hand**
  A. An instrumented glove provides VA systems a natural method for human-machine interaction. An intuitive and realistic virtual manipulation for users is an important requirement of a good VA system.

Liu et al [48] described a new technique to transform topology information by constructing a five-hierarchy topology structure (FHTS) **(Fig.2-16)**. The geometry information was transformed by a data transformation interface (DTI). The assembly information was transformed by using database technology. In particular, the key algorithm applied for VA collision detection is called object-oriented collision detection algorithm (OOCDA) which is put forward during this research.

Liu et al also [48] presented serial toolkits and programming languages that were applied to this study for constructing the VA system. SolidWork^{TM} was used as the platform for assembly modeling. World ToolKit^{TM} (WTK) was used as the platform for VA. SQL Server was selected as the platform of database and Visual C++ was selected as the platform for application programs.

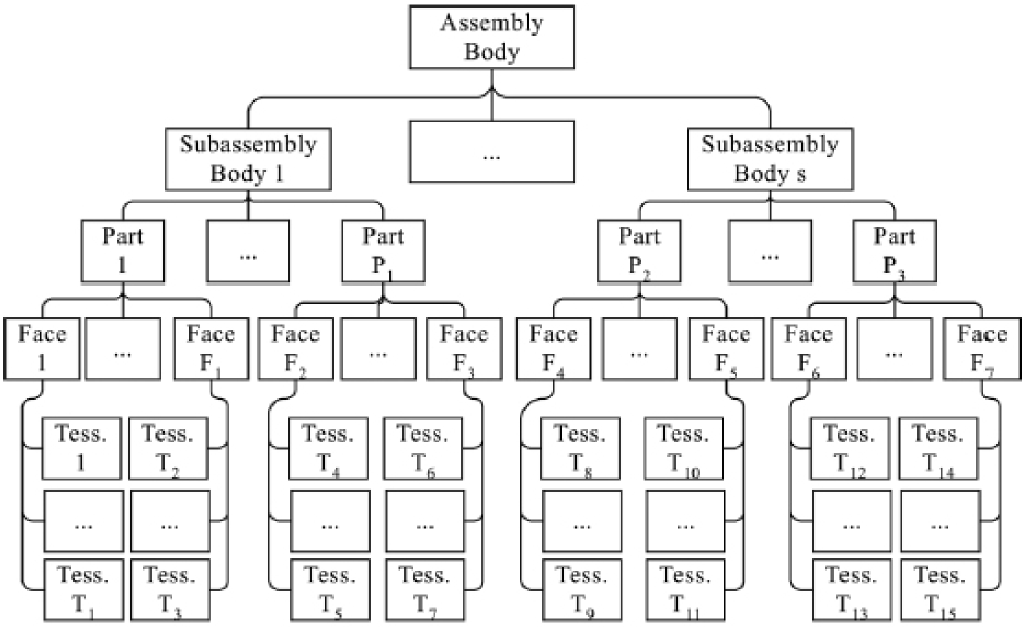


Fig.2-16. Five-hierarchy topology structure (Liu et al, 2004)

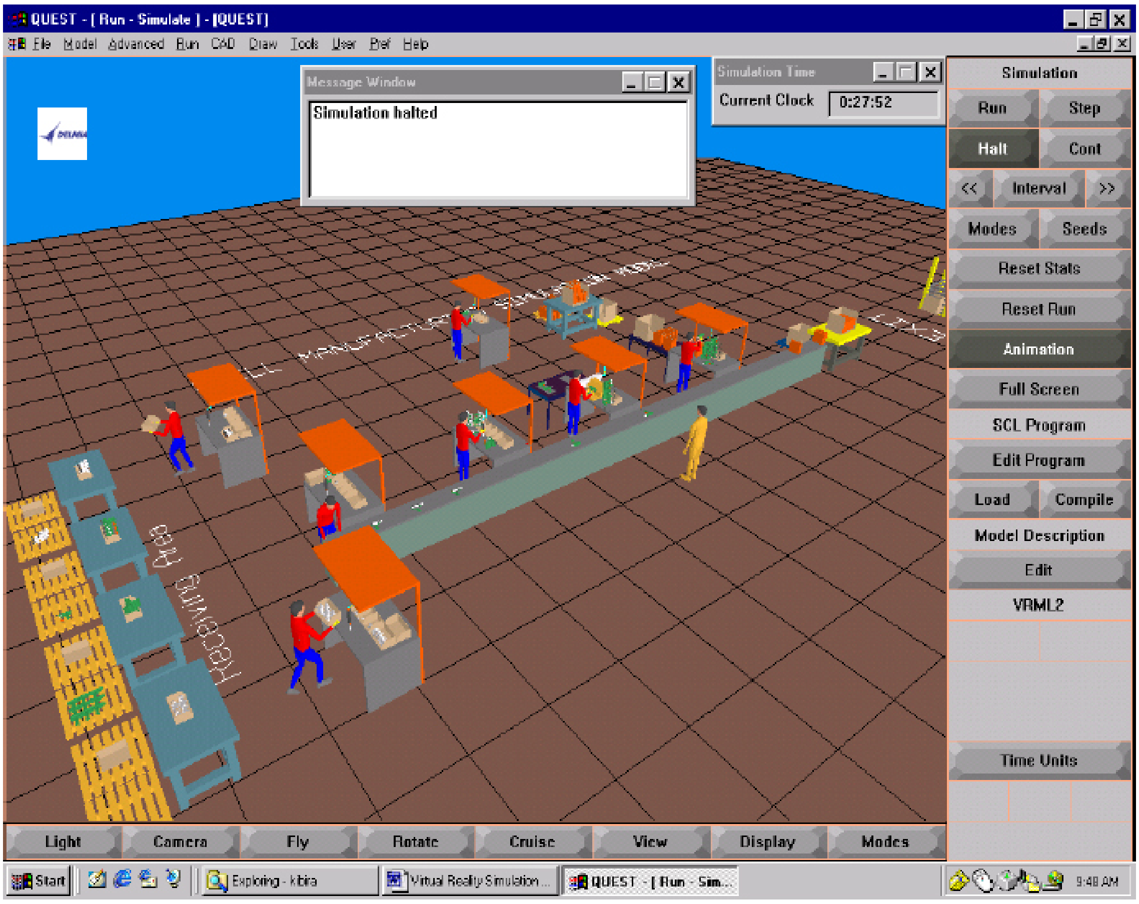
The "Virtuelle Werkstatt" [49] study was carried out to develop a demonstration platform in relation to VR based prototyping, for which the research methodology, the integration of previous research results in multimodal human-computer interaction and immersive CAD system were exploited by researchers. The main objective of the proposed VR system aims to provide users a facility for a natural, multimodal communication in a virtual construction environment in order to achieve the manipulation and interaction intuitively controlled by human speech and gestures. In particular, the multimodal interaction can be fulfilled by the combination of VR and advanced Artificial Intelligence (AI).

The mainstream research of VR simulations for supporting manufacturing focuses on dynamics of the current system and how it can be improved by additional equipment or better scheduling and a resources allocation system [50]. VR Simulation research is also engaged with the development of visual interactive simulation systems to act as a medium of communication between specialists from different domains. Albastro et al [51] mentioned that VR simulation could be the best solution in decision making during the design of a manufacturing system.

A good example of VR simulation for supporting manufacturing process is the application of VR simulation to the design of production line for a mechanically-assembled product **(Fig.2-17)**, which was performed by Kibira et al in 2002 [52]. The capability and uses of this VR simulation system regarding the issues from a small assembly line are outlined below:

- Support the definition, design and actualization of overall system
- Evaluate different decision options for the design or reconfiguration of the production line
- Generate cost and performance data for operations that can not be calculated in a straightforward way
- Demonstrate and visualize the operation of the line

The synergistic software applications provide a platform upon which the VR simulation of manufacturing design is constructed. The Quest^{R} package is an object-based, discrete event simulation tool and employed in this VR simulation [52]. It is used to model, experiment, and analyze the facility layout and process flow as well as provide the capability of visualization and data import/export. IGRIP^{R} and the ERGO packages are used for the script development and animation of work cell operations; AutoCAD, a computer aided design application is used for object modeling.


Message Window

Simulation halted

Simulation Time

Current Clock 0:27:52

Fig.2-17. VR Simulation for Production Line (Kibira et al, 2002)

2.5.2 VR for Other Industrial Applications

VR is a well-known technology which is being investigated for practical uses in various industrial domains. A few application systems that are developed for other industrial uses will be presented in the following paragraphs.

A relax/fresh system incorporated with VR technology is employed to deal with the issues existing in the health care field. This VR system is integrated with a physiological feedback mechanism in which the VR simulation can be generated by detecting the sequential data of the users' heart rate [53].

Another VR-based system that simulates the effects of horseback riding therapy was specifically developed to rejuvenate both the patients' mind and body. It may also be enjoyed by healthy senior citizens [54].

Furthermore, a VR-supported kitchen layout design system has a capability to provide customers with a pseudo-experience of the kitchen layout and modify its design intuitively. It also assists customers to make decisions when designing the layout of their new kitchen. The VR system has been used for practices in Matsushita's showroom since 1994 [55].

Simmons et al addressed a novel use of immersive VR for the design of cable harness. A VR system named CHIVE (Cable Harnessing In Virtual Environment) was established and the functionality of the VR system is reported [56]. The initial research shows that more considerable productivity can be achieved in comparison with the way of using conventional desktop CAD system. In general, the immersive CHIVE **(Fig.2-18)** system is useful and promising in supporting the actual routing process of cable harness as an interactive design tool.

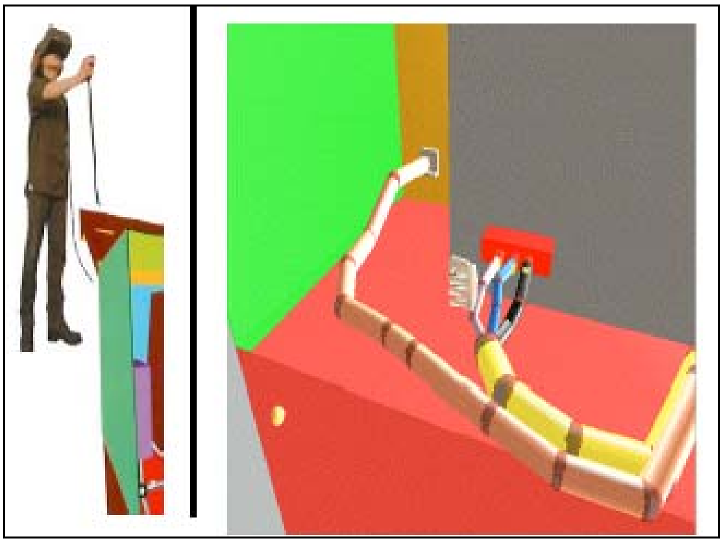


Fig.2-18. A user utilizing CHIVE for routing a cable harness in the virtual world

Murray [57] presented another possibility of using VR simulation to eliminate unforeseen design errors for large scale mechanical products, which result in subsequent problems in relation to maintenance operations. The simulation of maintenance operation allows that the relevant maintenance issues may be identified at the early design stage. The assessment of assemblability and maintainability of mechanical products can be carried out using the virtual prototyping environment that is presently under the further development.

2.6 Summary

Whilst there is an increasing use of computer-aided systems for fixture design, and recent interest in the potential application of interactive VR systems in industry, there is very little evidence of research exploring the use of VR for fixture design and assembly. It is considered that the interactive features of VR could provide benefits to improve the fixture design and assembly process. This PhD aims to fill this research gap by exploring the feasibility of developing VFDAS.

Chapter 3. Methodology for VFDAS Design and Development

3.1 Introduction

Fixturing is an important manufacturing activity in the production cycle. The computer aided fixture design (CAFD) technique has become part of CAD/CAM integration and facilitated efficient workholder design [58]. Virtual Reality (VR) technology and its benefits are employed to enhance the CAFD systems' capability so as to simplify the conventional fixture design process. As a result, the VFDAS system originated from the concept of adopting VR technology into fixture development, was developed as the primary research deliverable in this thesis.

The research in this thesis is a cross-disciplinary project which relates to various research aspects such as: fixture development, interactive VR programming, computer aided design (CAD), the integration of VR and CAD and usability design. Consequently, a series of different approaches relating to these research aspects are explored and identified. These individual approaches jointly compose an incorporated methodology that was applied for the development of VFDAS.

In fixture design, the common locating rule in practice is the 3-2-1 principle, and is described in section 3.2.3 and this was applied in VFDAS. The principle related to 12 degrees of freedom (DOF) for the workpiece in 3D was taken into account during the VFDAS construction. This is explained in section 3.2.4. In addition, the integration method between VR and CAD that was identified and used for coping with the difficulty of CAD model conversion, is explained in section 3.3. The interactive VR programming method based on Virtools Dev 3.0 is described in section 3.4.2 in order to realize VR scenarios, interactivity and physical property simulation in VFDAS. Finally, the overall technical procedures applied to develop the VFDAS system are summarized to produce an overview of the complete practical methodology.

Fig.3-1 represents the structure of chapter 3 and how a series of different approaches relating to various research aspects jointly form an incorporated methodology which was applied for the development of VFDAS.

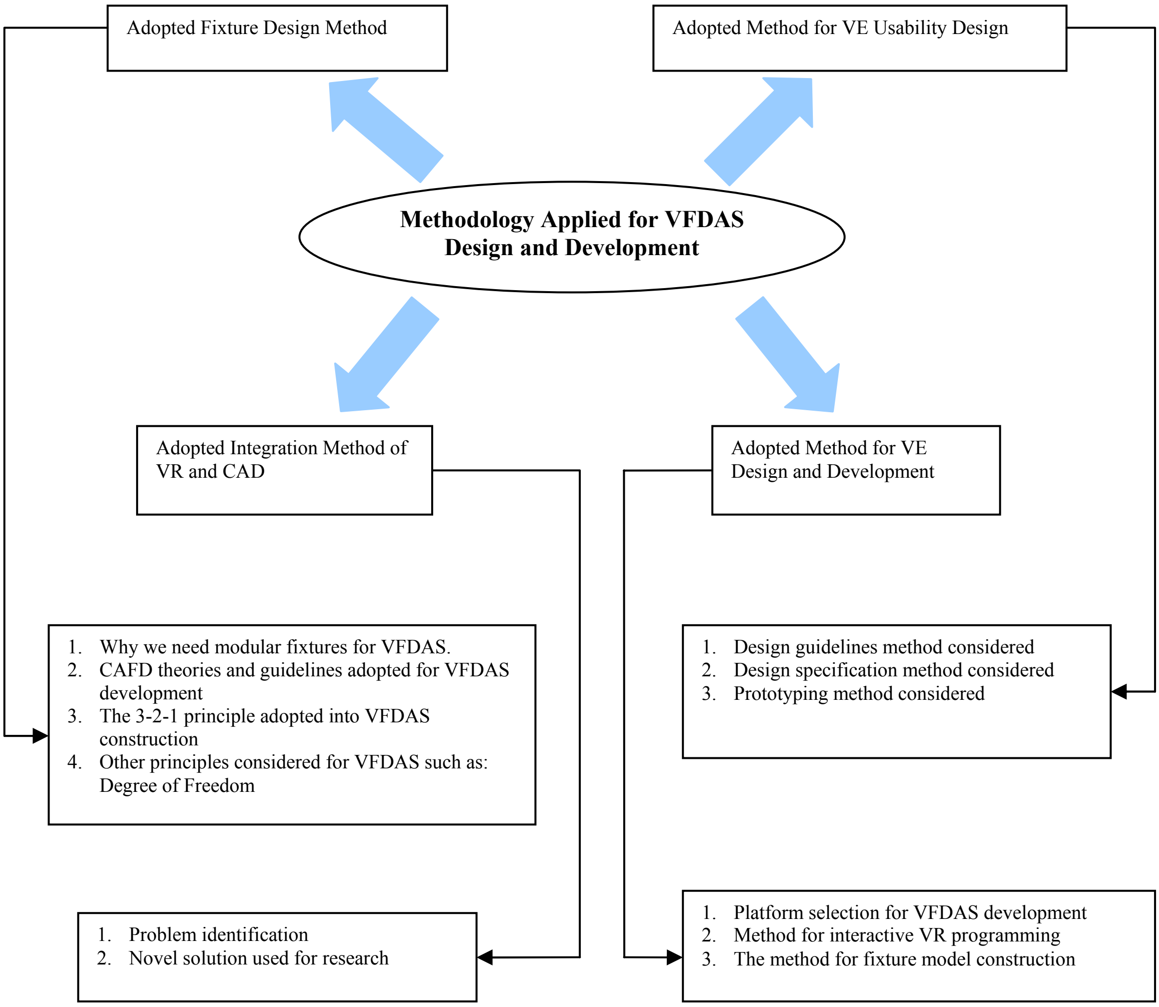

Adopted Fixture Design Method
Adopted Method for VE Usability Design
Methodology Applied for VFDAS Design and Development
Adopted Integration Method of VR and CAD
Adopted Method for VE Design and Development
1. Why we need modular fixtures for VFDAS.
2. CAFD theories and guidelines adopted for VFDAS development
3. The 3-2-1 principle adopted into VFDAS construction
4. Other principles considered for VFDAS such as: Degree of Freedom
1. Design guidelines method considered
2. Design specification method considered
3. Prototyping method considered
1. Problem identification
2. Novel solution used for research
1. Platform selection for VFDAS development
2. Method for interactive VR programming
3. The method for fixture model construction

Fig.3-1. The Methodology Structure

3.2 Fixture Design Methodology

3.2.1 The Advantages of Modular Fixtures

Fixtures are important devices used in machining, assembly, inspection and other manufacturing operations to locate and hold a workpiece firmly in the correct position and orientation so that the required manufacturing processes can be carried out according to design specifications [18]. Due to the current trend for flexibility and productivity in manufacturing systems, fixturing systems are required to be as flexible as possible. Thus, flexible fixturing has become the important aspect in flexible manufacturing systems (FMS) and computer integrated manufacturing system (CIMS). There are several types of flexible fixtures e.g. modular fixtures, phase-change material applications, programmable fixtures, and adjustable fixtures etc. Modular fixtures are the most popular flexible fixtures in industrial practice. A typical modular fixture built on the T-slot base plate is shown in **Fig.3-2.**

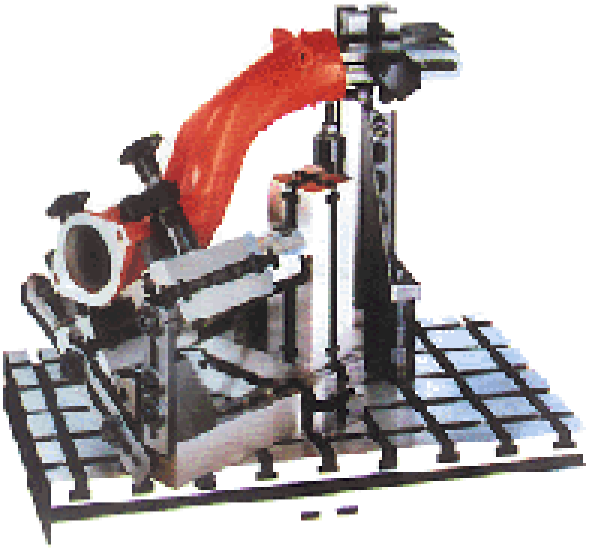


Fig.3-2. Erwin Halder Modular Jig & Fixture System with a workpiece

Modular fixtures aim to achieve flexibility via multi-purpose fixturing elements. The flexibility of modular fixtures is achieved by the great number of fixture configuration gained from multiple combinations of fixture elements. A complete modular system is made up of a number of standard fixture elements such as: baseplates, locators, clamps, and supporting elements. Standard elements are selected and applied to construct the appropriate fixture configuration to securely hold the workpiece in position.

Furthermore, modular elements can be re-used for other manufacturing operations once they are disassembled because they are produced with high tolerances to comply with multiple workpiece requirements. Accordingly, the need for fixture storage space can be greatly reduced comparing with dedicated fixtures and the labour expense and time for designing dedicated fixtures are also minimized [5].

Along with the increasing use of CAD/CAM (Computer Aided Design/Manufacture) systems, modular fixtures have more advantages to provide a workholding solution that is compatible with CAFD. Modular fixtures are the most applicable type of flexible fixturing systems for performing a CAFD process [11].

Due to these advantages of using modular fixtures presented above, the VFDAS framework focused on the research of modular fixtures. They were employed to develop a VR library module in VFDAS system that contains a number of standard fixture elements.

3.2.2 CAFD Systems' Theories Considered for the VFDAS Development

Modular fixture design is an intuitive process that is based on heuristic knowledge and experience of fixture designers. The computer aided fixture design (CAFD) technique is being rapidly developed to reduce lead time and cost, to optimize manufacturing operations, and to verify manufacturing process planning [59].

CAFD normally consists of three major aspects: set-up planning, fixture planning, and fixture configuration design. The objective of setup planning is to determine the number of setups required, the position and orientation of workpiece in each setup, and the workpiece surfaces to be machined. Fixture planning aims to determine the locating, supporting, and clamping surfaces and points on the workpiece. Additionally, fixture configuration design aims to select suitable fixture elements and place them into a final position in order to fulfill the functions of locating and clamping the workpiece [10]. As a result of modular fixtures being the most popular flexible fixture systems in industry, computer-aided modular fixture design (CAMFD) has become the research emphasis required to make manufacturing systems fully computerized and meet the increasing requirement on rapid fixtures [60]. The research in this thesis focuses on the fixture planning and fixture configuration design rather than multi-setups planning owing to the time limitation of PhD research.

The requirements of developing the VFDAS needed to be identified to direct the development of VFDAS framework as design guidance. At the preliminary stage of the VFDAS development, the requirements towards the VFDAS development were extracted from existing CAFD and CAMFD research. It was not necessary to explore and determine the design requirements of VFDAS as a separate research procedure because the majority of the requirements for the conventional fixture development have been already investigated and addressed in several CAFD research papers [5][7][11][13].

The VFDAS system was considered to be a particular type of CAFD or CAMFD system. The requirements used for the development of existing CAFD systems were regarded as useful for the development of VFDAS so that additional values generated by VR could be used to improve the current CAFD systems' applications and functionality. Requirements derived from the existing CAFD research [7], were therefore used to identify the research objectives for VFDAS development. More details related to these objectives of VFDAS can be found in previous section 1.4.

The final system architecture and functionality of VFDAS that have been established will be described in section 5.3 in the Chapter 5.

3.2.3 The 3-2-1 Principle for VFDAS

The 3-2-1 fixturing principle **(Fig.3-3)** is incorporated into the development of VFDAS based on a simple demonstration example by which a typical fixture design and assembly process is fulfilled in VFDAS. This example of fixture design demonstrated in VFDAS was applied to perform a machining operation in which there is a vertical hole produced on the top surface of the workpiece. As a result, the fixture design with regard to this example was conducted in compliance with the 3-2-1 design principle to determine both locating and clamping points. More details with regard to this example will be introduced in section 5.3.1.

The major concept of the 3-2-1 design principle aims to identify the three important reference datum planes by which the three relevant sets of locators are defined in appropriate locations and orientations. The first reference datum is to correspond with the first set of three vertical locators. The second reference datum is to correspond with the second set of two horizontal locators and the tertiary reference datum is also to correspond with the third set of one horizontal locator respectively [6]. Afterwards, the reasonable number of the clamp units can be employed to fulfill the clamping features upon the opposite facets of each reference datum. More details with regard to the guidelines of 3-2-1 principle can be found in previous section 2.2.2.

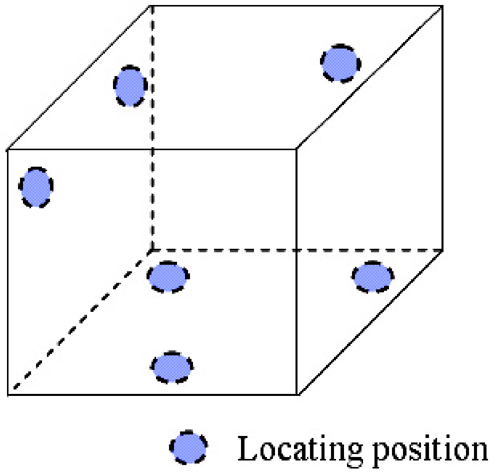


Fig.3-3. The 3-2-1 principle for locating determination

3.2.4 Other Fixturing Principles Considered within VFDAS

Importantly, the external cutting forces need to be counteracted by a fixture to keep the workpiece in static equilibrium while a machining operation is being carried out. The locators are applied to locate the workpiece in static equilibrium to remove all Degrees of Freedom (DOF). Clamps are used to hold the workpiece securely against the locators during machining process. Supporters are applied to improve the stability of workpiece.

A workpiece has six DOFs: translations along the X, Y and Z axes; rotations around three axes, and each DOF contains two moving directions **(Fig.3-4)**. It is considered that the workpiece is deterministically located as soon as one moving direction of every DOF is constrained. To immobilise the workipiece and counteract the machining forces, the locating and clamping rules from 3-2-1 principle were applied into the development of VFDAS for the fixture design [9].

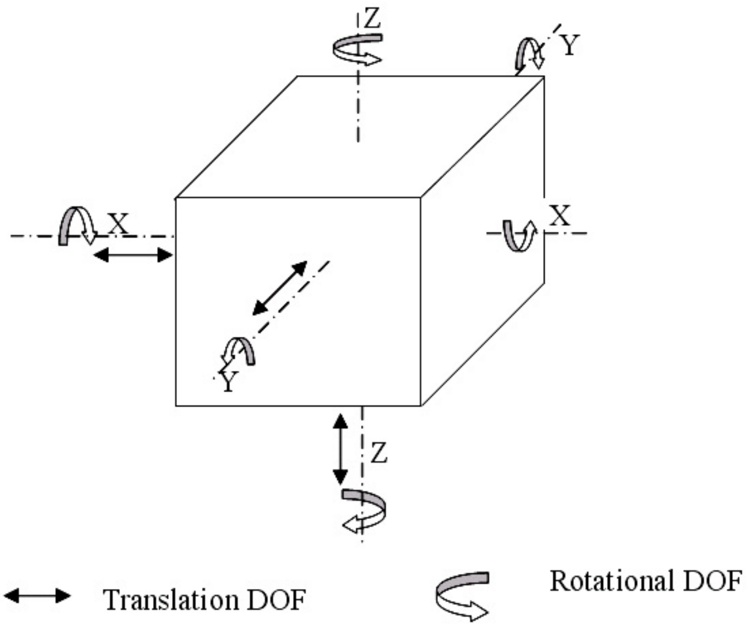


Fig.3-4. The Degrees of Freedom for a Workpiece

3.3 Integration Method of VR and CAD

3.3.1 Problem Identification

One of the difficulties of using VR for the VFDAS is the problem of integration of 3D models between CAD toolkits and VR-based systems. This bottleneck related to the integration study often results in information loss for solid CAD models throughout the integration process. Many original functionality and attributes become unavailable or incompatible after data transfers between VR and CAD systems. Multi-optional user preferences and different algorithms in VR and CAD systems are not standardised for data exchange. The commonly used exchange formats are *VRML, *DXF and *NMO etc and they serve as communicative media to bridge the gap between CAD system and VR system. However, data loss and incompatibility still remains a problem.

As majority of industrial VR designers have experienced, the massive nurbs and splines of 3D CAD models created in Pro/ENGINEER **(Fig.3-5)** not only result in rendering lag in the virtual environment but also produce unsolidified and untessellated 3D entities after they are transferred into the VR authoring software. These cause problems for manipulating the 3D virtual entities with good interactivity and visualization in a real-time VR environment. The reason that unsatisfied 3D CAD models are caused is due to the computational method used in Pro/ENGINEER especially referred to *VRML format models, which has nothing to do with the integration method used for 3D model data conversion. Fixture designers involved in this research used Pro/ENGINEER to fulfill their conceptual, structural and configurable design for fixture development.

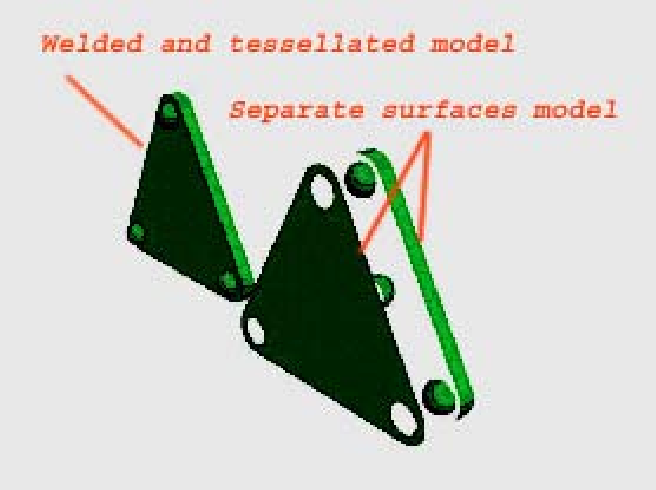


Fig.3-5. Unsolidified and Untessellated Models of a Fixture Element Derived from Pro/ENGINEER Wildfire

Owing to the research being focused on VR applications in fixture design and assembly, the conversion difficulty between CAD and VR systems is a problem that must be resolved. Thus, an integration solution as an essential method had to be defined and applied to the research in this thesis.

3.3.2 The Solution Used for This Research

To facilitate 3D model data exchange and integration process between CAD and VR, a few relevant literature and methods are explored and reviewed. Steffan et al addressed a new method with regard to integrating an interactive VR based assembly simulation of a digital mockup into an existing CAD/CAM infrastructure. The research also described the feasibility about how VR based assembly simulation improves the CAD/CAM procedures [42]. Furthermore, there is another method with regard to the integration of VR and CAD used in the virtual design and assembly system (VDAS) established by Ji and Choi et al [1]. The VDAS system is a VR-based engineering application which allows engineers to design, modify and assembly mechanical products and this research was carried out by the Hong Kong Polytechnic University [1].

According to the result of experimentation, the appropriate method for solidifying 3D Pro/ENGINEER model within an integration process was identified. Another important 3D solid modeling software, Rhino 3.0 was used for welding separate 3D entities together as a whole 3D object in order to enable further interactive VR development. These included 3D interactive VR simulation, assembly animation and adding essential physical properties e.g. object collision, friction, gravity and elasticity etc. They are regarded as the important design components for the VFDAS system development. This method was used instead of the function of "Attach" or "Collapse" inside the modelling software applications e.g. 3Ds Max because this often leads to additional integration problems within the interactive VR authoring environments e.g. "untidy" 3D models and insufficient solid 3D models. These models could be wrecked while developers attempt to carry out conversion process, especially for complex geometric shape of 3D models. The fundamental method employed for completing the integration process of CAD and VR are schematically described below **(Fig.3-6)**.

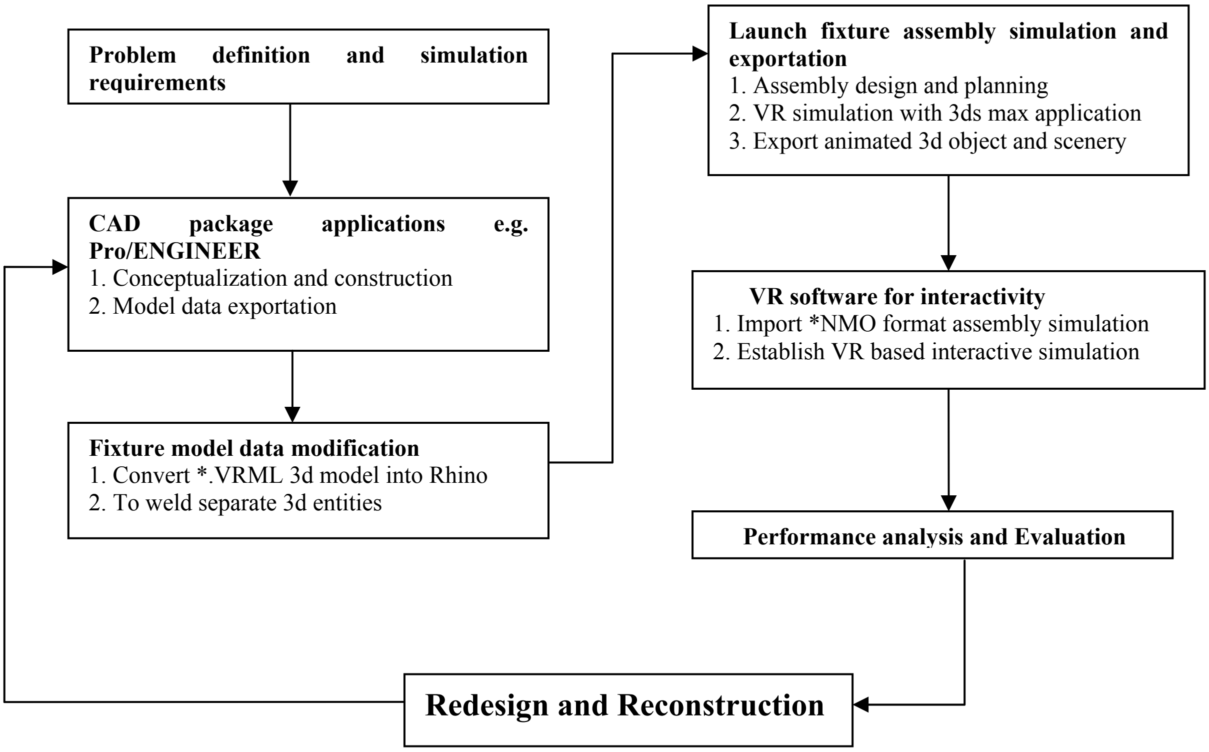
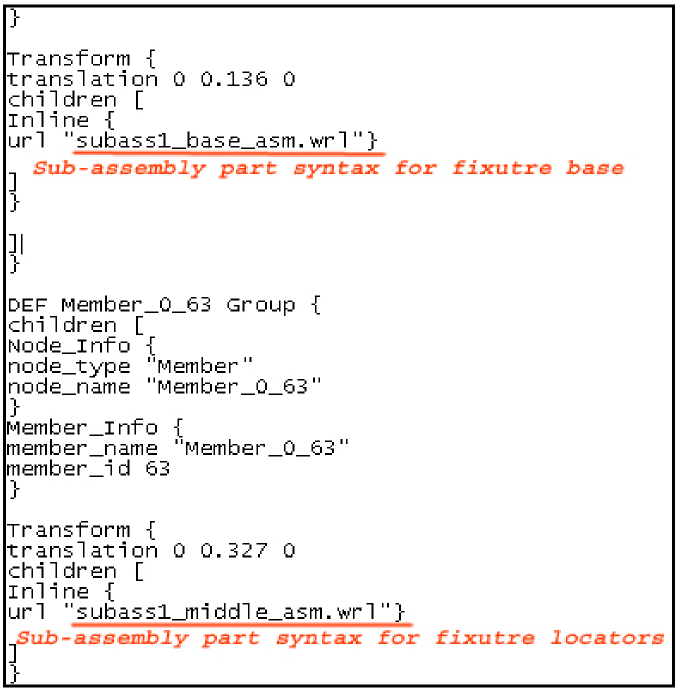

Problem definition and simulation requirements
CAD package applications e.g. Pro/ENGINEER
1. Conceptualization and construction
2. Model data exportation
Fixture model data modification
1. Convert *.VRML 3d model into Rhino
2. To weld separate 3d entities
Launch fixture assembly simulation and exportation
1. Assembly design and planning
2. VR simulation with 3ds max application
3. Export animated 3d object and scenery
VR software for interactivity
1. Import *NMO format assembly simulation
2. Establish VR based interactive simulation
Performance analysis and Evaluation
Redesign and Reconstruction

Fig.3-6. Integration Method Used for the Pilot Study Interactive VR Simulations and the VFDAS Construction

The following procedures describe the integration method developed through the pilot studies and used for the VFDAS system. (1) After discussion in relation to potential problems for the modular fixture design amongst the specialists from different domains, requirements with regard to interactive VR simulation were identified. (2) Thereafter, the conceptualization and construction for modular fixture assembly model including each assembling element may be performed by the fixture engineers using a CAD package, such as: Pro/ENGINEER. Once the conceptual design for a fixture at the early stage is preliminarily finalized, the CAD models for fixture design are exported with portable format, *.VRML. (3) The CAD models with *.VRML format are converted into an acceptable Rhino format e.g. *.3DS with using Deep Exploration suite and then imported into Rhino package for further development. Rhino is then used to weld separate 3D entities for each element model to generate a solidified 3D model other than the main assembly structure *.VRML file. This main structure file determines the spatial relationship and relative position for each assembly element according to the final assembly. The detailed syntax of main *.VRML assembly file with spatial relationship is described in **Fig.3-7**. (4) After obtaining the solidified CAD models, a fixture assembly animation design and production can now be carried out using 3ds Max toolkit in terms of the identified assembly requirements. Afterwards the VR simulation animation for fixture assembly comprising animated 3D objects and scenery can be exported with *.NMO format using Virtools Max exporter or Virtools CAD package. (5) Import *NMO format animation into Virtools Dev so as to establish interactive functionality, real-time visualization and design Human-Machine Interface (HMI) for the VR based interactive simulation. (6) According to the completed VR based simulation for fixture assembly, the relevant performance analysis and evaluation may be now conducted by both fixture design engineers and assembly engineers. (7) If necessary, the fixture design and assembly plan can be brought back to redesign and reconstruction afterwards.



```
}

Transform {
translation 0 0.136 0
children [
Inline {
url "subass1_base_asm.wrl"}
  Sub-assembly part syntax for fixutre base
]
}

]
}

DEF Member_0_63 Group {
children [
Node_Info {
node_type "Member"
node_name "Member_0_63"
}
Member_Info {
member_name "Member_0_63"
member_id 63
}

Transform {
translation 0 0.327 0
children [
Inline {
url "subass1_middle_asm.wrl"}
 Sub-assembly part syntax for fixutre locators
]
}
```

Fig.3-7. An Extract of Main Assembly Structure File

3.4 VE Design and Development

3.4.1 Platform Selection

To identify the most appropriate VE platform for this research, a number of potential VE platforms were reviewed by the author. These were Virtools Dev, Superscape VRT, WorldUp, Java 3D and other relevant VE platforms which will be respectively described in the following sections. The suitability of the VE platform for the proposed VFDAS system was considered carefully. In general, the system was based on how well it supports the 3D applications with interactivity and physics simulation.

3.4.1.1 Virtools Dev

Virtools Dev is made up of five essential components which are the Graphical User Interface, the Behaviour Engine, the Render Engine, the Virtools Scripting Language and the SDK [61]. They execute their own independent functionality within the Virtools developer platform in order to work jointly to fulfil interactive 3D applications and performance. The Graphical User Interface (GUI) is to establish interactive 3D applications through visually assembling behaviour objects so that developers may drag, drop and attribute modular behaviours to form 3D applications with acceptable graphics and interactivity based on this Virtools GUI **(Fig.3-8)**. The Behavior Engine is deemed as the key driving power based on which interactive 3D media content created in Virtools developer toolkit can run properly. Similarly, the Render Engine aims to render graphics, images and animation in real-time. The Virtools Scripting Language (VSL) is a scripting language that complements the Virtools Dev Schematic editor by offering script level access to the Virtools SDK (Software Development Kit). In addition, the VSL Editor provides a context-sensitive text highlighting system, context-sensitive completion and automated display of function arguments. Lastly, the Virtools SDK is a suite of development tools which allow developers access to all low level specific functionality driven by Virtools platform in order to create the custom Behaviours, Media Importer, Manager and Render Engines Plugins etc.

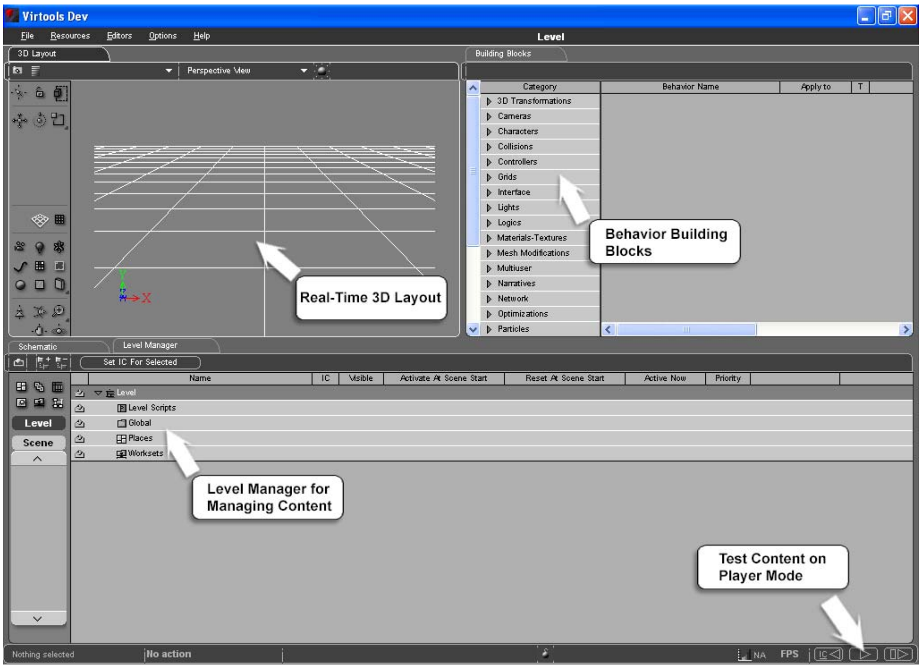


Fig.3-8. Virtools Graphical User Interface

Virtools Dev is an authoring VR toolkit that enables users to generate "compositions" (CMOs) that consist of full of rich, interactive, 3D assets. Virtools Dev is capable of activating industry standard media to life e.g. models, animations, sounds, images and so on [61]. On the other hand, a behaviour is simply defined as a description of how a certain element acts in an environment. Therefore, there is a standardised library of reusable behaviour building blocks (BBs) available within Virtools Dev in order to produce interactivity or insert users' customized components with using Virtools Dev SDK.

One advantage was that the realistic 3D virtual environment with many physical properties can be simulated by using Virtools Physics Pack embedded within Virtools Dev suite. VEs associated with physical properties are able to obey the fundamental Newton's laws of physics such as: mass, gravity, friction, elasticity, collision detection, physical constraints amongst objects and advanced physics features that comprise buoyancy, car behaviours and force fields [61]. In order to find a solution using VR technology to simplify the complex fixture design process, the development of VEs within VFDAS system requires that every virtual fixture element could be attributed with these physical properties which are normally taken into account throughout the actual fixture design and evaluation process. These physical properties of VR can be utilized to improve the modular fixture design and assembly process as a novel method. As a result, Virtools software has great capability for conducting the development of VFDAS system according to the requirements of physics simulation and VR interactivity creation where the fixture design and assembly process is fulfilled realistically within the VR environment as if it is carried out in the real physics world. Particularly, the accurate real-time collision detection affiliated to the physical property simulation is regarded as the significant functionality towards VFDAS development because this feature may greatly enhance the realism of the VEs and contribute the process of fixture design and assembly.

3.4.1.2 Superscape VRT

Superscape VRT (Virtual Reality Toolkit) was made in 1998 [62] and has a number of editors inbuilt into the programme to enable the creation of VEs. These editors are summarized as: Shape Editor, World Editor, Visualiser, Layout Editor, Resource Editor, Texture Editor, Sound Editor [63]. The Shape Editor is used to create shapes, the basic building blocks of objects placed in the VE. The World Editor operates as a central hub of VRT, where design components from the other editors are incorporated together to establish the potential VE. The Visualiser is recognized as a platform which allows the user to interact with rather than edit the VE. The Resource Editor has ability to create pop up dialogue boxes to provide information to users as well as allow users to input data or make selections from a group of menu choice. The Texture Editor can import graphic files such as 'jpg', 'gif', 'bmp' etc which are then modified and positioned on the facets of 3D objects. The Sound Editor makes 'wav' files imported and allows further edit to pitch, volume and playback speed of sounds applied in VE. VRT virtual worlds **(Fig.3-9)** are 3D environments, encompassing groups of objects which consist of groups of shapes at the root level. These shapes and objects are placed within a bounding cube, which normally defines its collision detection [63]. Consequently, the incapability of collision feature in Superscape induces an inaccurate collision detection between virtual objects. The basic reason for this is that the algorithm applied in Superscape only determines the rough collision detection between the cuboid envelopes **(Fig.3-10)** that surround 3D objects instead of the actual geometry collisions. It also misleads the ineffective collision volume between the cuboid envelope and the actual geometry of 3D objects. Interaction in the VE between the user and virtual objects is controlled by Superscape's Command Language (SCL) programming code and this can be one large piece of code stored on one object or it can be a series of pieces of code stored on separate objects which communicate with each other. More information can be found in the "VRT for windows User Guide" published in 1998 [62].

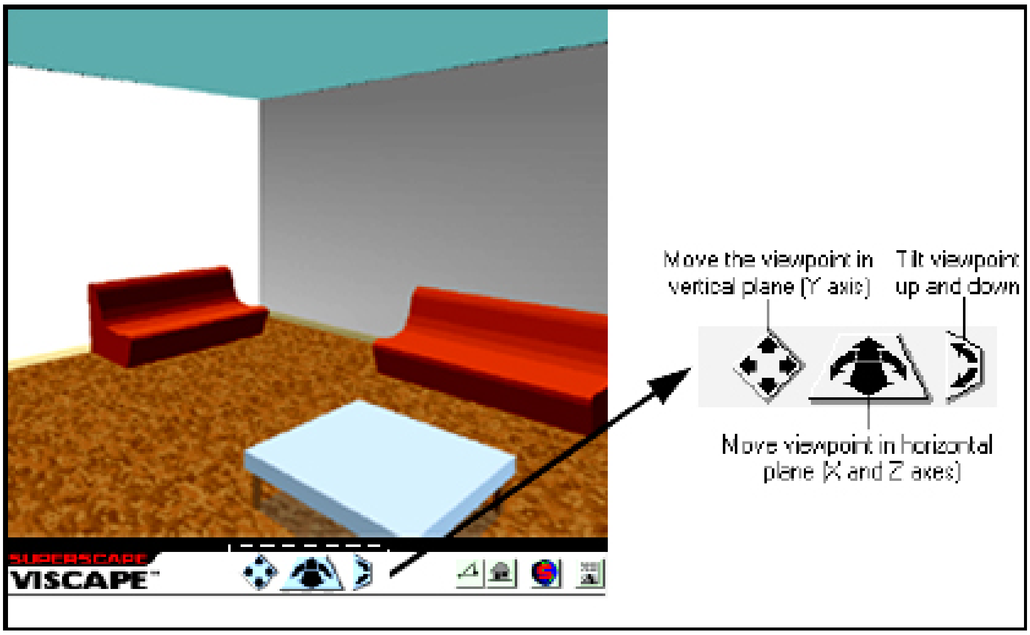

Move the viewpoint in vertical plane (Y axis)
Tilt viewpoint up and down
Move viewpoint in horizontal plane (X and Z axes)
SUPERSCAPE
VISCAPE

Fig.3-9. Controlling the View of a Superscape 3D Virtual World

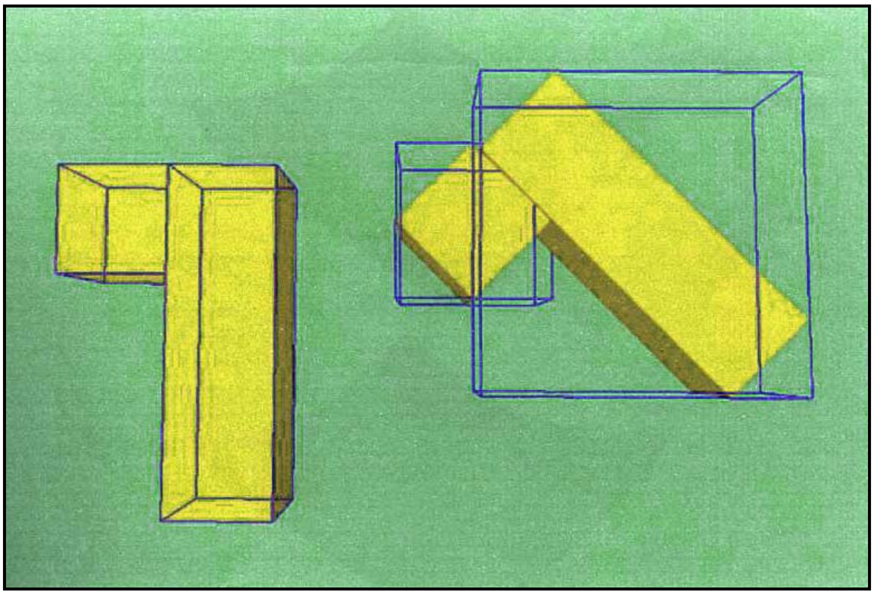
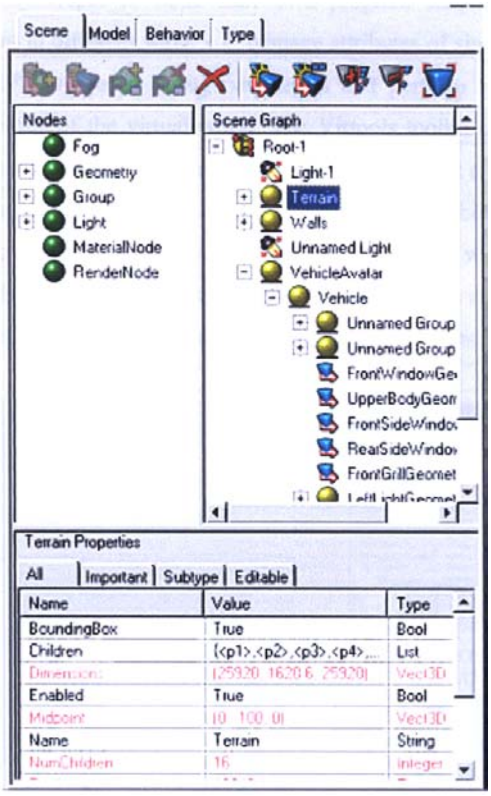


Fig.3-10. Bounding Cuboid Envelopes that Induce Inaccurate Collision (Kerr, 2005)

3.4.1.3 Worldup

The Worldup VE development tool offers an object-oriented environment which is designed to accelerate VR application development [63]. Predetermined methods and properties are contained in an extendable object hierarchy, which can be accessed through the developer interface or the Visual Basic style scripting language. The WorldUp development environment **(Fig.3-11)** provides a script debugger and a performance profiler to optimize performance. It also provides a series of tools which include a combined modeler, content delivery device and internet support etc. Unlike Superscape VRT, Worldup object's properties and interaction between them are shown in a list of 'graphs', *scene*, *behaviour*, *type* and *models* which are contained in a 'workview' window. It is adjacent to the development window that displays the environment being created. The workview graphs are a hierarchical list showing the sequence in which actions are taken, such as: the sequence of events when objects are interacted with or the sequence of rendering or activating objects [63].



Name	Value	Type
BoundingBox	True	Bool
Children	{<p1>,<p2>,<p3>,<p4>,...	List
Dimension	[25920 1620.6 25920]	Vect3D
Enabled	True	Bool
Midpoint	[0 100 0]	Vect3D
Name	Terrain	String
NumChildren	16	integer

Fig.3-11. Hierarchical Structure of Worldup Objects (Kerr, 2005)

3.4.1.4 Java 3D

Java 3D is an extension to the Java programming language that forms a connection between the JRE (Java Runtime Environment) and a computer 3D graphics support. The main difference of Java 3D from other VE platforms is that Java 3D application is compiled programs and Java 3D worlds are hard coded with the viewer. This means that Java 3D worlds can not be exported to file, edited and re-launched. The development process involves working with 3D models and other files which comprise the 3D virtual world in their original file format. The source files that compose a Java 3D world, e.g. the 3D objects, textures, sounds and interactions are compiled together with the viewer into an application that collaboratively runs on the operation system. There are several limitations existing in Java 3D. Java 3D applications run more slowly than some other VR viewers and Java 3D plug-in is not available for Apple Macs, therefore will not run on all operation systems.

3.4.1.5 Other VE Platforms

Other VE platforms that currently exist are not development toolkits but programme engines that usually contain the capability of producing interactive 3D environments.

These platforms are discussed by Reynard and Eastgate in 2001 [64]. These VE developer platforms demand more programming application and development which mean they possess the advantage of greater control in programming what is required. However, with the downside of taking longer to program and being less robust, which indicates more likely to fail [64]. Moreover, the importance for developing an applicable VFDAS system is to no longer manage the fundamental details of low-level programming, but emphasize on the content of VE construction.

3.4.1.6 Choice of VE Platform for VFDAS

Having reviewed the available VE platforms, the reason why some potential VE developer toolkits described above can not be employed to perform the research, was identified such as: Superscape VRT, WorldUp, Java 3D etc. The reason was that they have inability or shortage to generate sufficient interactivity applications, provide advanced physics simulation and establish real-time 3D collision detection in accuracy during the VFDAS's VEs being constructed. This would hinder the development progress of VFDAS system.

Specifically, the inability of collision feature in Superscape normally induces the inaccurate collision detection between virtual objects and can not comply with the requirements of VFDAS system development. The development of VFDAS has clarified the need of accurate real-time collision detection between fixture elements to fulfil the process of fixture design and assembly.

In addition, Java 3D and WorldUp can not provide the same interactivity and physical features as other VR development software, which will significantly impede the research outcomes and the development progress of VFDAS.

In contrast, Virtools Dev has many advantages for this research. As a result, the final VE authoring platform choice identified to carry out the development of VFDAS was Virtools Dev software, version 3.0. Essential versatility and functionality of Virtools Dev are considered the most appropriate according to the requirements of VFDAS construction. Importantly, the function of reversely converting the resulting fixture assembly model generated by VFDAS system back to the universal CAD toolkits can be developed in VFDAS using Virtools CAD Pack in order to achieve the best integration between VFDAS and CAD systems.

3.4.2 Interactive VR Programming Based on Virtools Dev 3.0

Virtools Dev programming is a real-time 3D programming method for complex VR interactivity, in which the programmers are able to logically exploit the behaviour building blocks library to perform interactive behaviours and even physics world simulations. The programmers are also allowed to compose custom building blocks based on C++ language using SDK module. Custom building blocks can be used to achieve some interactive 3D applications in particular.

The approach used in developing the VFDAS system was based on the appropriate integration of the Virtools Dev and Pro/ENGINEER toolkit. The CAD models produced for fixture design are available from the Pro/ENGINEER. The 3D fixture assembly model could then be transferred into Virtools Dev after using the conversion approach described in section 3.3.2 in order to further establish the interactive 3D simulation and physics simulation during the development of VFDAS.

3.4.2.1 The Graphical User Interface (GUI)

Virtools Graphical User Interface allows VE developers to carry out the behaviour programming process to create applications with good graphics and interactivity. The Graphical User Interface is used as a main workplace throughout multiple stages of behaviour programming and VE development. It consists of several components and features which are presented as follow [61].

- **Graphical Tools** for creating, editing, selecting and navigating 3D objects, cameras, lights, curves, materials, textures, grids, paths and so on
- **3D Layout** window which is used to visualize interactive content in a real-time environment
- Translation, rotation, navigation and scaling of 3D entities within a virtual world
- **Schematic View** to assemble and attribute behaviour building blocks for the production of interactive applications
- **Script Debugger** to fine-tune the content

- **Entity Setup Tools** to edit the parameters of any object that has behaviours within a virtual environment
- **Attribute Manager** used to facilitate edit of attribute values for objects
- **Hierarchy Manager** to show a tree view of objects contained in different levels.
- To attribute behaviours onto 2D and 3D entities
- To produce new reusable behaviours by combining existing ones
- **Parameters Debugger** to validate and edit data values
- **Path Manager** to identify paths to data resources which include image, audio and video etc.
- **Action Manager** to produce scripts for often used functions which execute a predetermined task on a parameter or a selection
- **Profiler** used to monitor how much computing time is devoted into specific operations.
- **Shader Editor** to generate programmable vertex and pixel shader in order to advance quality of visualization in a VE

3.4.2.2 Class Hierarchy and Design Elements

Design elements in the Virtools environment are classified into different classes and comprise entities created in Virtools such as: Curves, Scenes and external media that are brought into Virtools platform such as: models, animation and sounds. Each design element in a VE is an instance of a class where every class is illustrated by a class definition. The term of 'CKClass' is a representative for any class or class definition used by Virtools virtual world, e.g. CKCharacter and CKLight [61]. For example, there is a light class in a VE to which all types of lights belong. Each light is an instance of the light class and has specific features which include its lighting colour and its radiation range etc. The Virtools Dev class hierarchy that clarifies the correlation of the different classes in a VE is demonstrated in the following graph **(Fig.3-12)**. This shows that VE developers are required to grasp the whole correlation picture and have an understanding about various types of CKClass in order to facilitate the behaviour programming process towards the development of the intended VE.

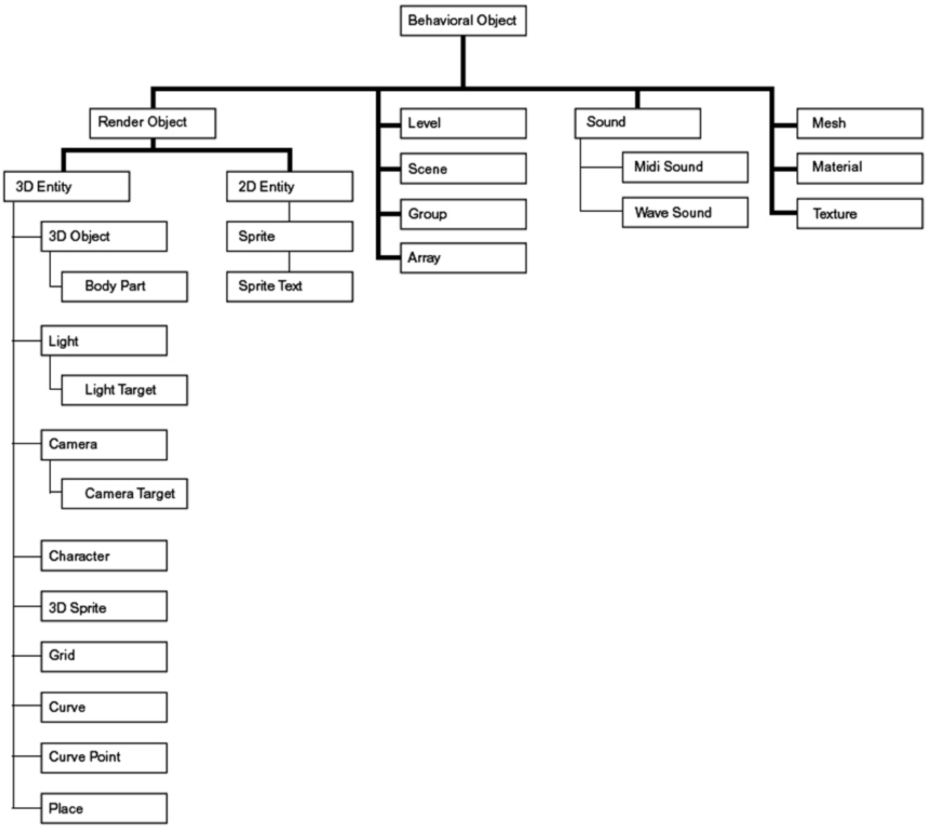


Fig.3-12. Virtools Dev Class Hierarchy

Design elements are controlled by executing methods encapsulated within behaviours and parameter operations. Each unique CKClass can only employ its type of behaviours which are specifically designed for the elements belonged to this CKClass. For example, there is a CKClass named CKCamera. Only CKCameras can use the behaviours particularly designed for cameras e.g. the Camera Orbit etc. Each design element has their own attributes that solely apply to objects of its CKClass. These attributes are adjusted through its CKClass Setup or a behaviour programming. For instance, a light comprises its own attributes that only apply to objects of the CKLight class. These attributes refer to the type of light such as: Point, directional, etc; the colour of the light such as: white, blue, etc; the range over which the light radiates and so on. These attributes are defined through the light Setup or a relevant behaviour programming. According to class types, a number of definitions for design elements in a VE are shown below and more details can be found in Virtools Dev User Guide in 2004 [61].

BeObject ─ an object to which a behaviour is applied

RenderObject ─ an object that is rendered, which can be seen in Play Mode

2D Entity ─ an object that has width and height but no depth

Sprite Text ─ a 2D Entity used to place text in the render window

3D Entity ─ an object that has width and height and depth

Character ─ a type of 3D entity that acts as a an intelligent entity, directed by the user or by software

Camera ─ an object that defines a point of view

Curve ─ a set of 3D entities that define a curve in 3D space

Grid ─ a 2D data set whose value depends on 3D coordinates

Light ─ an object that provides illumination

Array ─ a data set expressed as a table

Group ─ an collection of elements with no restriction as to type

Level ─ the parent object of the entire composition

Material ─ the surface characteristics of a mesh

Mesh ─ the set of vertices that define the shape of an object

Place ─ a collection of geographically related objects

Scene ─ a collection of temporally related objects

Sound ─ a sound stored in digital form

Texture ─ an image used to provide fine detail on the surface of an object

3.4.2.3 General Introduction for Behavior Programming

A behaviour is defined as a description of how a design element responds to certain form of input stimulus. Assigning a behaviour to a design element makes the element to be interactive either with the user or other elements in a virtual environment. As a result of behaviours being the central concept in Virtools environment, design elements can be considered that all of them descend from the Behavioural Object (BeObject).

In a Virtools environment, a behaviour can be expressed as a script which is regarded as the visual representation of a behaviour and applied to an design element in Schematic View. A script is normally text. However, the script described in Virtools is actually a schematic diagram of behaviours. There are two parts that compose a script and these two parts are a script header and a script body. The script header demonstrates the owner of the script and the name of the script. The script body commonly comprises one or more behaviour Building Blocks (BBs), Behaviour Graphs (BG), Parameters, bLinks, pLinks and so on. In a nutshell, a complete process of behaviour programming can be considered as VE developers assembly and attribute the reusable behaviours applied into objects. These behaviours can be obtained from a standard library that contains a bundled set of Behaviour Building Blocks and extended with custom behaviours developed by SDK. In addition, overall behaviour programming process normally comprises several components and features, which will be explained in the following paragraphs.

Behaviour Building Blocks (BBs) are the primary mechanism used to encapsulate a specific task and fulfil behaviours. There are various types of BBs shown below **(Fig.3-13)**. A typical Building Block can be divided into a few constituent elements **(Fig.3-14)** e.g. Behaviour Input (bIn), Behaviour Output (bOut), Behaviour Link (bLink), Parameter Input (pIn), Parameter Output (pOut), Target Parameter etc.

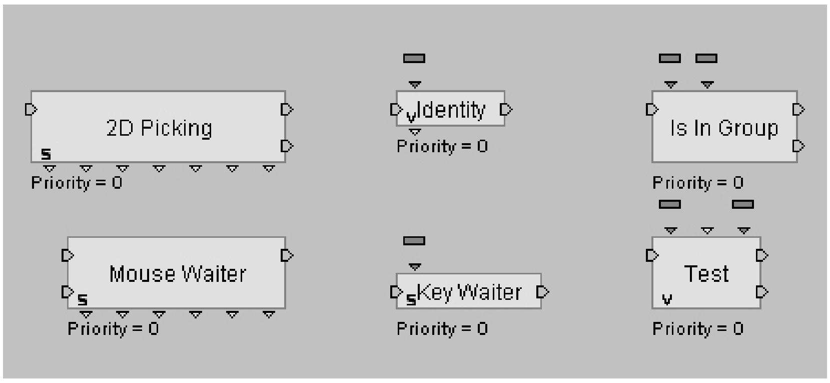


Fig.3-13. Several Behaviour Building Blocks

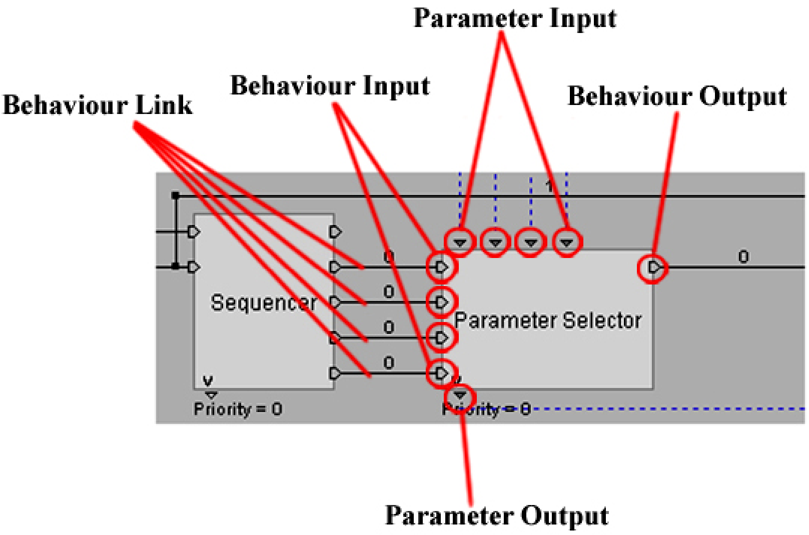


Fig.3-14. The Primary Constituent Elements for a BB

Behaviour Input is located on the left side of a BB and a BB has one Behaviour Input at least. Some part of the algorithm within a BB can be implemented while a bIn is activated or deactivated. Behaviour Output is located on the right side of a BB and a BB has a minimum of one Behaviour Output. A bOut can be activated as soon as the processing carried out by a BB in current frame (process loop) is finalized. The Behaviour Links are used to make connections between BBs, by which the sequence that BBs are consecutively processed is determined. Behaviour Links propagate activation flow from one BB to next one in order to allow sequential implementation of behaviours. A bLink is also able to connect the first BB to the script as well as connect bOuts to bIns. In addition, a bLink is associated with delay where the link delay can be defined during behavioural frames.

A Parameter Input (pIn) is regarded as only a connecting point used to receive a data value from another behaviour or from a parameter operation. Parameter Inputs are the arguments to the function contained by BB and make their name and type modified if applicable. A Parameter Output (pOut) is a data value generated by a parameter operation or a BB. Parameter Outputs are the values returned by the function encapsulated within a BB and have a destination. Parameter Outputs also can have their name and value modified if necessary. A Target Parameter is a particular type of pIn applied to define the element affected by the BB.

Messages are as a means of transmitting information between design elements or between behaviour scripts, which is used to signal a change in state, to activate allocated tasks being carried out as well as to signal that proposed operations are completed. Besides, a relevant BB comprises a message icon **(Fig.3-15)** which indicates that this BB is able to send or receive messages in virtual environment. Propagation of Messages always results in a delay of one pass via the process loop, which means there is a one frame delay between the duration from sending to receiving a message.

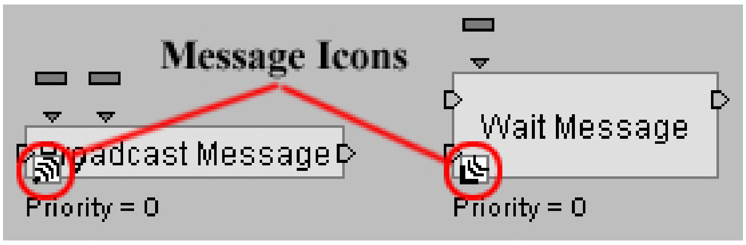


Fig.3-15. Two BBs for Sending and Receiving Messages

A **Behaviour Graph (BG)** is identified as an author-defined behaviour which consists of one or a set of BBs, Parameter Operations, Parameters, Parameter Links, Behaviour Links, Shortcuts, Comments and possibly associated Behaviour Graphs etc. A Behaviour Graph **(Fig.3-16)** has great similarity with a behaviour script because VE programmers encapsulate the behaviour into the BG, in which standardised interactivity and application are saved in a reusable form. It improves VE programming productivity in future work. A Behaviour Graph also contains pIns, pOuts, bIns, bOuts and so on. It comprises a part of a behaviour script or a complete behaviour script as long as the intended function is fulfilled. Moreover, a BG can be expanded to display the programming elements as well as collapsed to hide the programming elements to facilitate the behaviour programming process.

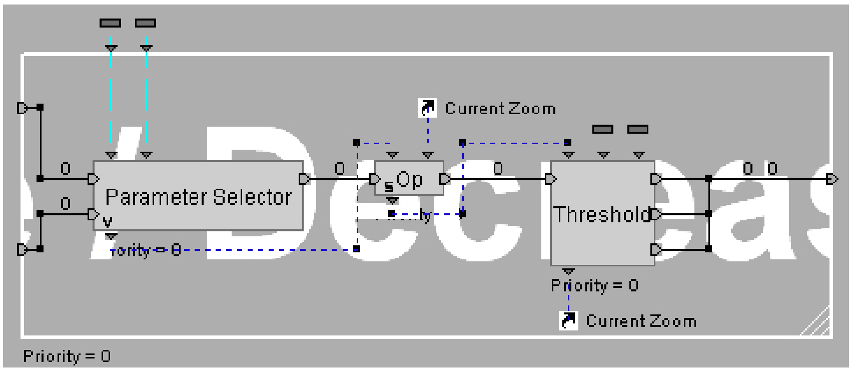


Fig.3-16. An Example of a Behaviour Graph

Parameters are used to transfer values between behaviours and assign values to the Attributes of Behavioural Objects (BeObject). Parameters are similar to variables in traditional programming. Many behaviours use the values of Parameter Inputs to control their processing. A Parameter **(Fig.3-17)** has a name and a value expressed in a given type such as: 3D entity, sting, integer, vector, etc. It is either static which means that the value is fixed or dynamic which means that the value varies depended on other factors.

The value of Parameter is often assigned to a Behaviour or a Parameter Operation by using a Parameter Link (pLink) or a Shortcut. A Parameter Link is a connection that passes a parameter value from a local parameter or a pOut to a pIn [61]. Unlike Behaviour Links, there is no link delay for Parameter Links. Besides, a Parameter Shortcut is used through a behaviour programming process and often comprises a destination and a source. It is considered as an instance of a Parameter.

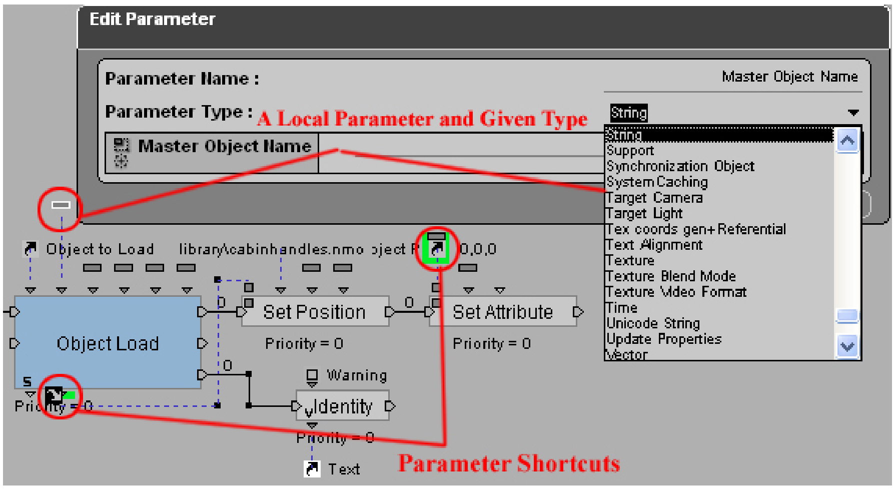


Fig.3-17. A Local Parameter and Parameter Shortcut Pairs

Parameter Operations (paramOp) are determined as simple operations conducted on a single Parameter or between a pair of Parameters. A paramOp is only implemented while the result is required by a BB or by another paramOp. The paramOps **(Fig.3-18)** are classified into three types in terms of different application requirements. They are ‘data retrieval’, ‘mathematical operation’ and ‘type conversion’ [61]. The paramOps that belong to the type of data retrieval can aggregate information on an object, e.g. CharacterBodyPart "GetBodyPart" Character String, Float "GetRadius" 3D Object etc. The paramOps that belong to the type of mathematical operation are applied to execute binary equations e.g. C=A*B, C=sin(A), V3 = Angle(V1, V2), V2=f*V1 etc. However, the paramOps that belong to the type of type conversion can convert one type of parameter to another [61].

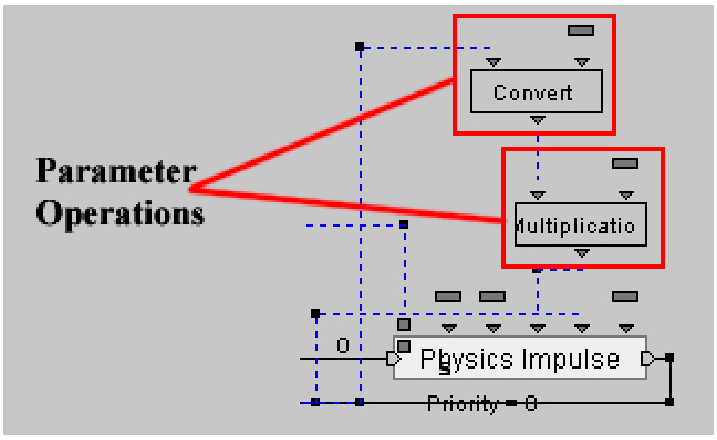
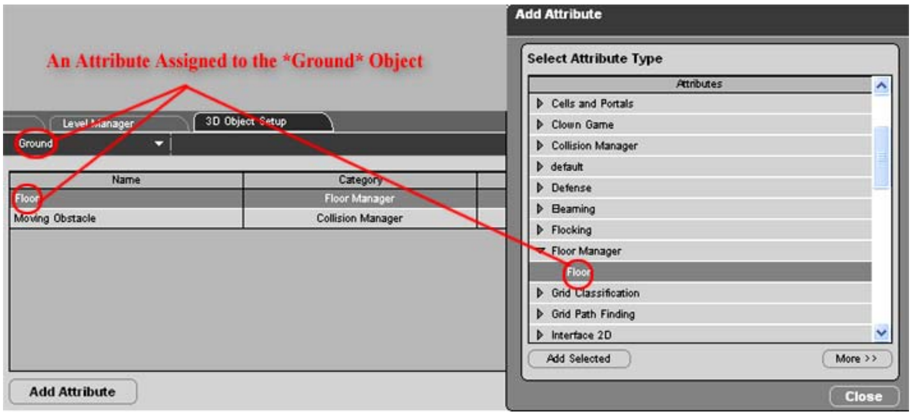


Fig.3-18. An Example of Parameter Operations

Attributes are recognized as an approach of interpolating information and values to design elements, which are similar to Parameters because they can store and apply information. The main difference between Attributes and Parameters are that Attributes are attached with a design element rather than belonging to a behaviour script. There are many predefined Attributes in the Virtools environment and VE designers are allowed to produce their own Attributes where appropriate. Attributes are only applied to a Behavioural Object (BeObject) and each Attribute has a name and a type **(Fig.3-19)**. An Attribute includes one or a set of parameters that may be edited. Attributes are assigned to design elements by using an element's Setup, Level Manager context menu or Attributes Manager. In addition, the BBs situated in Logics/Attribute e.g. the BBs of 'Set Attribute', 'Remove Attribute' and 'Has Attribute' are used to add, remove or retrieve an Attribute performed on design elements.



Name	Category
Floor	Floor Manager
Moving Obstacle	Collision Manager

Fig.3-19. An Example of Defining Attributes for an Object

3.4.2.4 Experimentation of Behaviour Programming

To demonstrate the approach for developing VEs and the process of behaviour programming, an experimental example is presented below **(Fig.3-20)**. This example is to construct an interactive VE in which a hydraulic fixture part can be viewed from different viewpoints and distances using keys on computer keyboard.

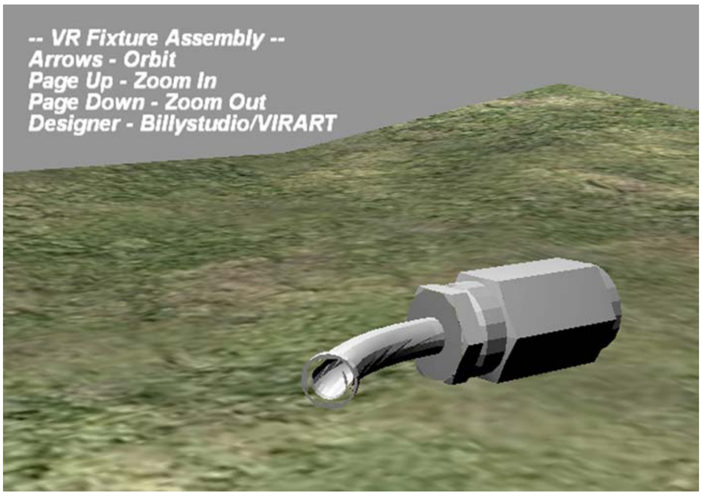


Fig.3-20. A Screenshot of Interactive VE for Hydraulic Fixture

Two Building Blocks (BBs) are involved into this behaviour programming process. One building block is called 'Camera Orbit' using which an assigned camera is made to orbit around a 3D entity. According to introductory graph below **(Fig.3-21)**, the Behaviour Input in this BB indicates that the BB process will be triggered as soon as an activation flow is received. The Behaviour Output means it is activated while the BB process is finalized. The Parameter named 'Target' denotes the 3D entity required to be targeted by the assigned camera. The Parameter Output named 'Distance' indicates the current distance between the camera and its target. In addition, the keys associated with the BB are used by default to move the camera orbiting around its target, such as: the default keys of *Page Up*, *Page Down*, *Up and Down Arrows*, *Left and Right Arrows*, *Right Shift*, etc. The *Page UP* key is used to zoom in. The *Page Down* key is used to zoom out. *Up and Down Arrows* are used to rotate around the view X axis. *Left and Right Arrows* are used to rotate around the view Y axis. The *Right Shift* key is used to accelerate motion by 2. However, the default keys can be replaced via the Edit Settings of the BB in order to allow VE programmers to customize controls **(Fig.3-22)**.

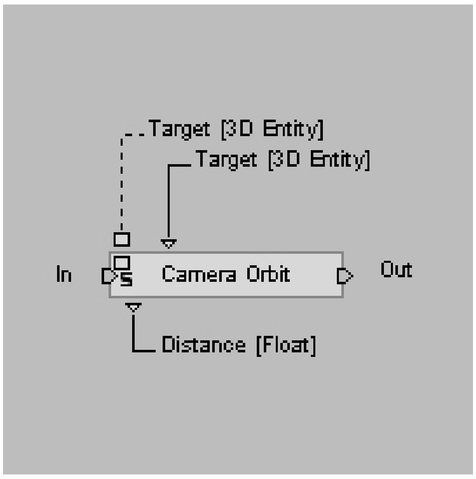
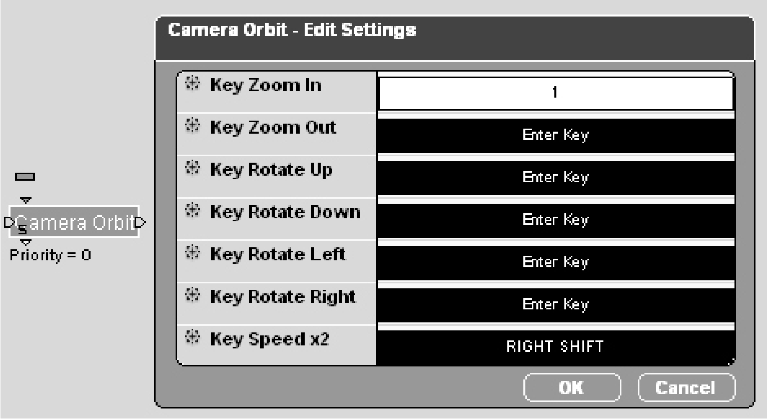


Fig.3-21. The Introductory Graph of the Camera Orbit BB



Camera Orbit - Edit Settings

Key Zoom In	1
Key Zoom Out	Enter Key
Key Rotate Up	Enter Key
Key Rotate Down	Enter Key
Key Rotate Left	Enter Key
Key Rotate Right	Enter Key
Key Speed x2	RIGHT SHIFT

Fig.3-22. The Edit Settings for the Camera Orbit BB

According to the introductory graph **(Fig.3-23)**, another BB used for this interactive VE is called ‘Set As Active Camera’ which implies that it contains function to change the active camera in the scene so as to realize switching between different cameras. The Parameter named ‘Target’ on the BB is applied to identify the specific camera which needs to be set as an active camera in the scene.

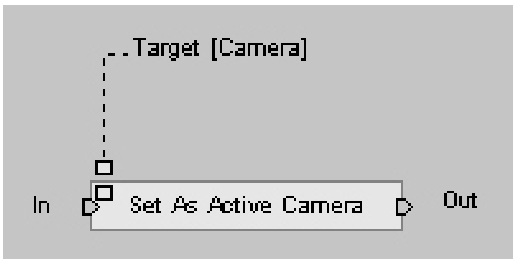
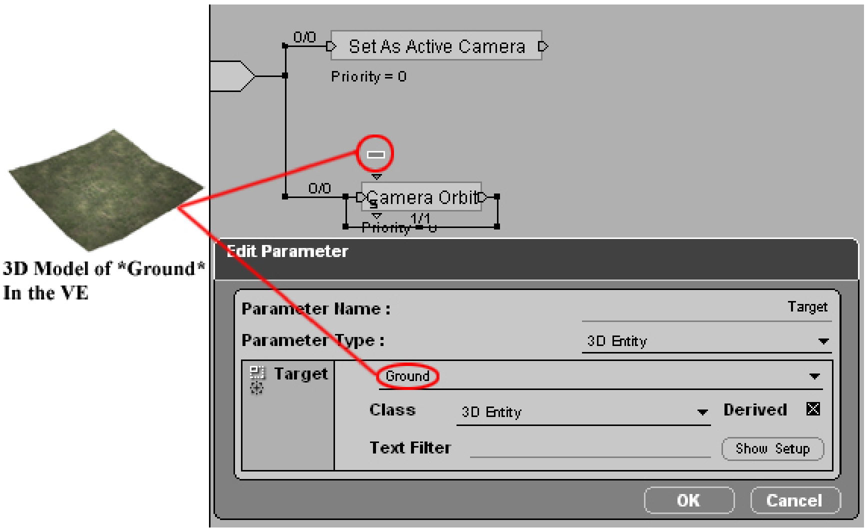


Fig.3-23. The Introductory Graph of the Set As Active Camera BB

These two BBs are assigned to the target camera named ‘My Camera’ in Schematic View and this means that these two BBs will take an effect on the ‘My Camera’ camera as a target. They operate jointly to perform the proposed interactive function. The BB of ‘Set As Active Camera’ is used to switch ‘My Camera’ as the active camera once VE plays at the run-time. The BB of ‘Camera Orbit’ is used to fulfil orbiting ‘My Camera’ around the specified 3D entity named ‘Ground’ **(Fig.3-24)** using keys on computer keyboard. Finally, a Behaviour Link between the bIn and bOut of Camera Orbit BB needs to be created to constantly carry out this BB function. The detailed behaviour programming is shown below **(Fig.3-25)**.



Parameter Name : Target
Parameter Type : 3D Entity
Target: Ground
Class: 3D Entity
Derived ☒
Text Filter
Show Setup

Fig.3-24. The Target Parameter Assigned for Camera Orbit BB

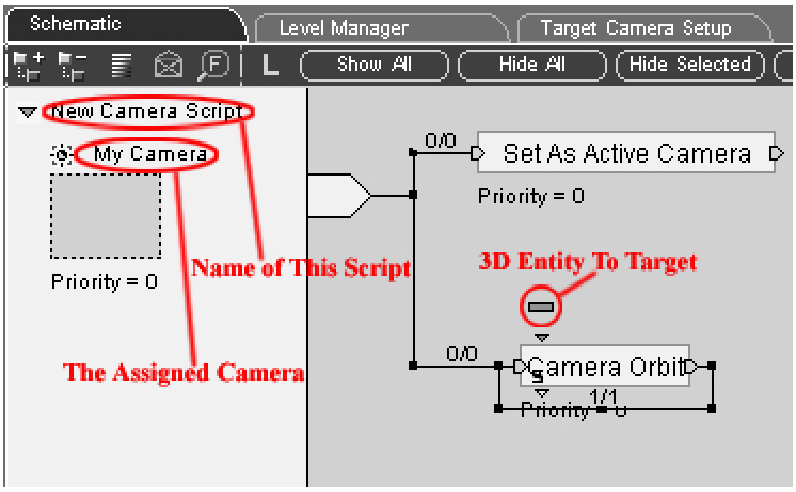


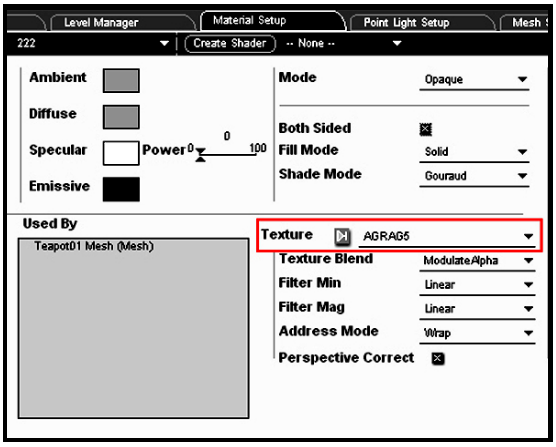
Fig.3-25. The Behaviour Programming for the VE

3.4.2.5 3D Models and Objects

Virtools Dev is not a modelling application and so the models can not be constructed in Virtools. They must be developed using external modelling software and then are imported into the VE developer toolkit where interactions are programmed in a hierarchal and schematic structure. Simple objects e.g. Cameras, Lights, Curves, Interface elements and 3D frames can be produced in Virtools platform. 3D frames are named dummies or helpers in 3D modelling software. The modelling software used for the model creation usually are Pro/ENGINEER, 3D Studio Max, or Lightwave in conjunction with Virtools platform for further interactive VE development.

3.4.2.6 Material, Texture and Mesh

The material used to define the surface characteristics of a mesh is simply created by using the button on the toolbar in Virtools. It is attributed by Material Setup **(Fig.3-26)**. The texture is created as an image used by a material to give an element an appearance. To ensure VE fidelity, photo realistic images can be imported into a texture through Texture Setup and are placed to an object's facets. A mesh is referred as a collection of facets that define the geometric surface of an element and it is often wrapped by a material. In a Virtools world, a texture is part of a material, a material is part of a mesh and a mesh is part of a 3D object. They are recognized as the basic elements of VE and work together to improve the quality of VE visualization and construct the fundamental layout of a VE excluding interactivity and lighting.



222 | -- None --

Power 0 — 100: 0

Mode: Opaque

Both Sided: ☒

Fill Mode: Solid

Shade Mode: Gouraud

Teapot01 Mesh (Mesh)

Texture: AGRAG5

Texture Blend: ModulateAlpha

Filter Min: Linear

Filter Mag: Linear

Address Mode: Wrap

Perspective Correct: ☒

Fig.3-26. Material Setup Panel with the associated Texture

3.4.2.7 Light Mapping and Material Baking

Due to the conversion issue between CAD and VR systems, the attributes related to 3D object texturing will be lost or reduced in graphics quality during the integration process. To perform interactive VR simulations for fixture assembly where the good visualization and real-time computer graphics of a VE are required, the method of light mapping and Material Baking is identified to carry out the research and achieve high-fidelity VR simulations.

The light mapping and material baking method is based on the function provided by 3D Studio Max named 'Render to Texture' and considered as the main approach for VE texturing and mapping in the research. This method is able to compensate the mapping shortage of CAD system and provide better texturing for VEs design.

Using this method, an object's material and lighting effect in the final rendering are baked and combined together into new texture maps. These texture maps are converted to Virtools platform associated with 3D models so as to fulfil the further behaviour programming in a VE. The baked texture maps of 3D models with good visualization can be fully accepted by Virtools as texture elements so that the high fidelity and real-time rendering in a VE are achieved. A good example of texturing result for a teapot model is presented below **(Fig.3-27)**.

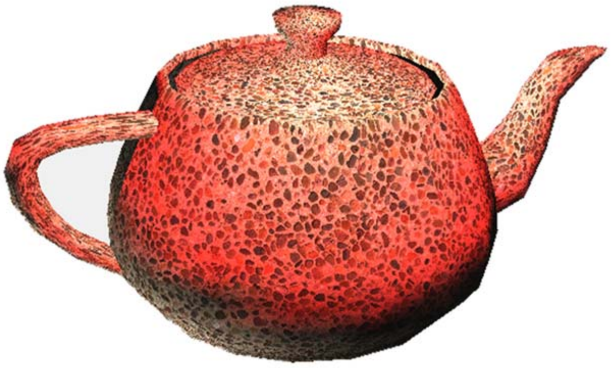


Fig.3-27. Texturing Result for a Teapot Using Light Mapping Method

3.4.2.8 Animation and Simulation

To achieve the objective of interactive VR simulation for fixture assembly, the prescribed animation has to be integrated into the VR-based simulation so that the corresponding needs for a simulation are met. Although Virtools has no capability to generate the prescribed animation, the solution is to employ other modelling applications, e.g. 3D Studio Max, Maya and Lightwave so as to produce the prescribed animation after simulation requirements are identified by a VE developer.

The prescribed animation is composed of keyframes, where each keyframe has a position, scale and orientation. After creating the prescribed animation in which pre-scheduled assembly sequence and trajectory for fixture assembly are demonstrated, 3D object models that contain the animation can then be transferred into the Virtools platform in order to interpolate interaction features on the animated 3D object and complete the interactive VR simulation. A typical example is the pilot case study in which the process of setting up the turbine blade on the fixture was demonstrated in a simulation. It was developed by the prescribed animation approach using 3D Studio Max. The picture to illustrate the production process of the animation with keyframe settings is shown below **(Fig.3-28)**.

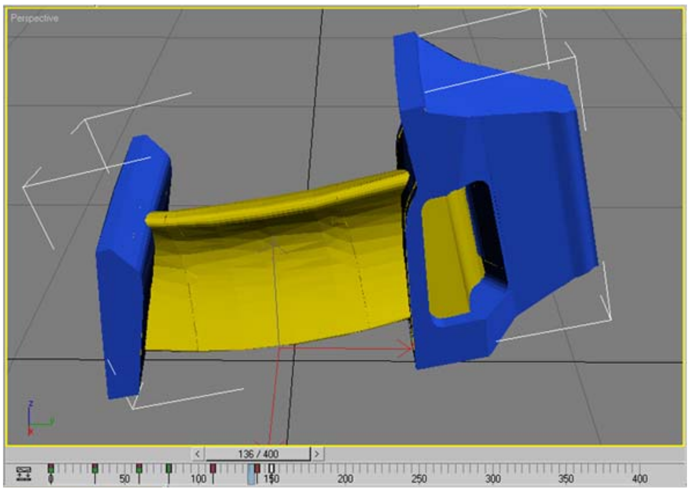


Fig.3-28. Prescribed Animation for Turbine Blade Assigned with Key frames

3.4.2.9 Multimedia

Virtools Dev is a VR authoring platform, used to integrate media and interactivity. Sound plays in virtools world by importing *.wav, *.mid or *.mp3 files etc and adding relevant BBs to activate them. The files need to be optimized in the audio software in advance. Using suitable behaviour programming, a series of sound effects can be generated in Virtools. Virtools allows frequency filtering and modifying the pitch and speed of sounds. Furthermore, sounds can be adjusted to be stereoscopic and fluctuate in volume depended on the angle and distance that the user keeps away from the object. This object is assumed to be propagating the sound.

Video plays in Virtools environment by applying *.avi or *.mpg file to a texture placed in a surface of a 3D object or 2D entity called '2D Sprite' placed on 2D front interface. The video clips require to be stored separately with the VE file as an external component because Virtools world can not synthesize video files into the VE file as a whole, which is similar to what sound does.

3.4.2.10 Input Devices

Virtools world supports a series of external devices which include VR input devices e.g. data gloves, 3D trackers, joysticks, spacemouse, and wands etc; VR displays e.g. VR headsets, active/passive stereo and multi-screen etc. They are associated with a group of corresponding BBs embedded inside Virtools to meet the requirements for the management of these devices. The research in this thesis focuses on the desktop-based VE construction at the preliminary stage. It aims to incorporate VR into real industrial applications where the standardization requirement needs to be met in order to widely spread the use of the VFDAS system. In addition, data gloves are the future research direction, which could be integrated into current version of VFDAS in order to improve intuitive manipulation and interaction. This would allow the fixture design process to be carried out in more natural way using VFDAS.

3.4.3 The Approach for Fixture Model Construction

3.4.3.1 3D Fixture Modeling in Pro/E

Pro/ENGINEER Wildfire is the CAE software that is used in mechanical engineering education throughout the degree courses at the University of Nottingham. This software allows the users to create 3D solid models and visualize parts and assemblies. Pro/ENGINEER was used to carry out the research in this thesis as the major CAD system and construct fixture models and produce engineering drawings of fixture assemblies.

Pro/ENGINEER models are created using a series of features. Each feature builds upon the previous feature, creating the model one feature at a time. Each feature is simple, but collectively forms complex parts and assemblies. Pro/ENGINEER models are driven by dimension values. If a dimension of a feature is changed, that feature will update simultaneously. While users fulfil a fixture design process from concept to completion in Pro/ENGINEER, users circulate the design information through three design steps: (a) creating the fixture parts of the design, (b) joining the fixture parts in a fixture assembly that records the relative positions of the parts, (c) creating mechanical drawings based on the fixture parts and fixture assembly. In addition, Pro/ENGINEER considers these steps as separate modes, each with its own features, file extension, and relation with the other modes. The change of your design at any mode, Pro/ENGINEER automatically reflects it at all mode levels.

Conceptual fixture designs start in Part Mode. The fixture parts created in part file (*.prt) will be associated together in a fixture assembly file (*.asm). Part Mode lets the users create and edit the features e.g. the extrusions, cuts, blends, and rounds which compose each fixture part users are modelling. Most features begin with a 2D section. As soon as the section is defined, a third dimension value needs to be assigned on it to make it a 3D shape. The 2D section is usually generated in a tool called Sketcher. Sketcher allows users to draw the section with lines, angles, or arcs, and then modify the precise dimensional values. The users use an interface called the Dashboard to enter and exit Sketcher mode in order to edit other 3D geometry features. A typical case study of the prismatic workpiece shown in **Fig.3-29** whose fixture models and assembly model are created by using Pro/ENGINEER, was employed to present the system operational mechanism within VFDAS as a demonstration example. More details with regard to this example will be introduced in section 5.3.1.

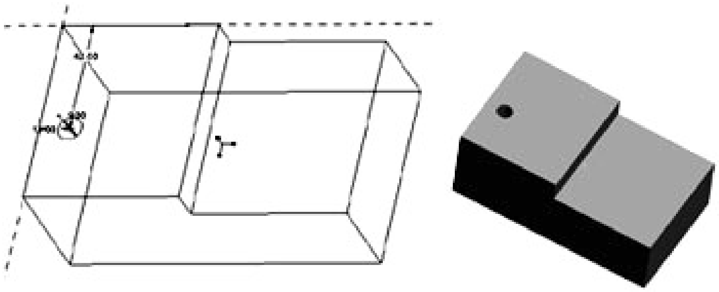


Fig.3-29. The prismatic workpiece in Sketcher and 3D Mode

Once the fixture parts that comprise a fixture assembly model are created, an empty assembly file for the fixture models will be created, in which the individual fixture parts are assembled. Via this process, the fixture parts are mated, aligned or inserted to the positions in the final fixture assembly. In a fixture assembly, the Explode Views are used to display or examine the fixture part relationships. The Explode View in relation to the fixture assembly of the prismatic workpiece is presented below **(Fig.3-30)**.

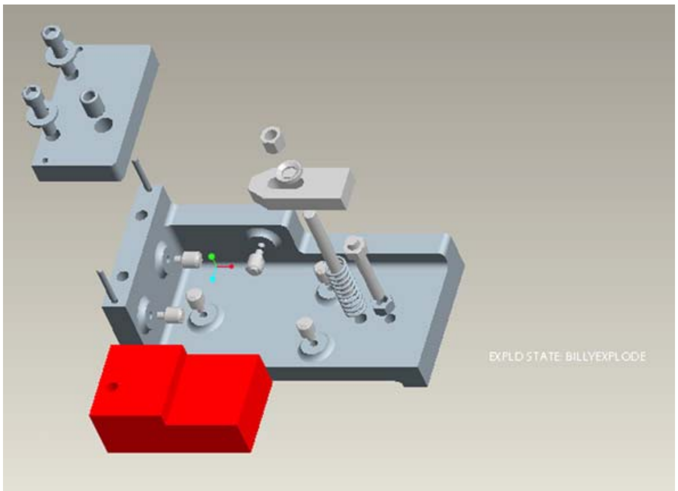


Fig.3-30. The fixture assembly in exploded view

The mechanical drawings of the fixture design can be created in Pro/ENGINEER Drawing Mode based on the dimensions recorded in the 3D fixture part and assembly files. Users can selectively show and hide dimensions that are inherited from the 3D fixture models. All information objects such as: dimensions, notes, surface notes, geometric tolerances and cross sections etc that are created for 3D fixture model can be passed to the Drawing mode. These objects remain associated with fixture models and may be edited to change the original 3D fixture model from the drawing.

3.4.3.2 The Design of VFDAS Motivated By Students' Needs

During the period of the PHD research, the author was involved in supporting laboratory classes for Design for Manufacture module of the Manufacturing Engineering and Management course at the University of Nottingham. In this module, Pro/ENGINEER was the CAD toolkit used by students to complete their fixture design assignment.

This opportunity allowed the author to understand the difficulties that students experienced in understanding the principles of fixture design and assembly. These engineering students can be regarded as novice fixture designers, and therefore junior system users of VFDAS. Accordingly, the requirements of junior users were identified and incorporated into the proposed VFDAS system so that the VFDAS can facilitate the junior fixture design process in VFDAS. For instance: the requirements of rapid fixture element selection, fixturing location identification and design interference check etc. The requirement of rapid fixture element selection indicates that the well-classified fixture part library within VFDAS enables the student fixture designers to make a decision for a fixture configuration design. The requirement of identifying fixturing location indicates that the geometric reasoning function in VFDAS relating to the 3-2-1 locating principle would help students determine the appropriate fixturing location. The requirement of design interference check indicates that student fixture designers normally encounter the issue of eliminating the design collision and conflict error during a fixture design process. VFDAS aims to establish an inbuilt solution to ensure a collision-free fixture design. Moreover, there are three types of interference often identified in the fixture design: (a) interference between fixture elements or sub-assemblies, (b) interference between fixture elements and the machining envelope, (c) interference between fixture elements and the workpiece [11].

According to the previous teaching experience, the specific requirements arising from junior fixture design process were used to identify the essential functionality of VFDAS as a useful guidance. Thus, the proposed VR system was developed with essential functions to support the use of novice fixture designers. As a result, the educational applications are comprised into the system as an additional value to further improve the versatility of VFDAS.

3.5 Flexible Development Method for VEs Usability Design

Even if there has been much research devoted into the development of Virtual Reality toolkits to ease the construction of VEs during last decade, such as: AVIARY [65], MASSIVE [66], SVE [67] and so on, few research has been conducted to examine how best to use these toolkits to create VEs that are applicable. Even less research has explored how to carry out the functionality that aims to provide VEs required utility. Furthermore, low level research which holds great importance to VE usability is in the field of offering cross platform toolkits for the development of VEs [68]. This would allow VE developers to create applications with a consistent user interface and a facility available to those building 2D WIMP (windows, icons, menus and pointers) based applications.

A series of VEs in this research were created for various VR applications in terms of fixture design. These VEs comprise the VR-base simulation for engineering education, the VR simulation for fixture assembly analysis and the VFDAS system development etc. All these VR applications are derived from the research theme “Virtual Reality for Fixture Design and Assembly”. The intended user groups for each of these proposed VEs ranged from engineering students, fixture designers, fixture assembly engineers and specialists from different domains. They would be expected to use these VEs as a communication platform for fixture development and evaluation.

In order to ensure suitability of VFDAS development for these different users, a user-centred method was adopted, although this was not a rigid method, but a fluid, dynamic process. This method consists of a series of techniques employed at various stages within the VFDAS design cycle. A variety of approaches were applied to gather, evaluate and use information to support the process of virtual environment design. These are classified into three core areas which include Design guidelines, Design specification and Prototyping presented in the following paragraphs [69].

3.5.1 Design Guidelines

A user-centred design method involves representative users during the experiment or evaluation phase of the VE development process so that VE solutions are appropriate to user's tasks and requirements. However the users play the role of design partners in participatory design. The development process of VE is often half way between user-centred and participatory with the users involved at appropriate levels throughout [70]. A user centred design approach is a remarkable means to ensure that a VE is both usable by, and has utility for, its target user population [71]. In addition, many insightful observations are formed in consensus [72], at the First International Workshop on Usability Evaluation for Virtual Environments (UEVE'98):

- Guidance require to be swift and flexible to apply, and informal to reinforce design practice.
- The reasons for the significance of using the guidance should be comprised to provide motivation to designers.
- Current successful games and VEs could be investigated to accumulate VE development techniques.
- User centred design should be encouraged.
- Usefulness and relevance of a VE application to an organisation need to be considered.
- Standards for VEs reduce the learning processes required to use a VE and allow users to identify features immediately, but are constrained on design and incur less fascinating VEs.
- The VE design process needs to be understood before proposing guidance.

As a result, one of the obstacles the author had to overcome was to understand the VE design process, which is broader than just VE development.

To some extent, VEs are developed in compliance with design process guidance for any human-machine system or human-computer interaction, of which there are great numbers [73] [74]. As soon as one of the issues with respect to VE development structure is explored and solved, it may be possible to develop a design recommendation that can be implemented universally using a user-centred design method. These guidelines assist VE developers to create useful VEs and enhance the efficiency of VE development [75].

There are two major categories incorporated with VE system usability: the VE system interface and the VE user interface. The VE system interface includes software, hardware and overall man-machine interaction design. The VE user interface comprises physiological, psychological and psychosocial factors and considerations [76]. Guidelines for VE design and usability development, MAUVE system **(Fig.3-31)** (the Multi-criteria Assessment of Usability for Virtual Environments) were addressed by Stanney et al in 2002. The MAUVE system provides a structured approach for achieving usability in VE system design and evaluation [76]. For instance: interaction should be natural, efficient and appropriate for target users, task goals and domains [77]. Therefore, according to MAUVE system's guidelines, usability criteria associated with interaction are sub-classified as: wayfinding (i.e. locating and orienting oneself in an environment); navigation (i.e. moving from one location to another in an environment); and object selection and manipulation (i.e. targeting objects within an environment to reposition, reorient and query). All relevant guidelines in relation to multiple aspects of VE design and usability evaluation are incorporated into MAUVE.

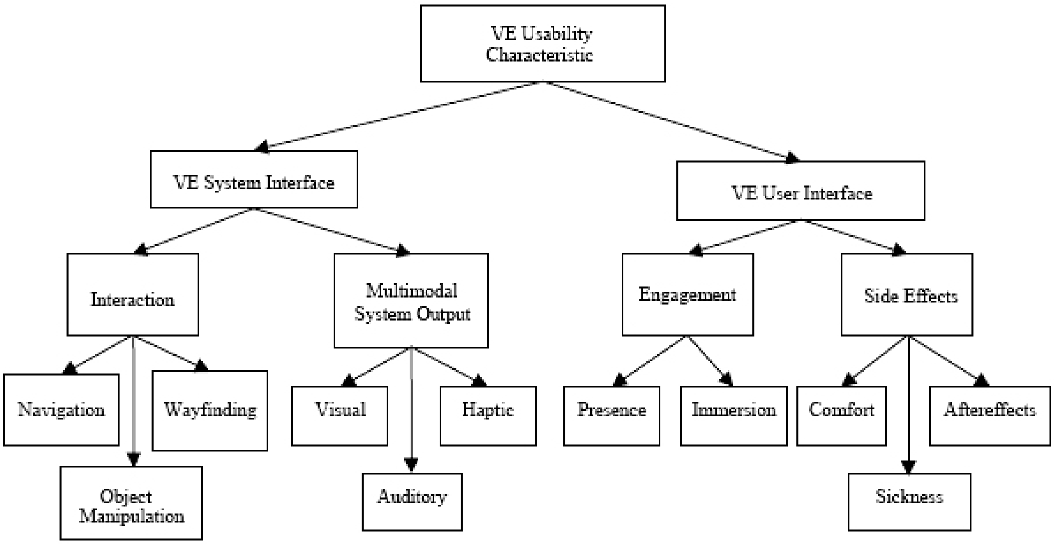


Fig.3-31. Usability design guidelines of VE addressed by MAUVE

Another framework that demonstrates usability design guidelines to help VE designers in specific circumstances was created by Gabbard et al in 1997. This study aimed to find a way to integrate the various existing usability research in human computer interaction and apply it to VE development. Thus, Gabbard et al provide 195 guidelines on VE design related issues [78]. They refer to object selection and manipulation, user goals, fidelity of imagery, VE context, locomotion, the use of metaphors, the use of avatars (computer controlled characters) as well as contribute guidance on the physical aspects of the VR system etc. Similarly, eight golden rules Schneiderman addressed in 1998 [79] and ten VE design principles Neilsen raised in 1994 [80] related to interface design are usable in the VE design process because there are many common issues which influence both a VE and a 2D interface with respect to the requirement for the user to enter information. Furthermore, the conclusion that there are no comprehensive guidelines for considering user issues was outlined by Kaur in 1998. She constructed a suite of 46 guidelines in the form of 'design properties' to be applied as a checklist to tackle VE design and usability issues and provide general solutions. These guidelines are used to handle interaction in VEs which covers the areas of spatial layout, user representation, viewpoint, objects, initial system behaviour, action, action feedback and the user task etc [81]. Each guideline is explored carefully and provides suggestions about what cues are needed to allow the users to understand and interact with a VE.

Alongside a series of VE usability design guidelines described above, Virtual Environment Development Structure (VEDS), a typical model that depicts a number of phases of decision making that VE developers will encounter, was conceived by Eastgate in 2001 [75]. This model aims to offer them with relevant guidance at each design stage. VEDS proposes seven major stages of VE development: 'Specification', 'VE Overall Design', 'Resource Acquisition', 'VE Design Detail', 'VE Building', 'VE Testing' and 'Implementation'. This is illustrated in **Fig.3-32**.

On the contrary, the argument that many existing guidelines and models are not always useful and effective for VE designers, was suggested by Wilson et al in 2001 because these guidelines and models usually relate to WIMP interface based styles rather specifically. New feasible guidelines are not always convenient to conceive [82]. In addition, the effort needed to model the objects and actions of the real world and the nature of the user cognitive interface imply that VE development also requires guidance unique to its own particular characteristics and requirements. Moreover, VEs have different attributes for their own structured development frameworks to be required, even though similar development guidance is used in VE simulation design [75].

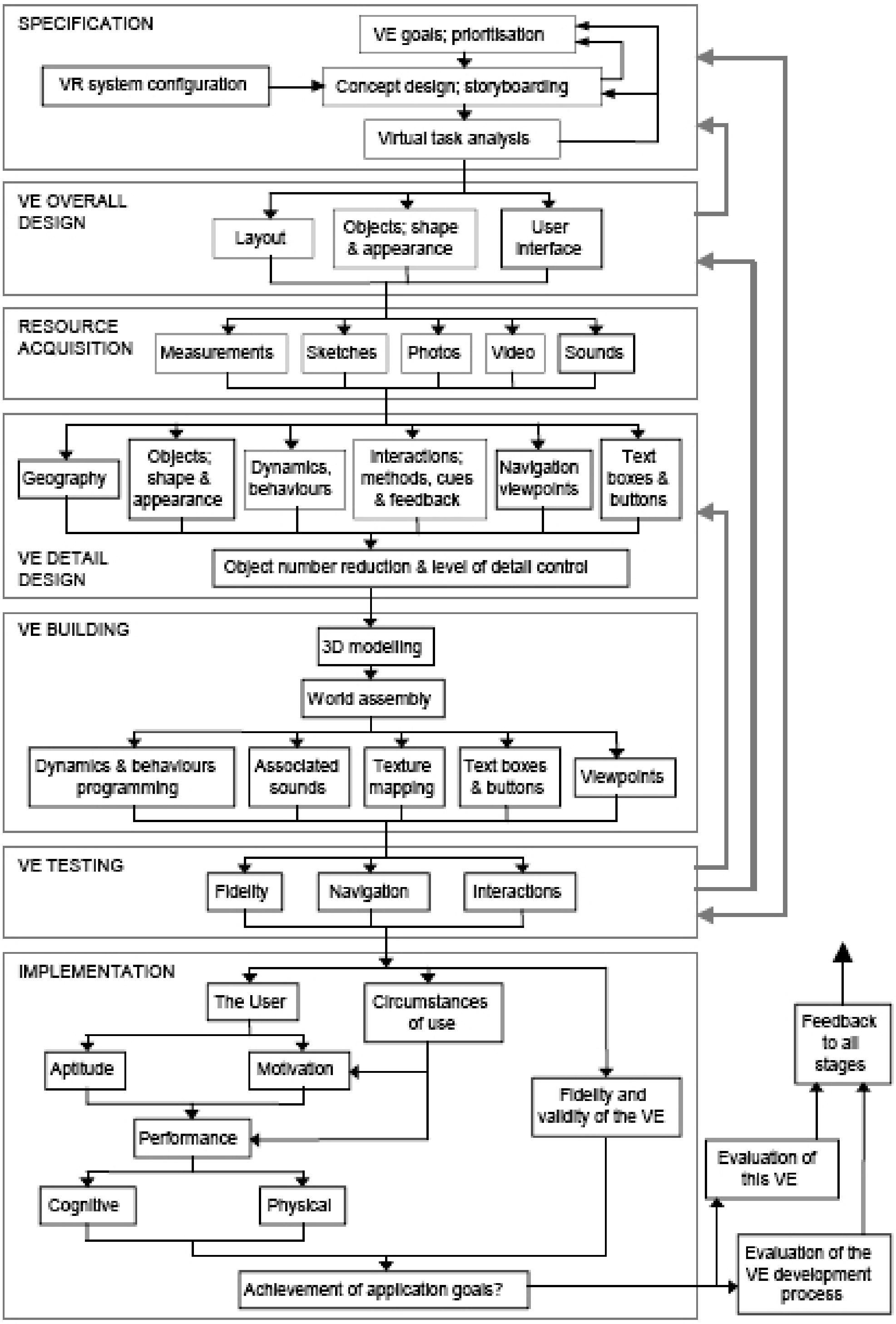


Fig.3-32. VEDS: Virtual Environment Development Structure (Eastgate, 2001)

There are no specific guidelines available for fixture designers who are the main users of VFDAS system. They are classified into both amateur fixture designers and fixture specialists. However, the existing usability guidelines in respect of general interface are incorporated with guidelines for VE design in order to improve many aspects of the VE design and create the basis for a design at least [75]. To obtain useful criteria for VE design in VFDAS, these existing guidelines were also combined along with the experience from the VE developer in constructing similar VEs and literature reviews of fixture development. Thus, it was important to use a cross-disciplinary approach to conduct the development of VEs in VFDAS so that all VE design requirements were considered and reflected.

3.5.2 Design Specification

Developing useful interactive virtual environment system is a new challenge for system developers and human factors practitioners [83]. Conventional usability principles do not consider characteristics related to VE systems, e.g. the design of wayfinding and navigational techniques, object selection and manipulation as well as the synthesis of audio, visual and tactile outputs [84]. To deliver the optimal VE design, the immersion, presence, interaction and system comfort also have to be considered by VE developers during a process of VE system construction [76].

A relatively appropriate approach about how to determine the design specification for VE development was clarified by Eastgate in 2001 [75]. This approach is partially affiliated into the VEDS model **(Fig.3-32)** which is used to fulfil the process of VE design and usability evaluation. To determine VE design specification, VE development is divided into three key steps: VE Goals, Concept Design and Virtual Task Analysis [75]. Initially, the user population provide ideas for which VE need to be developed, but VE developers may be aware that the proposed ideas are either intangible or unachievable. Therefore, the first step of VE specification is to identify a list of practical goals which provide the potential VE the necessary utility. Concept design is the step of determining how the goals will be achieved using VR technology. Background review and preliminary resource acquisition are required. Subsequently, the proposed VE specification for design is confirmed. Finally, a Virtual Task Analysis is analysis of the tasks to be conducted by users in the VE. The first purpose of this is to ensure whether users will find it easy to carry out the operations in the VE. The second purpose is to explore whether the operations developed in the VE are beneficial for fulfilling the goals of the VE. The third purpose is to identify health related issues that may result from using the VE, e.g. simulator sickness.

In this research, the preliminary procedure was to show existing VEs acquired from pilot study cases to both amateur fixture designers and professional fixture designers. An assessment with regard to the usability, appropriateness, acceptability of these pilot VEs was conducted amongst fixture designers. A review of characteristics from fixture designers was also required to be carried out before achieve the goal of developing VFDAS system.

To obtain a usable VE design specification of VFDAS, a set of approaches were applied in the research. These included the observation of user performance in a task related to fixture development, contextual interviews with questionnaires, the identification of essential requirements by review of previous CAFD systems, and online hand-on VE trials for the VFDAS prototype with users' feedback. Moreover, a number of workshops were conducted as another user-centred approach where the potential VR technology and existing VEs were evaluated by fixture designers. Along with a series of interviews and questionnaires, the requirements of a new VE in VFDAS system were determined.

3.5.3 Prototyping

Another approach used in this research to help build effective, user centred VEs, was to develop design prototypes which were used in early discussion amongst fixture designers. These prototypes were used to determine task objectives, understand the characteristics of fixture development and identify the function, VE layout and user interaction required by users of the proposed VFDAS system. VE prototypes were early versions of the proposed VE with incomplete functionality and content. These were useful for the evaluation of VE usability design to determine overall design specification.

Observation of user behaviour and interaction with the VE prototypes was used to determine how and where the system users require additional assistance and cue messages from the proposed VFDAS system. During the process of task analysis, the fixture designers were involved to implement related activities via which metaphors and user interaction approaches were used to simulate how the full VFDAS system would operate. These approaches were also applied to identify where users had difficulties in order to determine what type of support the fixture designers might require in the VFDAS system.

3.6 Summary

The main methodology used for the research in this thesis was an incorporated means which comprises various research areas, e.g. fixture development, VE behaviour programming, computer aided design, the integration of VR and CAD and even usability design etc.

In addition, the practical methodology that was applied to perform the total technical procedure of developing the VFDAS system is also presented according to the concise flowchart shown below **(Fig.3-33)**.

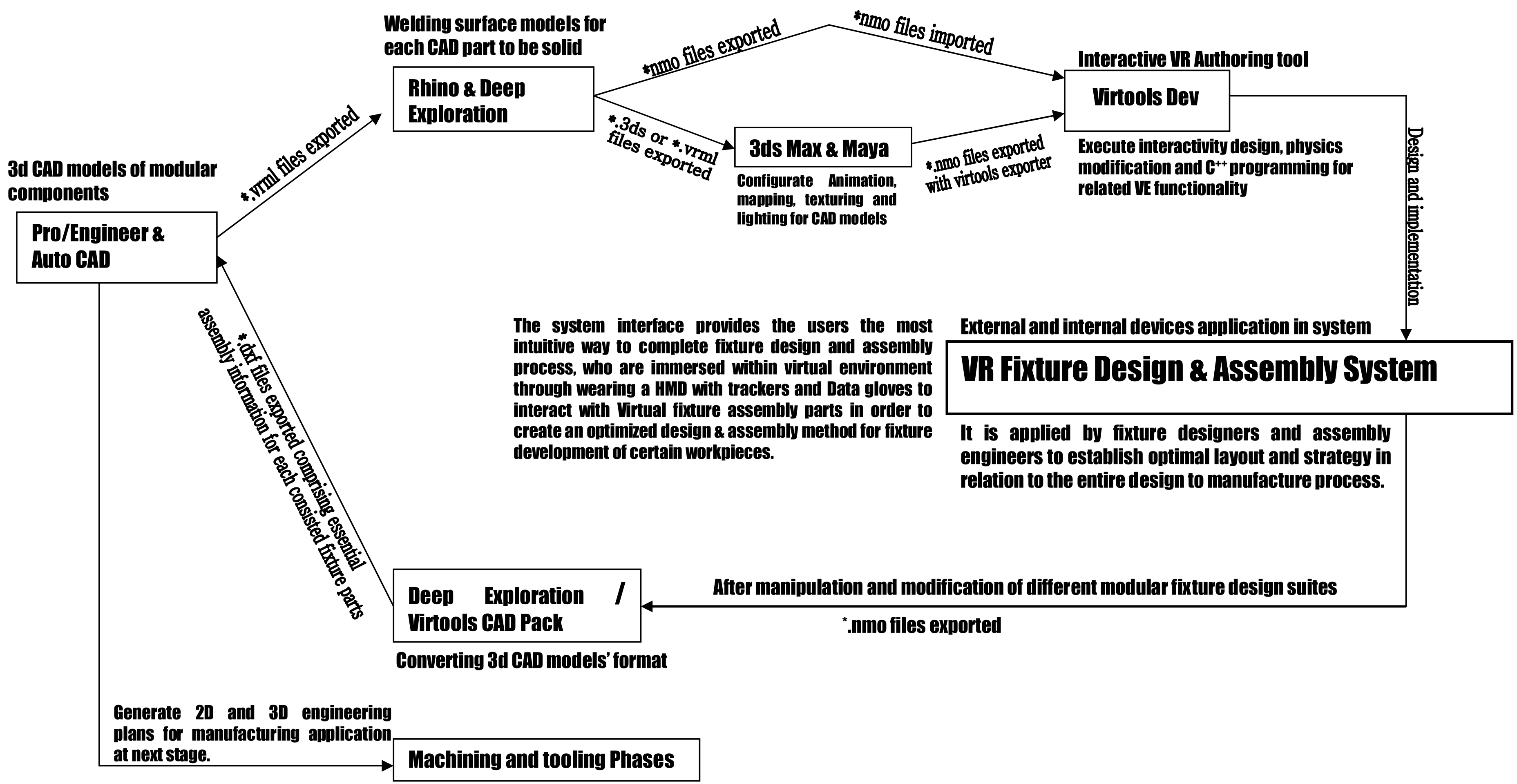

3d CAD models of modular components
Pro/Engineer & Auto CAD
*.vrml files exported
Welding surface models for each CAD part to be solid
Rhino & Deep Exploration
*nmo files exported
*nmo files imported
*.3ds or *.vrml files exported
3ds Max & Maya
Configurate Animation, mapping, texturing and lighting for CAD models
*.nmo files exported with virtools exporter
Interactive VR Authoring tool
Virtools Dev
Execute interactivity design, physics modification and C++ programming for related VE functionality
Design and implementation
External and internal devices application in system
VR Fixture Design & Assembly System
It is applied by fixture designers and assembly engineers to establish optimal layout and strategy in relation to the entire design to manufacture process.
The system interface provides the users the most intuitive way to complete fixture design and assembly process, who are immersed within virtual environment through wearing a HMD with trackers and Data gloves to interact with Virtual fixture assembly parts in order to create an optimized design & assembly method for fixture development of certain workpieces.
After manipulation and modification of different modular fixture design suites
*.nmo files exported
Deep Exploration / Virtools CAD Pack
Converting 3d CAD models' format
*.dxf files exported comprising essential assembly information for each consisted fixture parts
Generate 2D and 3D engineering plans for manufacturing application at next stage.
Machining and tooling Phases

Fig.3-33. Practical Development Methodology Adopted in VFDAS

Chapter 4. Pilot Case Studies

4.1 Introduction

In the early stages of the PhD research, the author was invited to develop several interactive VR simulations that support fixture design for different applications. This provided an opportunity to treat them as pilot case studies that could be used to determine the appropriate methodology and estimate potential difficulties and requirements involved in development of the VFDAS system. These pilot studies were also used to explore the advantages of interactive VR technology that could be used to support the fixture design, assembly planning and optimization process.

This chapter presents two pilot case studies that were derived from real industrial applications. The first, commissioned by the UTC research group at the University of Nottingham, required an interactive desktop-based VR simulation of the fixture assembly for a casing component mounted within an airplane engine. This was for the purpose of detecting design errors and assembly interference as well as gathering the fixture design data and requirements. The second, commissioned by Rolls-Royce Limited, required an interactive VR learning tool related to the fixture design for a turbine blade. This was used for the purpose of teaching manufacturing students to make more conceptual understanding about fixture design and assembly process.

The following sections describe each case in detail and examine specific aspects related to the construction of the proposed VFDAS, such as: the integration method of CAD and VR systems, the means of constructing fixture assembly models, the usable theory of fixture development, VE behaviour programming, the approach of VE usability design, the technique of conducting the prescribed animations and the method of executing light mapping and texturing etc.

4.2 Pilot Case Study One: VR Simulation for a Casing Component

4.2.1 Project Summary

This project aimed to develop an interactive desktop-based VR simulation with regard to the modular fixture design for a casing component within airplane engines. For the industrial sponsor, this VR research project had four main objectives: (1) to explore and identify how the interactive VR simulation improves the fixture design at early construction design stage. (2) to optimize the assembly planning process and discover potential improper designs and assembly interference prior to the real manufacturing process. (3) to act as an effective role of communication between fixture designers and other experts from different domains. (4) to recognize and gather the relevant fixture design data and requirements associated with a VR simulation study [26].

4.2.2 The Problem for Fixture Design & Assembly

The conceptual design of fixture elements and assembly are carried out by using CAD software such as Pro/engineer or Auto/CAD etc. After the conceptual design process, the assembly engineer who is responsible for integrating the fixture components into assembly workstation uses 2D and 3D mechanical drawings to design detailed assembly procedures. This technique sometimes causes potential assembly problems.

The assembly engineer is not often engaged with the previous construction process. Lack of communication between engineers in different roles is regarded as the main obstacle during the fixture design process. The overall process from the conceptual design to assembly planning for the fixture development gives rise to many unexpected problems which are often solved by awkward processes [26].

Many design problems that emerge in latter fixture assembly planning can be recognized and eliminated at the early construction process. In terms of the concept of concurrent engineering, the interactive VR simulation is identified as an appropriate method to avoid these design errors and assembly interference in order to optimize the fixture design and assembly plan.

4.2.3 The Interactive VR Simulation System for Fixture Assembly

The main purpose of creating the interactive VR simulation system was to explore the possibility of using VR technology to support the conventional fixture design and assembly process of a casing component. The finished simulation is shown in **Fig.4-1**.

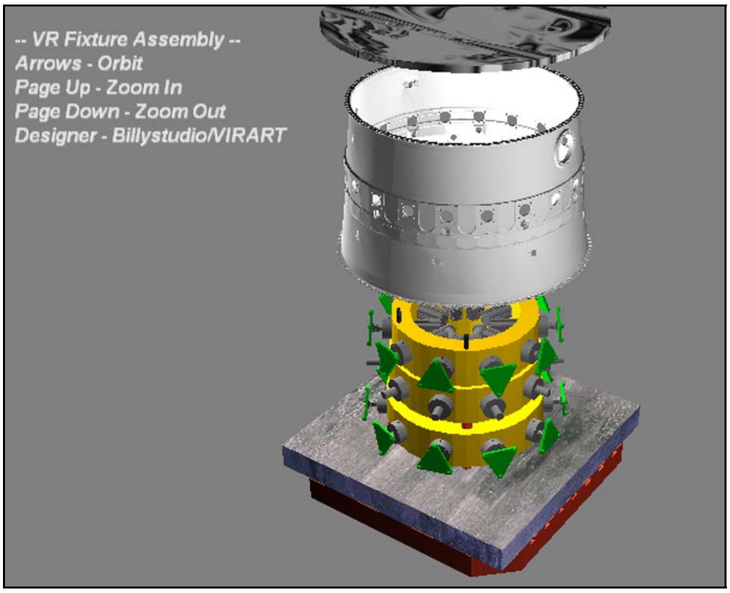


Fig.4-1. Interactive VR-Based Simulation for a Casing Component

For this case study, we used the lowest level of desktop-based VR system in order to reduce costs and maximize benefits. This could also enhance the possibility of more widespread use of the simulation during the practical fixture design process. The interactive VR simulation system can be easily operated by industry users using the standard keyboard interface. Keys are used to switch between different viewpoints and angles, and zoom in-out of the fixture model. Instructions for the control panel are presented clearly inside VR interface, as can be seen in **Fig.4-1**.

Experts from different fields were able to observe the pre-determined assembly process in detail, which comprises the fixture assembly sequence and trajectory. It was hoped that they communicate with each other more efficiently so that the process of concept delivery is facilitated by using the interactive VR simulation system. Subsequently, the related data analysis with regard to fixture design and assembly is also conducted. The fixture design and assembly plan are now examined for final decision making and it is possible to identify critical aspects of fixture element design during the assembly process by both the fixture designer and assembly engineer. These constituent fixture elements can be reversed back into the redesign procedure for further development until the optimum fixture design is achieved.

4.2.4 Developing the Interactive VR Simulation

The main procedures that were used to complete this interactive VR simulation for the casing component can be found in section 3.3.2 in Chapter 3. In particular, **Figure 3-6** in that section graphically describes how these procedures were carried out. **Fig.4-2** shows the model of the casing component.

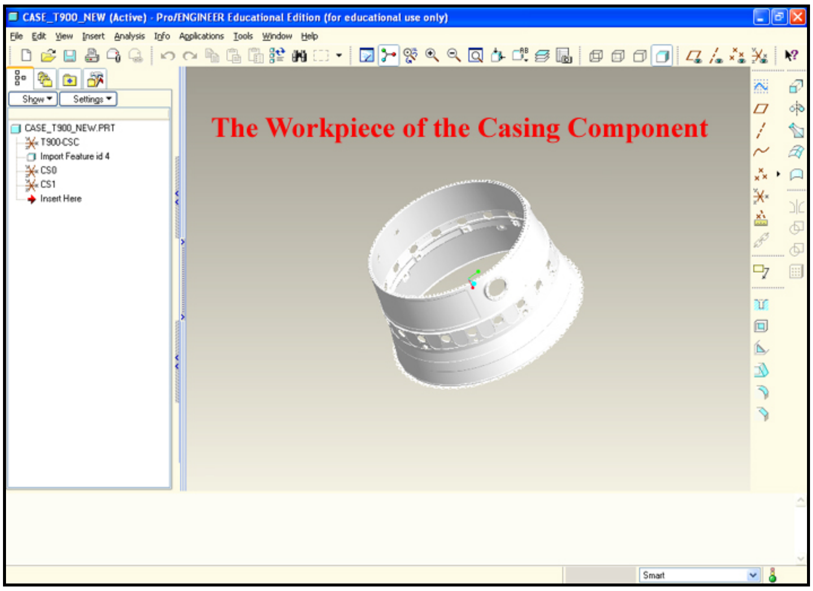
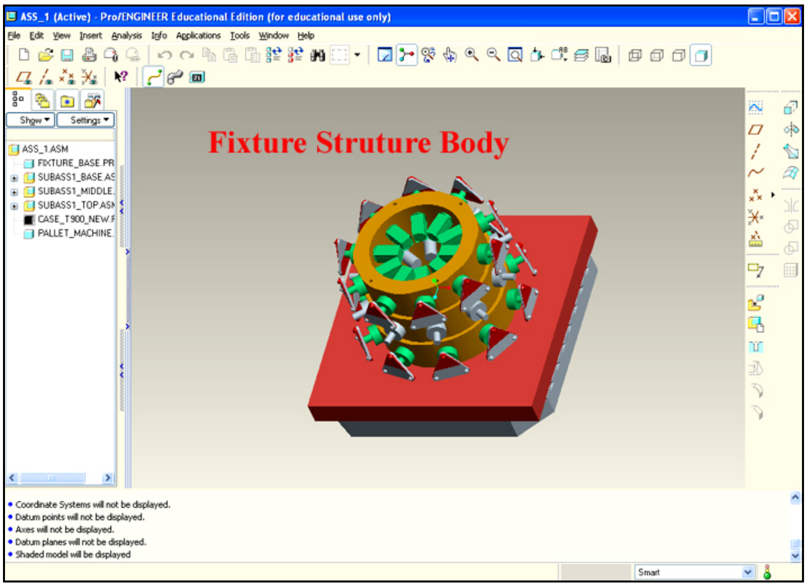

CASE_T900_NEW (Active) - Pro/ENGINEER Educational Edition (for educational use only)
File Edit View Insert Analysis Info Applications Tools Window Help
Show
Settings
CASE_T900_NEW.PRT
T900-CSC
Import Feature id 4
CS0
CS1
Insert Here
The Workpiece of the Casing Component
Smart

Fig.4-2. The Workpiece Named the Casing Component

This section will focus on details related to the integration of both the VE behaviour programming using Virtools and the creation of prescribed animation using 3D Studio Max.

4.2.4.1 The Creation of the Fixture Assembly Model

The complex model of the fixture assembly **(Fig.4-3)** for the casing component was constructed at early conceptual design stage using Pro/ENGINEER. More details about how to construct the fixture assembly model in Pro/ENGINEER can be found in previous section 3.4.3.


ASS_1 (Active) - Pro/ENGINEER Educational Edition (for educational use only)
File Edit View Insert Analysis Info Applications Tools Window Help
Show Settings
ASS_1.ASM
FIXTURE_BASE.PR
SUBASS1_BASE.AS
SUBASS1_MIDDLE.
SUBASS1_TOP.ASM
CASE_T900_NEW.F
PALLET_MACHINE.
Fixture Struture Body
Smart

- Coordinate Systems will not be displayed.
- Datum points will not be displayed.
- Axes will not be displayed.
- Datum planes will not be displayed.
- Shaded model will be displayed

Fig.4-3. The Fixture Assembly Model Created in Pro/ENGINEER

4.2.4.2 The Creation of the Prescribed Animation

The industry requirement for this case study was that the sequence of fixture assembly would be interactively simulated. This meant that the prescribed animation requires to be played in this proposed VR simulation. The prescribed animations are associated with animated objects and are defined using the 3D Studio Max. In 3D Studio Max, each animated 3D object comprises an independent animation which is considered as a pre-defined movement assigned to the corresponding object. Each prescribed animation associated with an object consists of a set of keyframes, where each keyframe includes a position, scale and orientation information **(Fig.4-4)**. Each keyframe is created and edited by specifying values associated with the 3D object regarding its position, scale and orientation after ‘Auto Key Mode’ is switched on. Using ‘Auto Key Mode’, the change of values starts to be recorded and the animation with a set of keyframes for an object is determined. The advanced animation edit used to handle complex motions is carried out by using the Curve Editor in 3DS Max. This Editor is regarded as a Track View Mode and allows motion expressed as functional curves on a graph in order to visualize the object motion and transformation. Curve Editor was considered as a tool to conduct the creation of complex animations. The Curve Editor **(Fig.4-5)** comprises a menu bar, a toolbar, a controller window, and a key window. There are also a time ruler, navigation and status tools at the interface bottom.

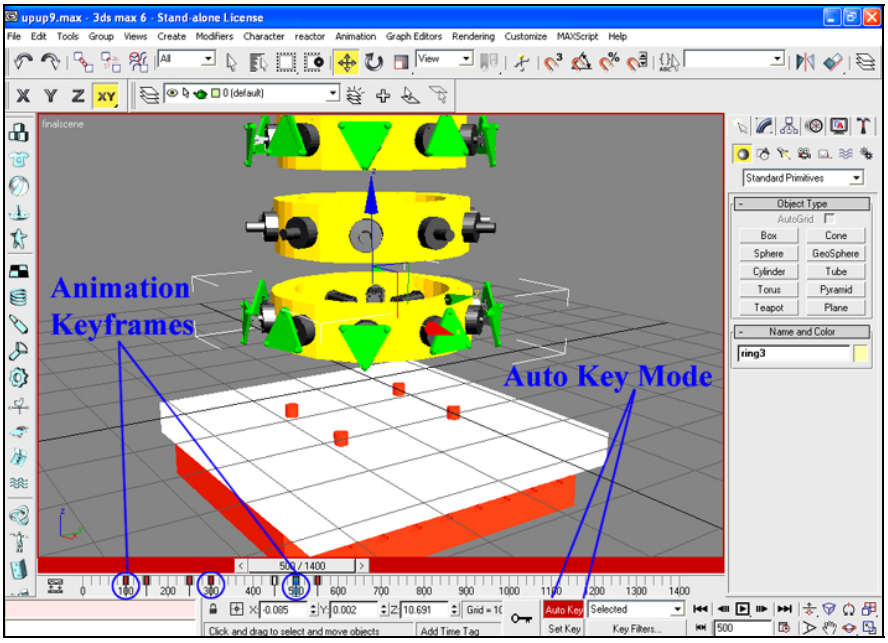


Fig.4-4. The Creation of a Prescribed Animation in 3DS Max

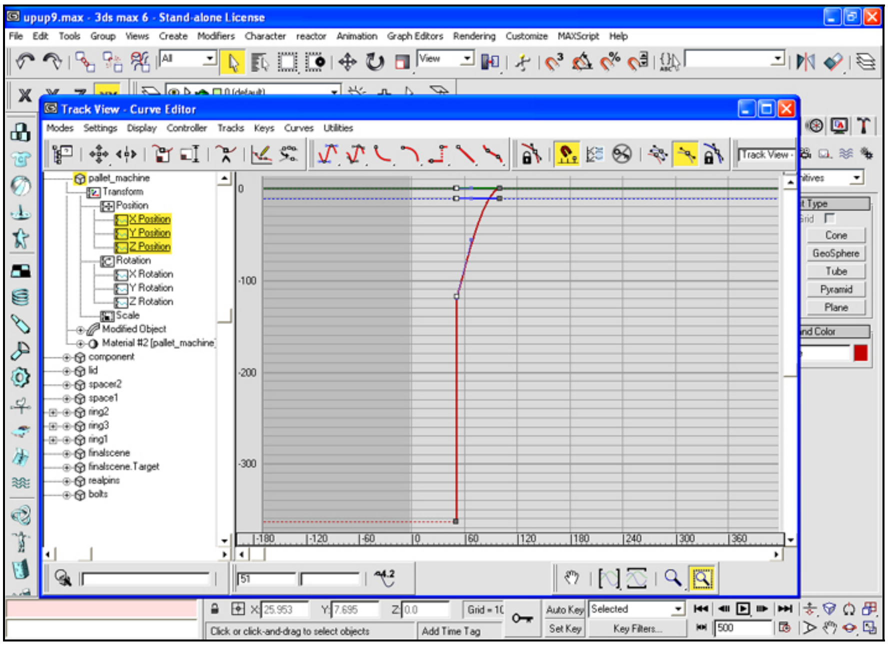
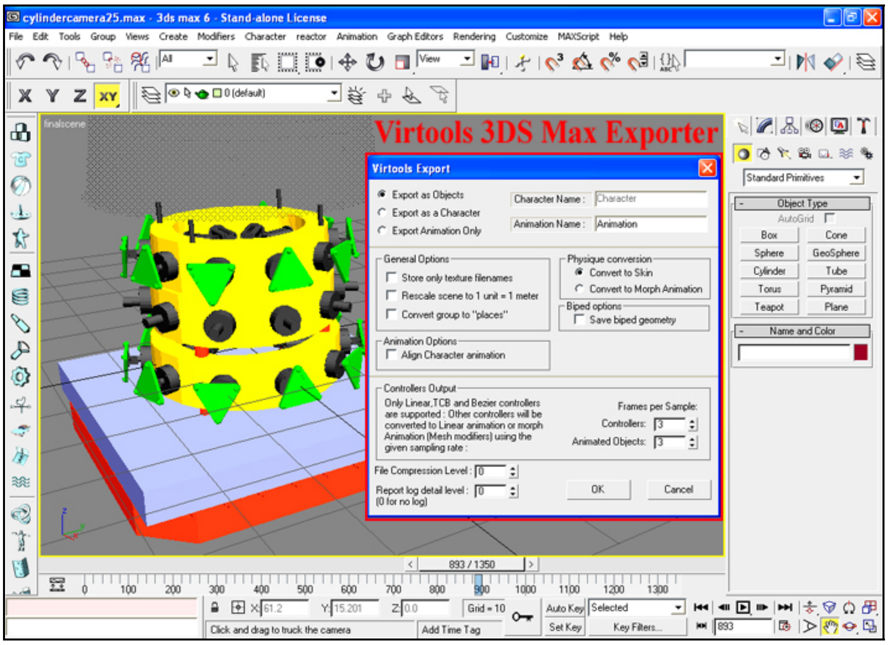


Fig.4-5. The Curve Editor for Advanced Animation Edit

After creating the prescribed animation, overall animated objects are then converted into Virtools to further add the interactions upon these objects and achieve the final interactive VR simulation. The conversion process of animated objects between 3D Studio Max and Virtools Dev are implemented by using the Virtools 3DS Max Exporter. Its settings and options related to exporting the animation are presented in **Fig.4-6.**


cylindercamera25.max - 3ds max 6 - Stand-alone License
File Edit Tools Group Views Create Modifiers Character reactor Animation Graph Editors Rendering Customize MAXScript Help
Virtools 3DS Max Exporter
Standard Primitives
Object Type
AutoGrid
Box Cone
Sphere GeoSphere
Cylinder Tube
Torus Pyramid
Teapot Plane
Name and Color
893 / 1350
Auto Key Selected
Set Key Key Filters...
Click and drag to truck the camera
Add Time Tag

Virtools Export

● Export as Objects
○ Export as a Character
○ Export Animation Only

Character Name : Character
Animation Name : Animation

General Options
□ Store only texture filenames
□ Rescale scene to 1 unit = 1 meter
□ Convert group to "places"

Physique conversion
● Convert to Skin
○ Convert to Morph Animation

Biped options
□ Save biped geometry

Animation Options
□ Align Character animation

Controllers Output
Only Linear,TCB and Bezier controllers are supported : Other controllers will be converted to Linear animation or morph Animation (Mesh modifiers) using the given sampling rate :

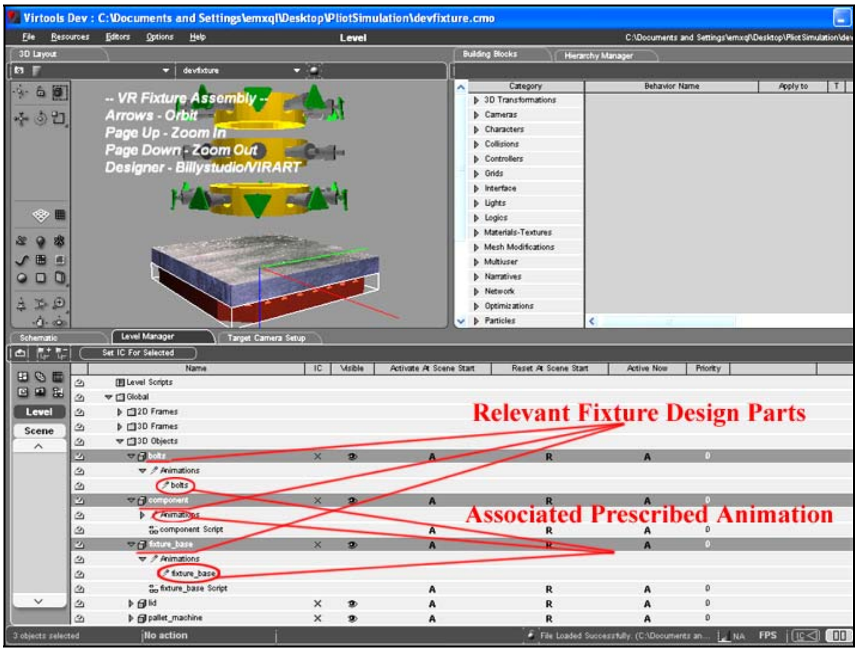
Frames per Sample:
Controllers: 3
Animated Objects: 3

File Compression Level : 0
Report log detail level : 0
(0 for no log)

OK Cancel

Fig.4-6. Export the Animated Objects with the Prescribed Animation

After the conversion process from 3D Studio Max to Virtools, each animated 3D object represents a fixture element that forms the fixture assembly model. The name of each fixture element is available in Virtools platform as the same as the name used in 3DS Max, e.g. ‘fixture_base’, ‘component’, ‘bolts’ and so on **(Fig.4-7)**. These fixture elements contain a set of their individual animations which jointly determine the entire fixture assembly sequence, movement, location and trajectory within the VR simulation.



Name	IC	Visible	Activate At Scene Start	Reset At Scene Start	Active Now	Priority
Level Scripts						
Global						
2D Frames						
3D Frames						
3D Objects						
bolts	X		A	R	A	0
Animations						
bolts						
component	X		A	R	A	0
Animations						
component Script			A	R	A	0
fixture_base	X		A	R	A	0
Animations						
fixture_base						
fixture_base Script			A	R	A	0
lid	X		A	R	A	0
pallet_machine	X		A	R	A	0

Fig.4-7. Fixture Elements with the Associated Animations in Virtools

4.2.4.3 The Behaviour Programming Process

After the conversion process, the behaviour programming process for the VR simulation is continuously carried out in Virtools. The prescribed animations in VR simulation will not be activated until the appropriate behaviour programming is applied to the target objects.

The BB of ‘Play Global Animation’ is normally used to make a group of 3D entities performing a global animation and cause the proposed animation to be executed as expected. As shown in the introductory graph **(Fig.4-8)**, the Behaviour Input called ‘On’ in this BB indicates that the BB process would be trigged while an activation flow is received by ‘On’. Another Behaviour Input called ‘Off’ represents that it would be deactivated while an activation flow is received by ‘Off’. The Behaviour Output called ‘One Loop Played’ represents that it is activated once the animation is played. The Parameter named ‘Animation’ is used to define the prescribed animation to be performed. The Parameter named ‘Duration’ is used to define how long the whole process lasts. The Parameter named ‘Progression Curve’ is used to define a 2D curve that represents the progression of the animation. The Parameter named ‘Loop’ indicates that if TRUE the animation is looped. Lastly, the Parameter Output named ‘Progression’ indicates the percentage between 0% and 100% which defines the progression of the BB process.

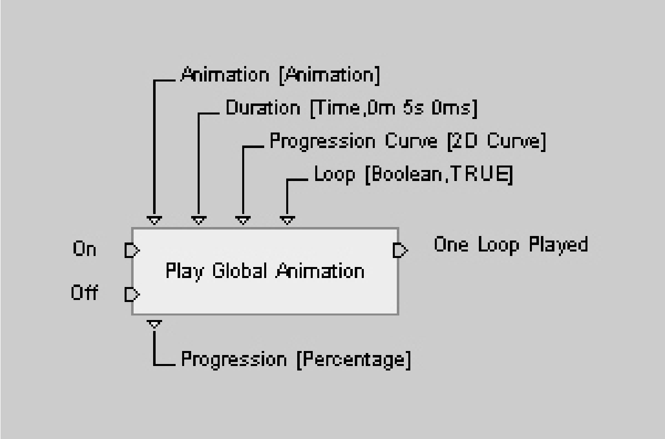
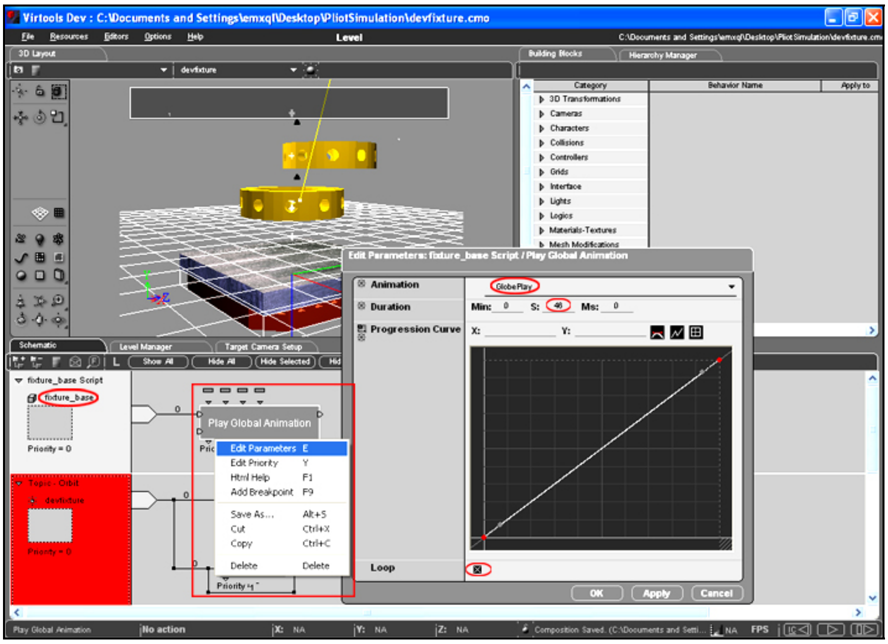


Fig.4-8. The Introductory Graph of the ‘Play Global Animation’ BB

In this interactive VR simulation, the ‘Play Global Animation’ BB is applied to the target 3D object named ‘fixture_base’ in the scene in order to activate and execute the global animation named ‘GlobePlay’. In addition, the four Parameters are specified together via Edit Parameters of this BB. The Parameter called ‘Animation’ is specified as the target global animation named ‘GlobePlay’. The Parameter called ‘Duration’ is specified as 46 seconds for playing the animation. The Parameter called ‘Progression Curve’ is defined as a 45° diagonal straight line. The Parameter called ‘Loop’ is checked to be ‘TURE’ which indicates the global animation is looped for replay **(Fig.4-9)**.


Virtools Dev : C:\Documents and Settings\emxql\Desktop\PilotSimulation\devfixture.cmo

Edit Parameters: fixture_base Script / Play Global Animation

Animation: GlobePlay

Duration: Min: 0 S: 46 Ms: 0

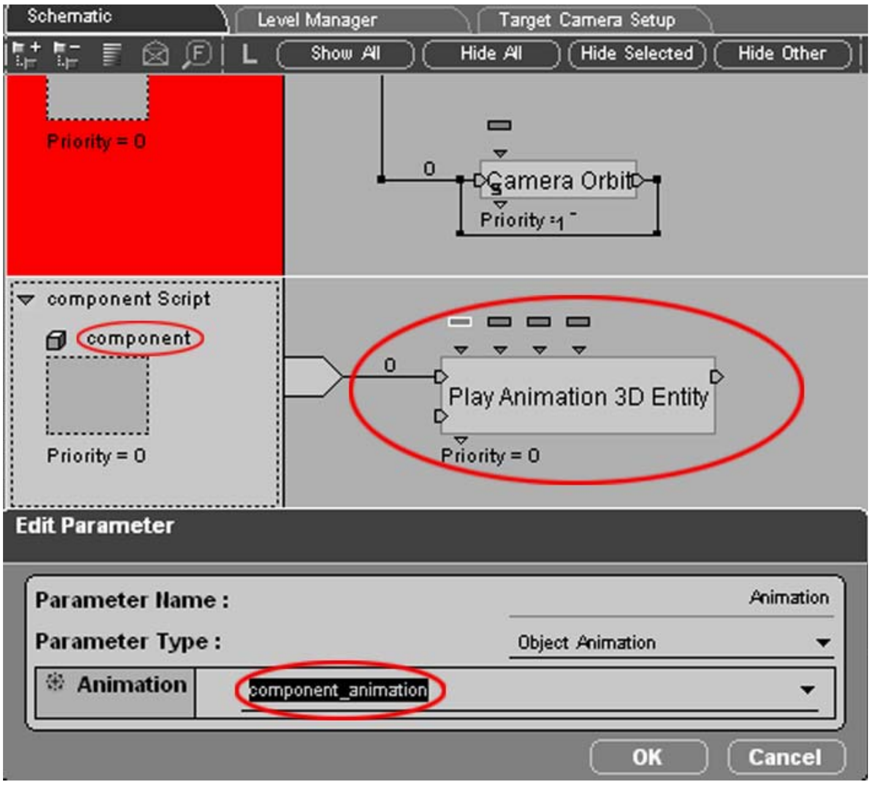
Progression Curve: X: Y:

Loop ☒

Fig.4-9. Specify Four Local Parameters for the ‘Play Global Animation’ BB

After performing the global animation in the simulation, the process of VE behaviour programming continues exploring the simulation issue related to the model of the casing component. The casing component model in this VR simulation is the complex geometry that often results in the simulation lag and a different behaviour programming approach needs to be applied to it so as to prevent this issue.

Therefore, the ‘Play Animation 3D Entity’ BB was applied to the casing component model named ‘component’ in the scene **(Fig.4-10)**. This BB allows an individual 3D entity separately performing an independent animation apart from the global animation. Using this BB, the independent animation called ‘component_animation’ **(Fig.4-11)** associated with the ‘component’ 3D object is performed properly. According to the introductory graph **(Fig.4-12)**, the ‘Play Animation 3D Entity’ BB comprises two Behaviour Inputs called ‘On’ and ‘Off’, the Behaviour Output called ‘One Loop Played’, a Parameter ‘Animation’, a Parameter ‘Duration’, a Parameter ‘Progression Curve’, a Parameter ‘Loop’ and a Parameter Output ‘Progression’.

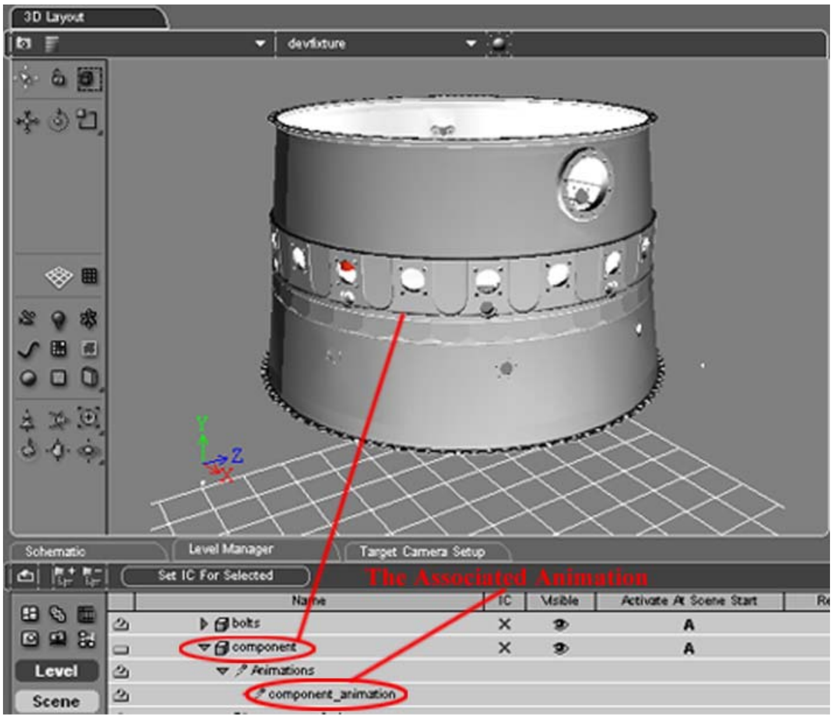


Parameter Name : Animation

Parameter Type : Object Animation

Animation: component_animation

Fig.4-10. The ‘Play Animation 3D Entity’ BB Applied to the ‘Component’ Object



Name	IC	Visible	Activate At Scene Start
bolts	X	👁	A
component	X	👁	A
Animations			
component_animation			

Fig.4-11. The Animation Associated with the ‘component’ 3D Object

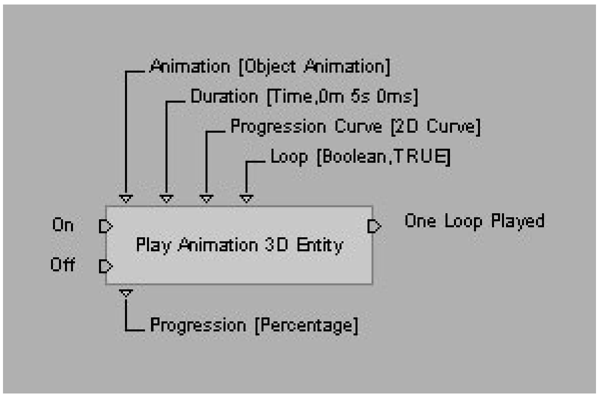


Fig.4-12. The Introductory Graph of the 'Play Animation 3D Entity' BB

To complete the simulation function of interacting and observing the fixture assembly from different angles, perspective viewpoints and distances, two BBs were applied to the behaviour programming process. One BB named 'Camera Obit' is applied to the assigned camera to make it orbit around a target 3D entity. Another BB named 'Set As Active Camera' is used to change the active camera in the scene in order to achieve the purpose of dynamically switching between different cameras. More details with regard to these two BBs can be seen in the previous section 3.4.2.4.

In this VR simulation, these two BBs are applied to the camera named 'fixturecamera' in Schematic View **(Fig.4-13)**. This indicates that the specific 'fixturecamera' camera is the execution target for the two BBs in order to work jointly to carry out the proposed interaction. Moreover, a 3D Frame called 'camerafocus' is created and placed in this simulation scene. It is only used as a reference point that has a position, orientation and scale and will not be visualized in the final simulation. To appropriately manipulate the target camera, the Target Parameter associated with the 'Camera Obit' BB is specified as the 3D Frame 'camerafocus'. This defines the 3D entity at which the 'fixturecamera' camera requires to target **(Fig.4-14)**.

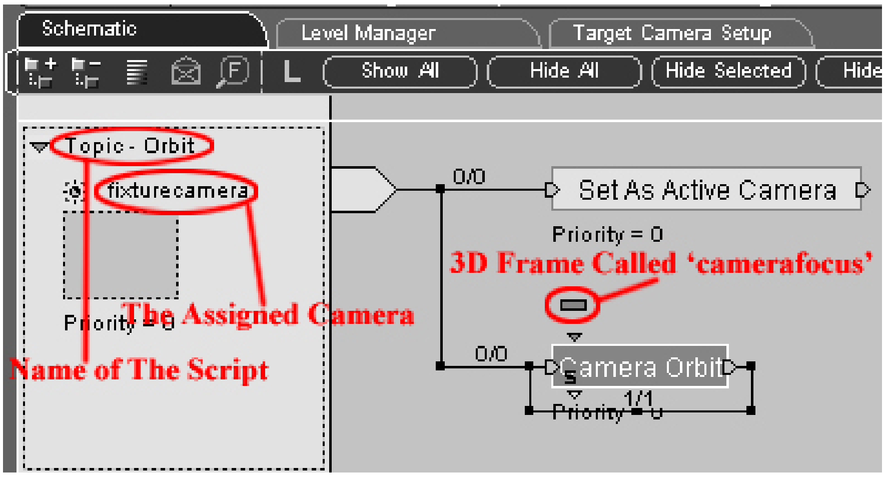
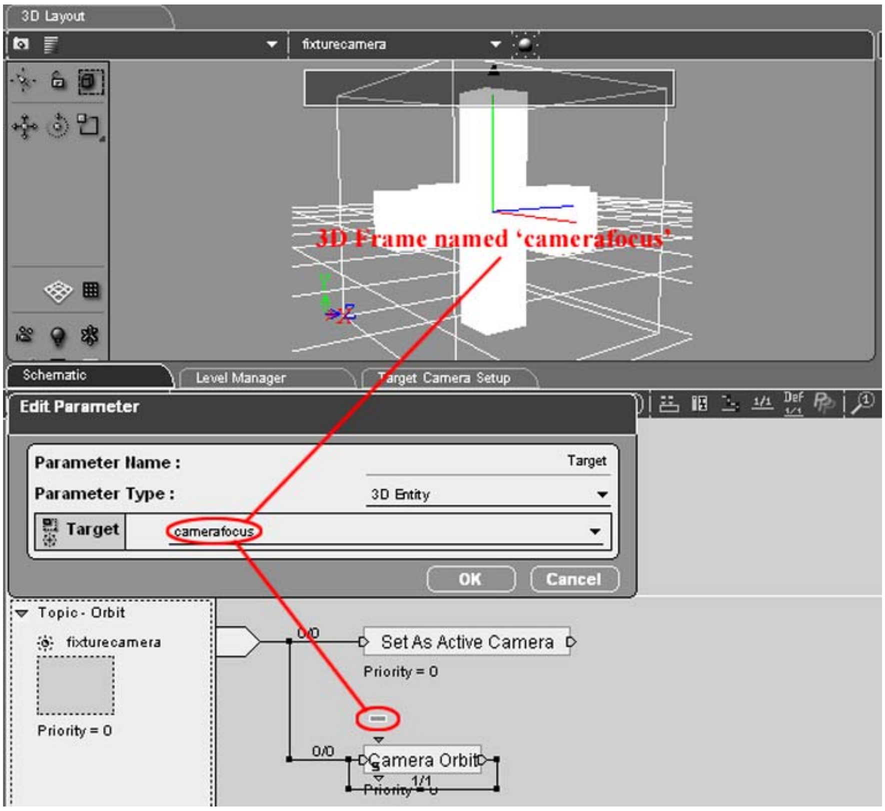


Fig.4-13. The Behaviour Programming Applied to the Camera 'fixturecamera'

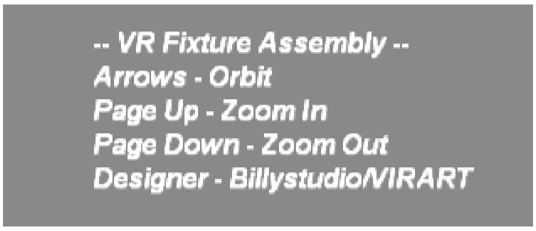


Parameter Name : Target
Parameter Type : 3D Entity
Target camerafocus

Fig.4-14. 3D Frame 'camerafocus' Specified to the Target Parameter

In order to display some guidance information in this VR simulation, three different BBs were applied to the behaviour programming process. The BB named 'Create System Font' is used to create a font in association with a system font name. The BB named 'Set Font Properties' is used to edit font properties and change the appearance of a font. The BB named '2D Text' is used to display text in a 2D Frame. A 2D Frame named 'Text Frame' is created in the scene to facilitate the display of a multi-line context, in which the guidance information **(Fig.4-15)** with regard to the use of the VR simulation is demonstrated in a suitable form. In fact, the target text is contained into this 2D Frame which determines the position of the proposed text in this simulation.

The introductory graphs in relation to these three BBs are presented in the following pictures **(Fig.4-16) (Fig.4-17) (Fig.4-18)**. These three BBs are finally applied to a 2D Frame 'Text Frame' in this simulation. The behaviour programming process for these three BBs with regard to the Behaviour Links, Parameter Links and Parameter Settings is graphically illustrated according to the Script named 'Display Text' **(Fig.4-19)**.



-- VR Fixture Assembly --
Arrows - Orbit
Page Up - Zoom In
Page Down - Zoom Out
Designer - Billystudio/VIRART

Fig.4-15. The Guidance Information in the VR Simulation

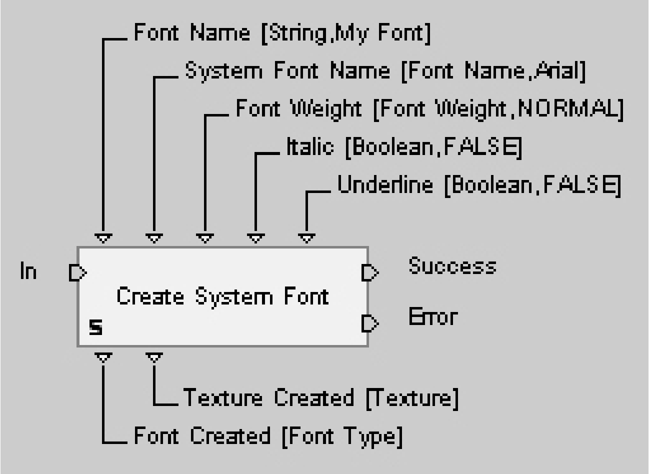

Font Name [String,My Font]
System Font Name [Font Name,Arial]
Font Weight [Font Weight,NORMAL]
Italic [Boolean,FALSE]
Underline [Boolean,FALSE]
In
Create System Font
5
Success
Error
Texture Created [Texture]
Font Created [Font Type]

Fig.4-16. The Introductory Graph of the 'Create System Font' BB

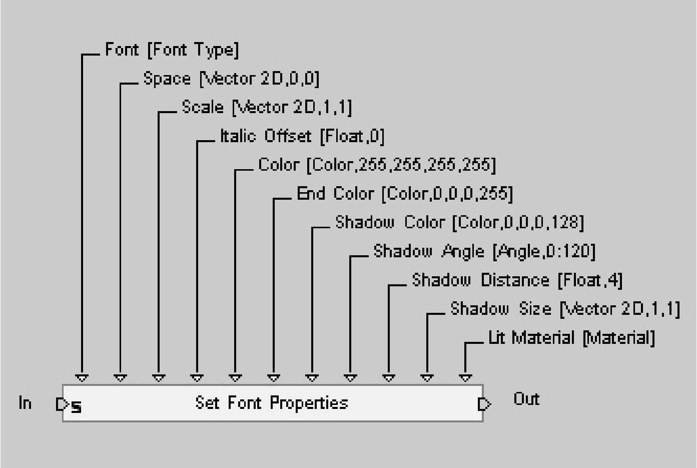

Font [Font Type]
Space [Vector 2D,0,0]
Scale [Vector 2D,1,1]
Italic Offset [Float,0]
Color [Color,255,255,255,255]
End Color [Color,0,0,0,255]
Shadow Color [Color,0,0,0,128]
Shadow Angle [Angle,0:120]
Shadow Distance [Float,4]
Shadow Size [Vector 2D,1,1]
Lit Material [Material]
In
S
Set Font Properties
Out

Fig.4-17. The Introductory Graph of the 'Set Font Properties' BB

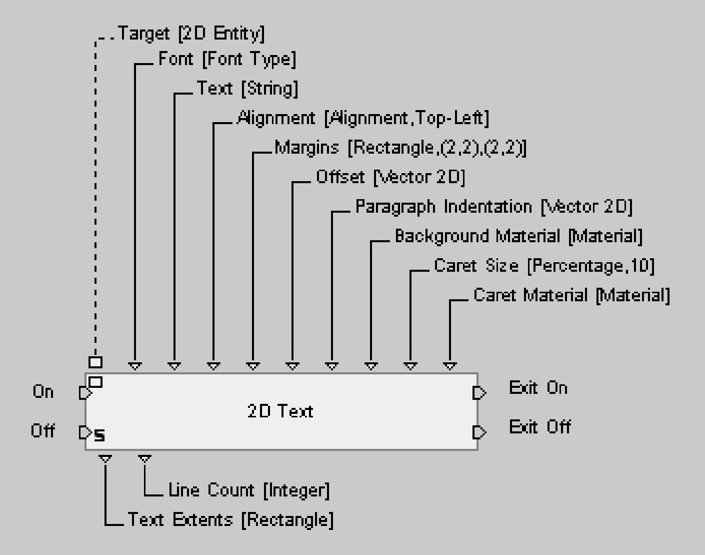

Target [2D Entity]
Font [Font Type]
Text [String]
Alignment [Alignment,Top-Left]
Margins [Rectangle,(2,2),(2,2)]
Offset [Vector 2D]
Paragraph Indentation [Vector 2D]
Background Material [Material]
Caret Size [Percentage,10]
Caret Material [Material]
On
Off
S
2D Text
Exit On
Exit Off
Line Count [Integer]
Text Extents [Rectangle]

Fig.4-18. The Introductory Graph of the '2D Text' BB

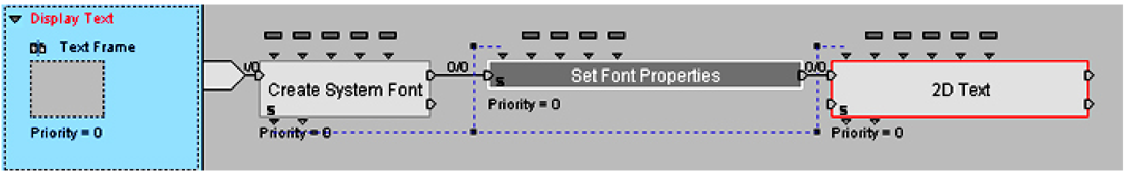


Fig.4-19. The Script 'Display Text' Applied to 2D Frame 'Text Frame'

To improve the quality of real-time 3D graphics in the VR simulation according to the requirement of high-level fidelity, the light mapping and material baking method was used to conduct this simulation study. The fixture base, that is an element of the fixture assembly, has realistic texture and lighting effect in the real-time rendering. The good texturing result of the fixture base model is shown below **(Fig.4-20)**. More details in relation to this method can be found in section 3.4.2.7.

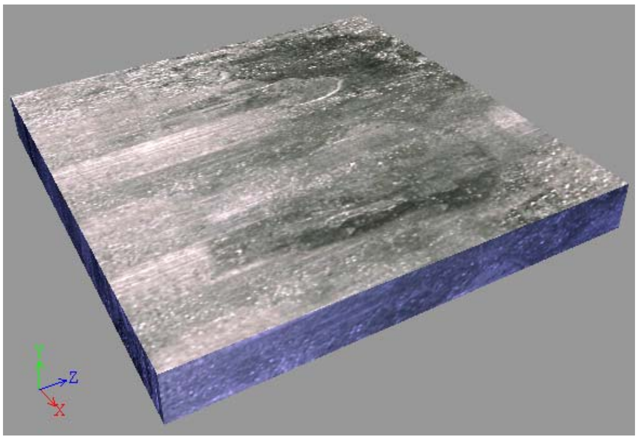


Fig.4-20. The Good Texturing Result of Fixture Base Model in VR Simulation

4.3 Pilot Case Study Two: VR Tool for Fixture Design Learning

4.3.1 Project Summary

In order to reduce the gap between academic education and real industry, it is desirable to use real industrial study cases in the engineering educational syllabus [26]. The turbine blade fixture design, a Rolls-Royce funded project was considered as a typical example for industry use, which was translated into educational purposes.

Pilot case study two assessed whether a VR simulation of fixture design for the turbine blade would be useful to support the second year manufacturing students in their conceptual understanding for the process of setting up a turbine blade workpiece upon a particularly designed fixture. The setting up process of this turbine blade is normally difficult to describe orally by the lecturers in class. The VR simulation aimed to solve this realistic teaching problem arising from the actual fixture design and assembly process. It was considered that the VR tool may also provide a potential platform for students to enhance their learning effectiveness.

The interactive VR simulation for turbine blade **(Fig.4-21)** that is regarded as a useful VR learning tool provides an autonomous, self-learning platform for engineering students, which can be accessible from the school network. It allows each student to perform the self-learning process after classroom time in order to offer a more flexible opportunity to support their education. Within the interactive VR simulation, it also allows the students to interact with the virtual environment and manipulate 3D virtual components in virtual environment according to their preferences. Students virtually select the 3D turbine blade workpiece and self–direct it to the determined location using a mouse device where the workpiece should be placed during the machining period. The target position of the turbine blade is indexed with a rotating star that provides a cue to where to place the workpiece. The prescribed animation about how the workpiece is set up in the fixture assembly is activated as soon as the selected object collides with the rotating star. This animation demonstrates the complex setting up process for locating the turbine blade, which is usually difficult for a lecturer to describe orally. Although a heavy physical model could be transported into classroom, the assembly process of the physical model is very time-consuming and may even cause the hazardous situation in rush occasions during the limited classroom teaching hours.

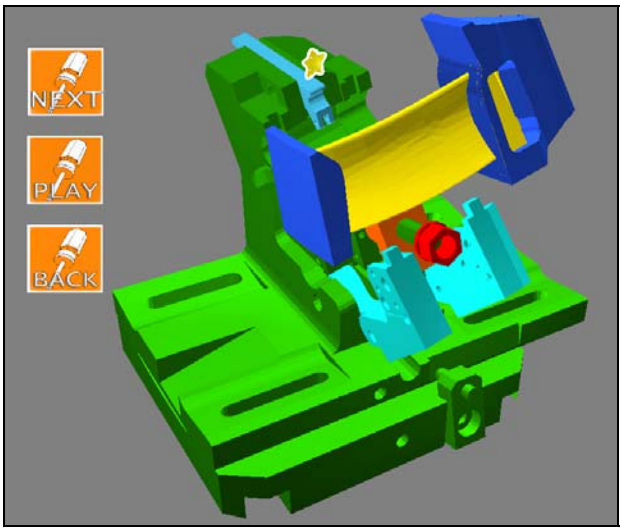
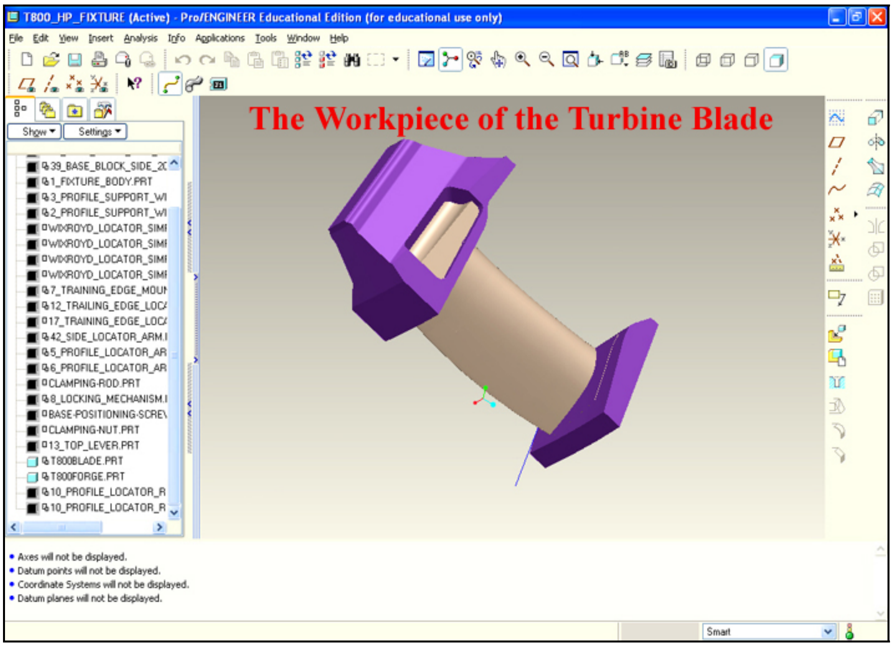


Fig.4-21. The Interactive VR Simulation for the Turbine Blade Fixturing

Alternatively, users may use the straightforward GUI (Graphical User Interface) in which a simple control panel is provided for students to directly activate and control an instructive animation of the turbine blade setting up process. Users are also allowed to change and manipulate the viewpoint of VR simulation by using “Arrows” keys on the keyboard to orbit the scene camera as well as using “PAGEUP&DOWN” to zoom in-out. These functions enable the students to undergo much better visualization in terms of their preferences during the learning process.

4.3.2 Developing the Turbine Blade VR Simulation

The main procedures that were used to complete this interactive VR simulation for the turbine blade fixturing can be found in previous section 3.3.2. In particular, **Figure 3-6** in that section graphically describes how these procedures were carried out. **Fig.4-22** shows the model of the turbine blade.



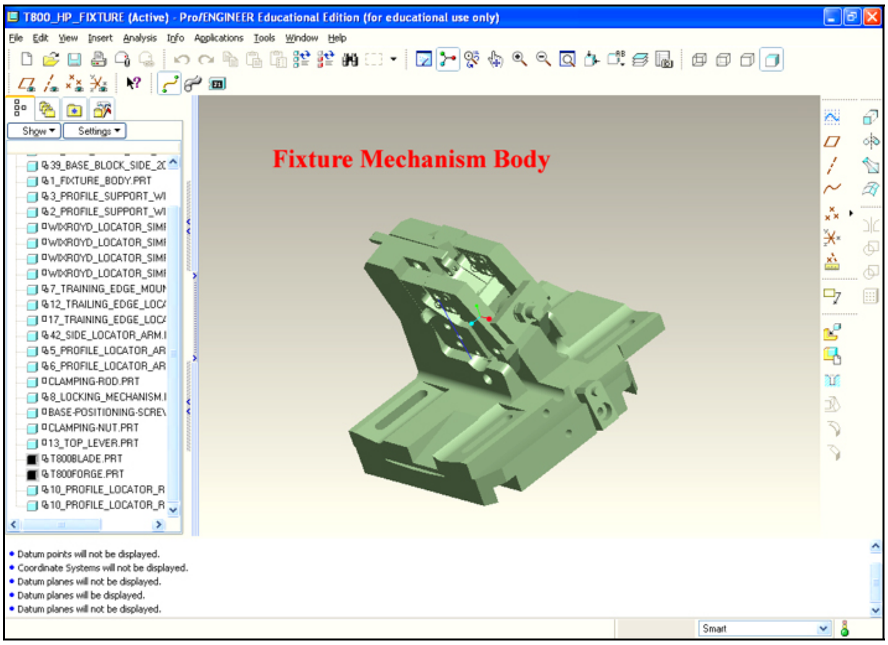
- Axes will not be displayed.
- Datum points will not be displayed.
- Coordinate Systems will not be displayed.
- Datum planes will not be displayed.

Fig.4-22. The Workpiece Named the Turbine Blade

This section provides details related to the integration of both the VE behaviour programming and the creation of prescribed animation. The specific behaviour programming content and the prescribed animation approach with regard to this VR simulation are presented in the following sections.

4.3.2.1 The Creation of the Fixture Assembly Model

The complex model of the fixture assembly **(Fig.4-23)** for the turbine blade workpiece was created at early conceptual design stage using Pro/ENGINEER as described in previous section 3.4.3.



- Datum points will not be displayed.
- Coordinate Systems will not be displayed.
- Datum planes will not be displayed.
- Datum planes will be displayed.
- Datum planes will not be displayed.

Fig.4-23. The Fixture Assembly Model Created in Pro/ENGINEER

4.3.2.2 The Creation of the Prescribed Animation

According to the requirement of this interactive VR simulation, the prescribed animation needed to be played in the simulation once an activation signal is received. This animation demonstrates the process of setting up the proposed turbine blade workpiece upon the particularly designed workholding fixture **(Fig.4-23)**. The animations associated with animated 3D objects were created using 3D Studio Max. In 3D Studio Max, each animated 3D object comprises an independent prescribed animation. The creation process of the prescribed animation for the turbine blade is illustrated in the picture below **(Fig.4-24)**.

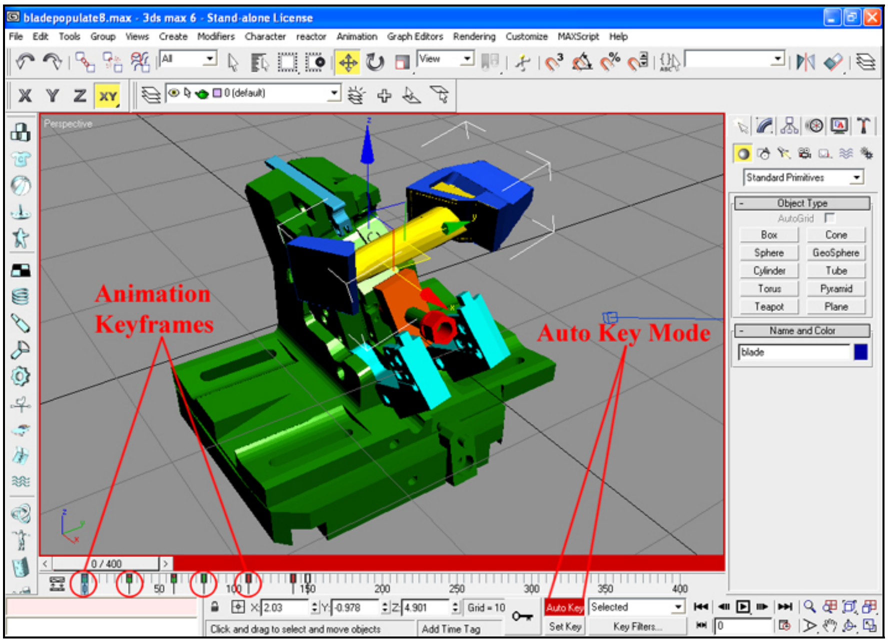


Fig.4-24. The Creation of the Prescribed Animation for the Turbine Blade

After creating the prescribed animation, overall animated objects were then converted into the Virtools platform by using the Virtools 3DS Max Exporter. In Virtools, each animated 3D object represents a fixture element that forms the fixture assembly model. The name of each fixture element is still available in Virtools platform as same as the name used in 3D Studio Max, e.g. ‘blade’, ‘arm1’, ‘arm2’ and so on **(Fig.4-25)**. These fixture elements contain a set of their individual animations which jointly determine the entire turbine blade fixturing sequence, movement, location and trajectory within the VR simulation.

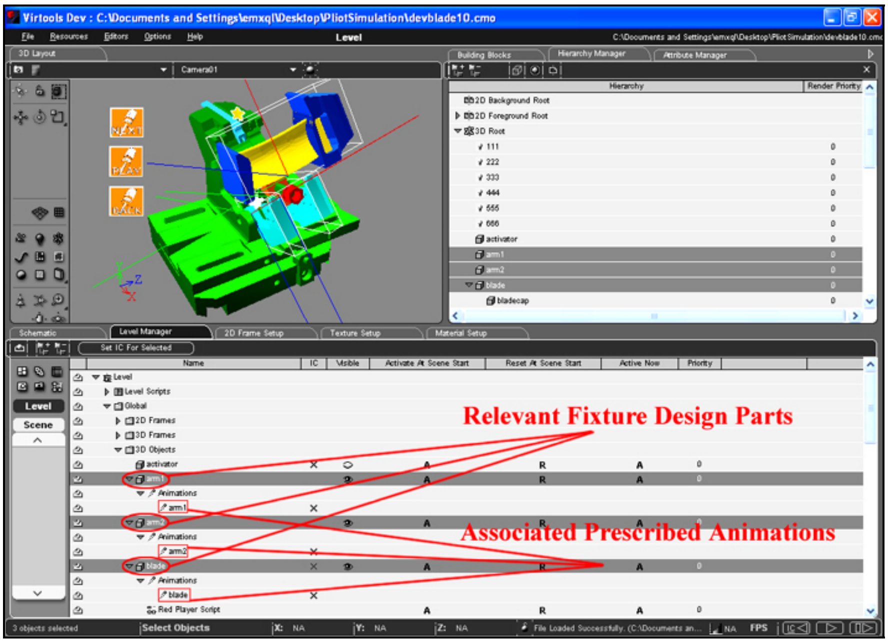


Fig.4-25. Fixture Elements with the Associated Animations in Virtools

4.3.2.3 The Behaviour Programming for the Core Interactive Features

After the conversion process from 3D Studio Max to Virtools, the behaviour programming process was continued to incorporate interaction features into this VR simulation. These interaction features were determined through discussion between the fixture design lecturer and the VR simulation developer.

The core interactive feature of the VR simulation was that the prescribed global animation would be activated as soon as the turbine blade is placed in the intended location and orientation in VE by the users. This global animation is used to demonstrate the process of setting up the turbine blade workpiece upon the particular designed fixture. To achieve this important interaction feature, the specific behaviour programming content is applied to the turbine blade 3D object named 'blade' **(Fig.4-22)** in the VR simulation. The behaviour programming process is explained according to the graph below **(Fig.4-26)**.

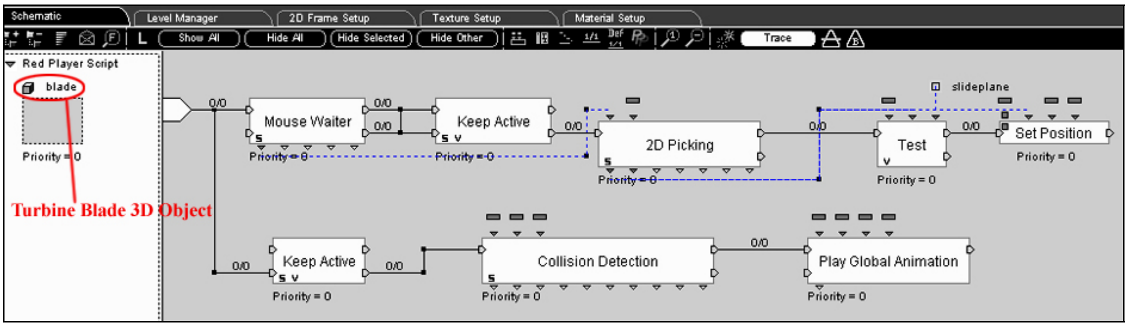


Fig.4-26. The Behaviour Programming Applied to a 3D object called 'blade'

The first row of five BBs is used to fulfil the interaction feature of locating the turbine blade workpiece to the proposed location and orientation according to what the users require. The second row of three BBs is applied to achieve the function of activating and executing the prescribed global animation as soon as the activation signal is received. The activation signal that activates the animation in this case was specified as a collision message. More details will be explained in the following sections.

4.3.2.4 The Core Interaction Feature of Locating the Workpiece

The first row of five BBs are the 'Mouse Waiter' BB, the 'Keep Active' BB, the '2D Picking' BB, the 'Test' BB and the 'Set Position' BB. The BB of 'Mouse Waiter' is used to activate different outputs according to mouse actions. Such as: 'Move', 'Left Button Down' and 'Left Button Up' etc. The BB of 'Keep Active' is used to keep streaming activated even if the input is activated only once. The BB of '2D Picking' is used to return the 3D entity, the normal and the exact position of the 2D mouse picking. The BB of 'Test' is used to generate the appropriate output according to the test between A and B. The BB of 'Set Position' is often used to set the position of a 3D entity. The introductory graphs in relation to these five different BBs are presented in the following pictures to set out more specifications about them **(Fig.4-27) (Fig.4-28) (Fig.4-29) (Fig.4-30) (Fig.4-31)**.

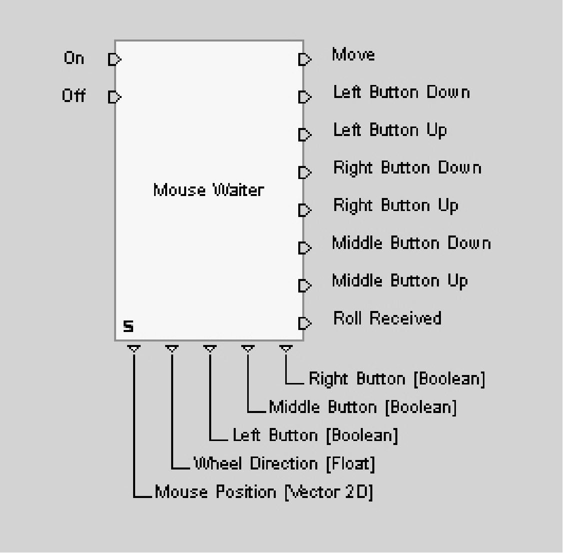


Fig.4-27. The Introductory Graph of the BB Named 'Mouse Waiter'

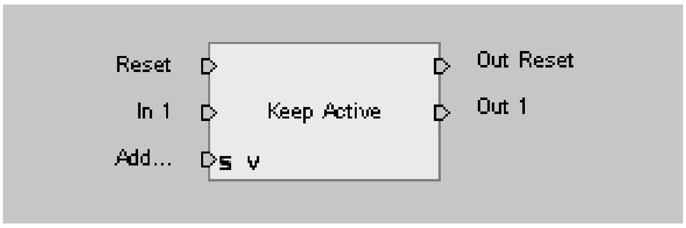


Fig.4-28. The Introductory Graph of the BB Named 'Keep Active'

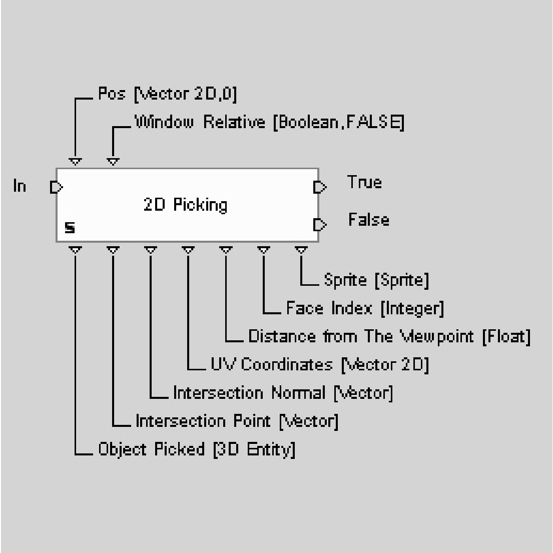

Pos [Vector 2D,0]
Window Relative [Boolean,FALSE]
In
2D Picking
s
True
False
Sprite [Sprite]
Face Index [Integer]
Distance from The Viewpoint [Float]
UV Coordinates [Vector 2D]
Intersection Normal [Vector]
Intersection Point [Vector]
Object Picked [3D Entity]

Fig.4-29. The Introductory Graph of the BB Named '2D Picking'

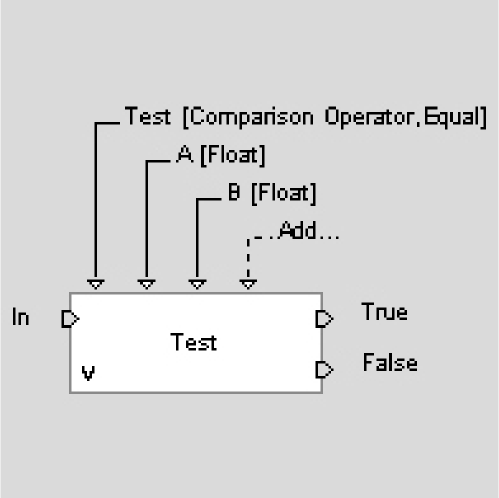


Fig.4-30. The Introductory Graph of the BB Named 'Test'

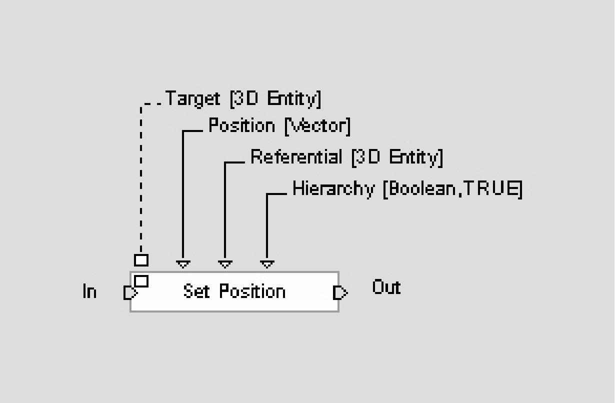
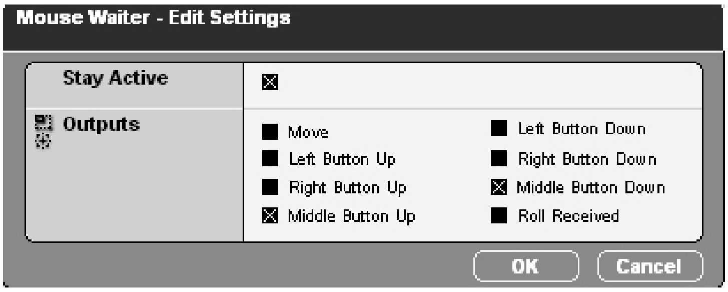


Fig.4-31. The Introductory Graph of the BB Named 'Set Position'

To present the logic and correlations for the behaviour programming amongst these five BBs, the process of attributing and connecting these logical BBs to a workflow will now be explained in detail. During this process, the relevant Parameters are defined as well as input and output ports of these BBs are connected by the Behaviour Links (bLink) or Parameter Links (pLink) so as to implement a data value and control flow within this visual scripting environment.

Initially, these five BBs are obtained from the BBs standard library according to the directory such as: 'Mouse Waiter' (Controllers/Mouse), 'Keep Active' (Logics/Streaming), '2D Picking' (Interface/Screen), 'Test' (Logics/Test), 'Set Position' (3D Transformations/Basic). All these BBs are applied into the Script named 'Red Player Script'.

Secondly, the Behaviour Outpts (bOut) of ‘Mouse Waiter’ BB are specified through Edit Settings. Then configure the Behaviour Outpts in terms of the Picture shown in **Fig.4-32**: the bOuts of both ‘Middle Button Up’ and ‘Middle Button Down’ need to be checked and the option of ‘Stay Active’ also requires to be checked. These two bOuts are respectively activated while the middle mouse button is released or pressed. Furthermore, the Parameters of ‘Test’ BB are specified via Edit Parameters. Then configure the Parameter ‘Test’ to be the operation of ‘Equal’ and the Parameter ‘B’ to be the 3D object named ‘slideplane’ that is an invisible sheet mesh upon which the turbine blade will be defined to move along the surface **(Fig.4-33)**. The Target Parameter of ‘Set Position’ BB is defined via Edit Parameters. Then configure the Parameter called ‘Target (3D Entity)’ to the 3D object ‘blade’ which indicates the turbine blade model **(Fig.4-34)**.



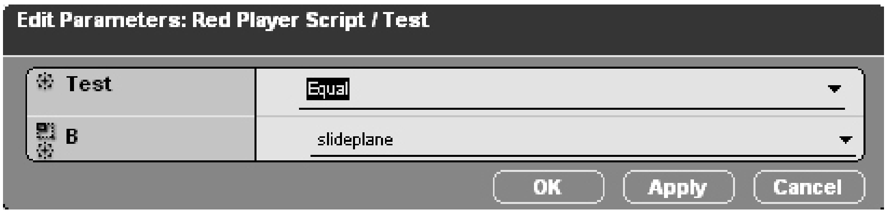
Mouse Waiter - Edit Settings

Stay Active ☒

Outputs

■ Move ■ Left Button Down
■ Left Button Up ■ Right Button Down
■ Right Button Up ☒ Middle Button Down
☒ Middle Button Up ■ Roll Received

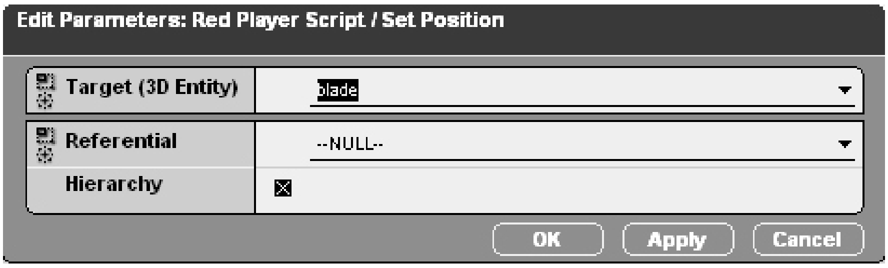
Fig.4-32. The Edit Settings for ‘Mouse Waiter’ BB



Edit Parameters: Red Player Script / Test

Test: Equal
B: slideplane

Fig.4-33. The Edit Parameters for ‘Test’ BB



Edit Parameters: Red Player Script / Set Position

Target (3D Entity): blade
Referential: --NULL--
Hierarchy: ☒

Fig.4-34. The Edit Parameters for ‘Set Position’ BB

Thirdly, all that remains is to add behaviour links between these BBs and parameter links from a pOut on one BB to a pIn on another BB. Working from the left, link the top bIn of ‘Mouse Waiter’ BB to the Script named ‘Red Player Script’. Next respectively link the top bOut of ‘Mouse Waiter’ BB to the second bIn of ‘Keep Active’ BB and link the second bOut of ‘Mouse Waiter’ BB to the top bIn of ‘Keep Active’ BB. Then link the second bOut of ‘Keep Active’ BB to the bIn of ‘2D Picking’ BB. Next link the top bOut of ‘2D Picking’ BB to the bIn of ‘Test’ BB. Thereafter link the top bOut of ‘Test’ BB to the bIn of ‘Set Position’ BB. All behaviour links (bLinks) are completed up to this point. Next link the first left pOut of ‘Mouse Waiter’ BB to the first left Parameter of ‘2D Picking’ BB. Link the first left pOut named ‘Object Picked’ of ‘2D Picking’ BB to the second left pIn named ‘Position’ of ‘Set Position’ BB **(Fig.4-35)**. After this, the process of behaviour programming for the first row of five BBs is achieved.

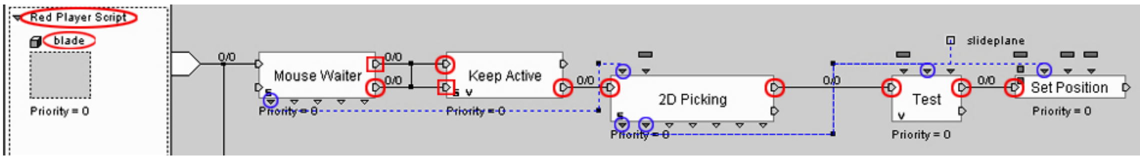


Fig.4-35. The Behaviour Links and the Parameter Links for the Five BBs

As a result, the logic and correlations of the behaviour programming semantics for these five BBs are understood as follows. The ‘Mouse Waiter’ BB is used to define the middle mouse button as the user input method for interaction. Once the middle mouse button is pressed, the activation flow propagates to trigger ‘2D Picking’ BB which is used to determine the selected 3D object in the scene according to the mouse position data. These position data are generated by ‘Mouse Waiter’ BB and passed by the Parameter Link between ‘Mouse Waiter’ BB and ‘2D Picking’ BB. As soon as the 3D object is selected by the mouse, the data value related to the ‘object picked’ will be passed to the ‘Test’ BB and used to execute the comparison operation with the pre-defined 3D object named ‘slideplane’. As long as the mouse position data created by the middle mouse button correspond to the surface data of ‘slideplane’ object, the turbine blade named ‘blade’ in the scene is re-placed to the new location where the user picks by the middle mouse button. This new location of the turbine blade is confirmed by releasing the middle mouse button. Therefore, the interaction feature of locating the turbine blade is fulfilled.

4.3.2.5 The Core Interaction Feature of Performing the Animation

The second row of three BBs is used to complete the interaction function of activating and executing the prescribed global animation as soon as the collision message is received. These three BBs are the BB named 'Keep Active', the BB named 'Collision Detection' and the BB named 'Play Global Animation'. Thus, the BB of 'Keep Active' is used to keep streaming activated even if the input is activated only once. The BB of 'Collision Detection' is often used to detect the collision between the 3D object and other 3D objects that are set as obstacles. The BB of 'Play Global Animation' is used to make a set of 3D entities performing a global animation. The introductory graphs with regard to these three BBs are shown in the following pictures to state more specifications about them **(Fig.4-28) (Fig.4-36) (Fig.4-8)**.

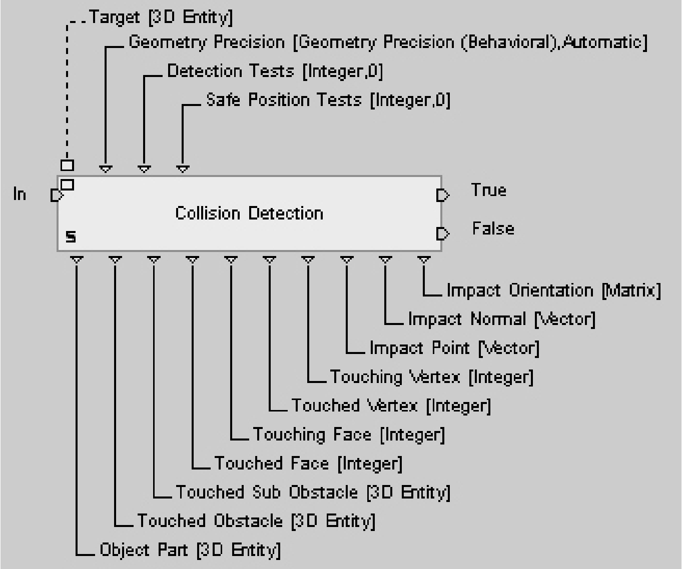

Target [3D Entity]
Geometry Precision [Geometry Precision (Behavioral),Automatic]
Detection Tests [Integer,0]
Safe Position Tests [Integer,0]
In
Collision Detection
S
True
False
Impact Orientation [Matrix]
Impact Normal [Vector]
Impact Point [Vector]
Touching Vertex [Integer]
Touched Vertex [Integer]
Touching Face [Integer]
Touched Face [Integer]
Touched Sub Obstacle [3D Entity]
Touched Obstacle [3D Entity]
Object Part [3D Entity]

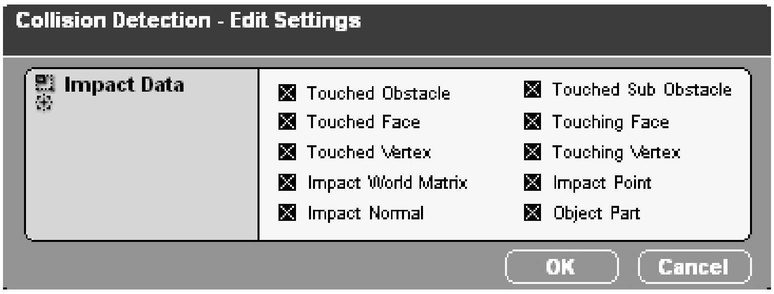
Fig.4-36. The Introductory Graph of the BB Named 'Collision Detection'

To explain the logic and correlations for this behaviour programming, the process of specifying and connecting these logical BBs to a workflow will be demonstrated in detail. Through this process, the relevant Parameters are defined as well as input and output ports of these BBs are connected by the Behaviour Links (bLink) or Parameter Links (pLink) in order to implement a data value and control flow in this visual scripting environment.

Firstly, these three BBs are obtained from the BBs standard library according to the directory such as: 'Keep Active' (Logics/Streaming), 'Collision Detection'

(Collisions/3D Entity), ‘Play Global Animation’ (3D Transformations/Animation) and applied into the Script named ‘Red Player Script’.

Secondly, the Parameter Outputs (pOut) of ‘Collision Detection’ BB are specified through Edit Settings. Then configure the Parameter Outputs in terms of the picture shown in **Fig.4-37**: the overall pOuts in Edit Settings dialogue e.g. ‘Touched Obstacle’, ‘Touched Face’ and ‘Touched Vertex’ etc, require to be checked. Furthermore, the Parameters of ‘Play Global Animation’ BB are specified via Edit Parameters. Then configure the Parameter ‘Animation’ to be the target global animation named ‘BladeAnimation’ and the Parameter ‘Duration’ to be 12 seconds to define the duration of playing this animation. The Parameter ‘Progression Curve’ is defined as a 45° diagonal straight line **(Fig.4-38)**.

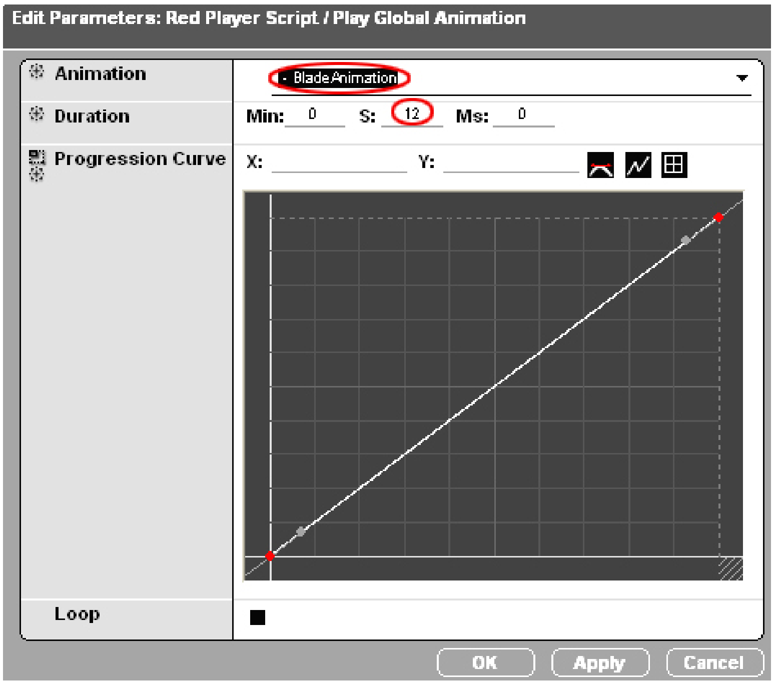


Collision Detection - Edit Settings

Impact Data

☒ Touched Obstacle
☒ Touched Face
☒ Touched Vertex
☒ Impact World Matrix
☒ Impact Normal
☒ Touched Sub Obstacle
☒ Touching Face
☒ Touching Vertex
☒ Impact Point
☒ Object Part

Fig.4-37. The Edit Settings for ‘Collision Detection’ BB



Edit Parameters: Red Player Script / Play Global Animation

Animation: BladeAnimation

Duration: Min: 0 S: 12 Ms: 0

Progression Curve: X: Y:

Loop ■

Fig.4-38. The Edit Parameters for ‘Play Global Animation’ BB

Thirdly, all that remains is to add behaviour links between these BBs. Working from the left, link the second top bIn of ‘Keep Active’ BB to the Script named ‘Red Player Script’. Next link the second top bOut of ‘Keep Active’ BB to the bIn of ‘Collision Detection’ BB. Lastly, link the top bOut of ‘Collision Detection’ BB to the top bIn of ‘Play Global Animation’ BB **(Fig.4-39)**.

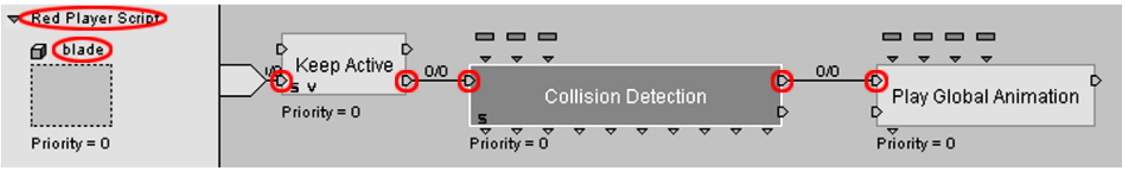


Fig.4-39. The Behaviour Links for the three BBs

In addition, the turbine blade 3D object named ‘blade’ **(Fig.4-40)** is required to add an Attribute value named ‘Moving Obstacle’ **(Fig.4-41)**. This Attribute is used to specify the turbine blade model acting as a moving obstacle in the VR simulation. In the meantime, the top lever, one of fixture elements named ‘activator’ in the scene **(Fig.4-42)** is required to add a Attribute value named ‘Fixed Obstacle’ **(Fig.4-43)** which is used to define this object operating as a fixed obstacle in the simulation. Thereafter, the process of behaviour programming for the second row of the three BBs is achieved.

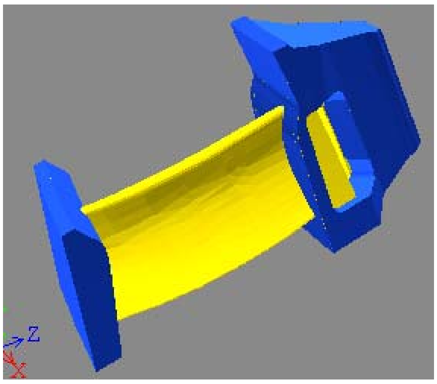
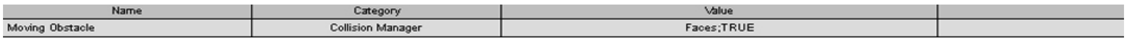


Fig.4-40. The Turbine Blade 3D Object in the Simulation



Name	Category	Value	
Moving Obstacle	Collision Manager	Faces;TRUE	

Fig.4-41. ‘Moving Obstacle’ Attribute Assigned to Turbine Blade

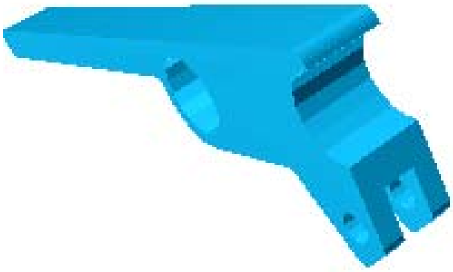
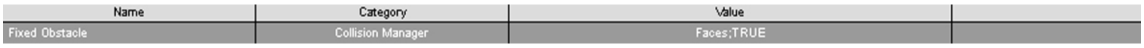


Fig.4-42. The Top Lever 3D Object Named 'activator'



Name	Category	Value	
Fixed Obstacle	Collision Manager	Faces:TRUE	

Fig.4-43. 'Fixed Obstacle' Attribute Assigned to Top Lever

The logic and correlation of this behaviour programming for the three BBs are understood as follows **(Fig.4-39)**. The 'Keep Active' BB is applied to keep streaming activated throughout the run-time of the simulation. This BB will remain transmitting the activation stream to next BB called 'Collision Detection' in order to keep it activated all the time for waiting the collision occurrence. As soon as the turbine blade is manipulated to the intended location and orientation by users, the scheduled collision between the turbine blade **(Fig.4-40)** and the top lever **(Fig.4-42)** models will take place. This collision will be detected by the BB of 'Collision Detection'. Subsequently, the 'Collision Detection' BB propagates the activation flow to the adjacent 'Play Global Animation' BB so that the prescribed animation is activated and played according to the settings provided by 'Play Global Animation' BB.

The interaction features of the VR simulation are established at last. The global animation used to demonstrate the process of the turbine blade fixturing will be activated and played as soon as the turbine blade workpiece is placed to the intended location and orientation in VE by the users.

Apart from these important interaction features that are presented above, several additional interaction features were also developed within this VR simulation. According to these additional features, a number of behaviour programming Scripts are applied during the process. More details with regard to these Scripts can be found in Appendix. A.

4.3.3 User Opinion Survey

Finally, a user opinion survey with regard to the assessment of the interactive VR simulation for turbine blade fixturing was conducted. Positive comment and suggestion included within the 25 questionnaires from engineering students encouraged our motivation to proceed with the interesting research further. 18 students provided positive feedback that they believed a further developed VR teaching simulation would be helpful for them to learn more about fixture design. 5 students had neutral opinions but they look forward to seeing the better VR simulation design in the future. Only 2 students claimed the VR simulation was unhelpful for them. The main questions from the survey questionnaire in relation to this VR simulation are addressed as follows:

1. Have you ever heard about the term of "Virtual Reality" before?
2. Have you ever used or experienced an interactive VR tools? (If yes) please state what type of VR tools you have used.
3. Do you think the desktop-based interactive VR simulation for turbine blade fixturing would help you understand the fixture functions? (If yes) please state what favourable functions you can consider.
4. Do you have any suggestion and comment in mind if we can use VR tools to support your learning with respect to design for manufactures courses e.g. fixture design?

4.4 Summary

After the two pilot case studies were completed, the methodology, difficulties and requirements involved in the development of the VFDAS system were appropriately determined in advance. The advantages of interactive VR technology that can be used to support the fixture design, assembly planning and optimization process, were carefully explored by using these pilot studies.

The methodology that was identified for the development of VFDAS refers to the integration method of CAD and VR systems, the means of creating the fixture assembly models, the usable theory of fixture development, VE behaviour programming, the approach of VE usability design, the technique of conducting the prescribed animations and the method of light mapping and texturing.

Chapter 5. An Overview of the VFDAS System and Implementation

5.1 Introduction

As a result of the two pilot case studies presented in Chapter 4, an appropriate methodology to inform the potential requirements and difficulties of developing the applicable VFDAS system was determined. The methodology included the integration method of CAD and VR systems, the means of developing fixture assembly models, the usable theory of fixture development, VE behaviour programming, the approach of VE usability design, the technique of conducting the prescribed animations, the method of light mapping and texturing and so on.

Furthermore, the two pilot studies also concluded that the VR advantages e.g. realistic visualization, high-level of interactivity, physics property simulation and real-time 3D collision detection in accuracy etc., can be exploited to support the conventional fixture design and assembly process as if the whole process is carried out realistically in the real physics world.

This chapter describes the VFDAS system operation and implementation. Section 5.2 presents a brief introduction about the VFDAS system. Section 5.3 describes the specific functionality and system architecture of VFDAS. Section 5.4 presents the application principles of the VFDAS system. Section 5.5 explains the role of the VFDAS system in real industry. Finally, section 5.6 explains how the development requirements relating to the VFDAS system were identified.

5.2 The Brief Introduction of the VFDAS System

The primary objectives and aims were to develop an interactive VR system named VFDAS, which is considered as the major research deliverable in this thesis. The VFDAS system provides an interactive design platform in which the fixture design and assembly process for modular fixtures can be fulfilled by the fixture designers in a VE. The comprehensive VFDAS system is employed to carry out the functionality of fixture element selection, fixture layout design, assembly planning and essential analysis etc. In VFDAS, the combination of many micro-focused research activities into a complete fixture design system was explored and accomplished. More specifically, the VFDAS system allows the users to select the fixture elements form VR libraries, spatially manipulate each element to assemble on the arbitrary workpiece users provide, detect assembly interference and perform design verification.

Even though ordinary CAD and CAFD systems allow configuration evaluation by letting engineers visualize the final fixture assembly, these systems can not support the visualization or planning of the fixture assembly process [44]. In contrast, the VFDAS system provides the function for fixture designers to visualize, observe, support and evaluate the fixture planning and assembly process. The VFDAS system is also considered as a particular type of CAFD or CAMFD system in which the advantages of VR are used to improve modular fixture designs within a VR environment. In addition, two types of fixture designers are regarded as the users of the VFDAS system. One type is the novice fixture designers such as: engineering students and academic technicians. Another type is the professional fixture designers from the industry.

According to a great deal of literature, the implementation of accurate collision detection is a critical aspect of VR simulation system during a virtual manufacturing process [3]. For the development of the VFDAS system, a VR behaviour programming methodology is used to develop the physics behaviour simulation and the real-time 3D collision detection in accuracy. This programming methodology improves the research bottleneck of collision detection related to concave 3D objects in VEs. The framework of the VFDAS system was developed using this methodology.

Moreover, the VFDAS not only provides real-time 3D collision detection in accuracy but also achieves the goal of good physics behaviour simulation. The VFDAS system also demonstrates an interactive design platform that is based on a physically realistic VR environment. The physical properties associated with each fixture element that were developed in the VFDAS system refer to mass, gravity, friction, elasticity, collision detection, applied force, toppling and so on [85]. These physical properties are normally taken into account during the actual fixture design and evaluation process. These physical properties obey Newton's laws of physics.

5.3 Functionality and System Architecture

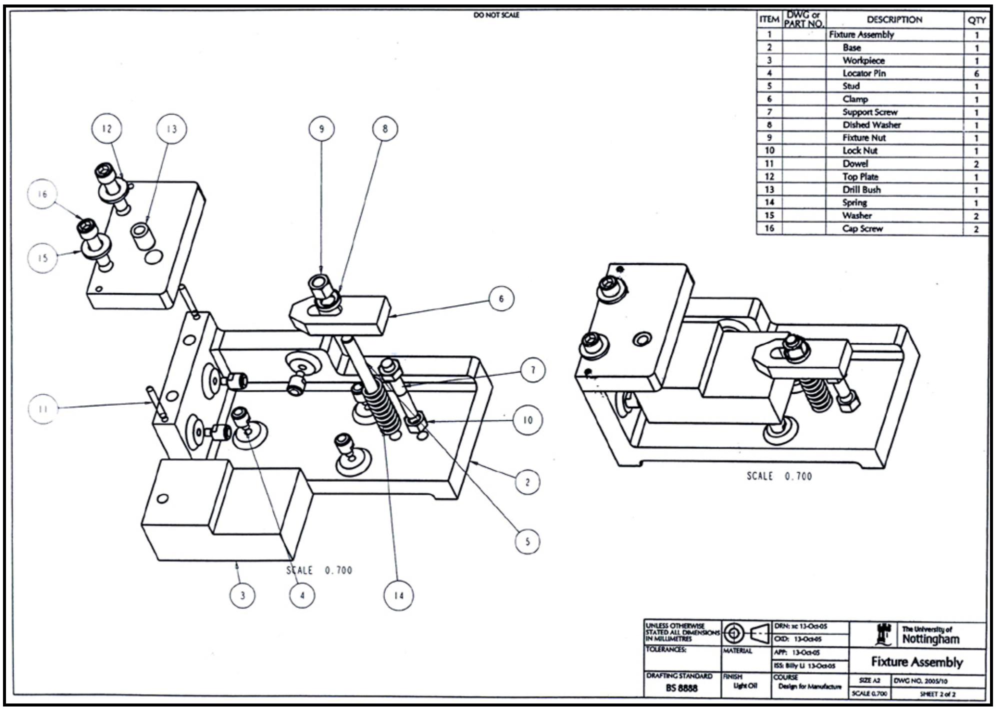
At the present stage of VFDAS system development, 14 main functional modules affiliated with the VFDAS that form the system framework were established. The VFDAS system was incrementally developed to provide a series of functionality and interactivity according to the fundamental needs of fixture development. These generally comprise fixture element selection, fixture planning, assembly evaluation, and essential analysis related to design verification etc. The VFDAS system modules that have currently been achieved are summarized as follows:

1. Pre-defined VR library of fixture elements module
2. Physicalization module
3. Interactive design module
4. Design modification module
5. Accurate 3D real-time collision detection module
6. Group assembly module
7. Design viewpoints control module
8. Set initial location module
9. Interface menu manager module
10. Carousel layout VR library module
11. Assembly process planning and evaluation module
12. Save and restore assembly module
13. Machining interference check module
14. Design configuration output module

In order to systematically present the interactive VFDAS system, the specific functionality and interaction activities associated with each functional module are explained in detail in the following sections.

5.3.1 The Demonstration Sample for the VFDAS

At the preliminary VFDAS development stage, a demonstration example of a simple fixture assembly **(Fig.5-1) (Fig.5-2) (Fig.5-3)** is presented. The example was selected and used to implement a fixture design and assembly process within VFDAS system in order to effectively demonstrate how the VFDAS system works in an interactive physics VR environment. The main operational process related to the VFDAS system is also demonstrated using this simple example, in which the suitable models of modular fixture elements are selected and placed around a prismatic workpiece to form a final fixture assembly.



ITEM	DWG or PART NO.	DESCRIPTION	QTY
1		Fixture Assembly	1
2		Base	1
3		Workpiece	1
4		Locator Pin	6
5		Stud	1
6		Clamp	1
7		Support Screw	1
8		Dished Washer	1
9		Fixture Nut	1
10		Lock Nut	1
11		Dowel	2
12		Top Plate	1
13		Drill Bush	1
14		Spring	1
15		Washer	2
16		Cap Screw	2

UNLESS OTHERWISE STATED ALL DIMENSIONS IN MILLIMETRES

DRN: xc 13-Oct-05

CKD: 13-Oct-05

The University of Nottingham

TOLERANCES:

MATERIAL

APP: 13-Oct-05

ISS: Billy Li 13-Oct-05

Fixture Assembly

DRAFTING STANDARD BS 8888

FINISH Light Oil

COURSE Design for Manufacture

SIZE A2

DWG NO. 2005/10

SCALE 0.700

SHEET 2 of 2

Fig.5-1. The Demonstration Example of a Simple Fixture Assembly

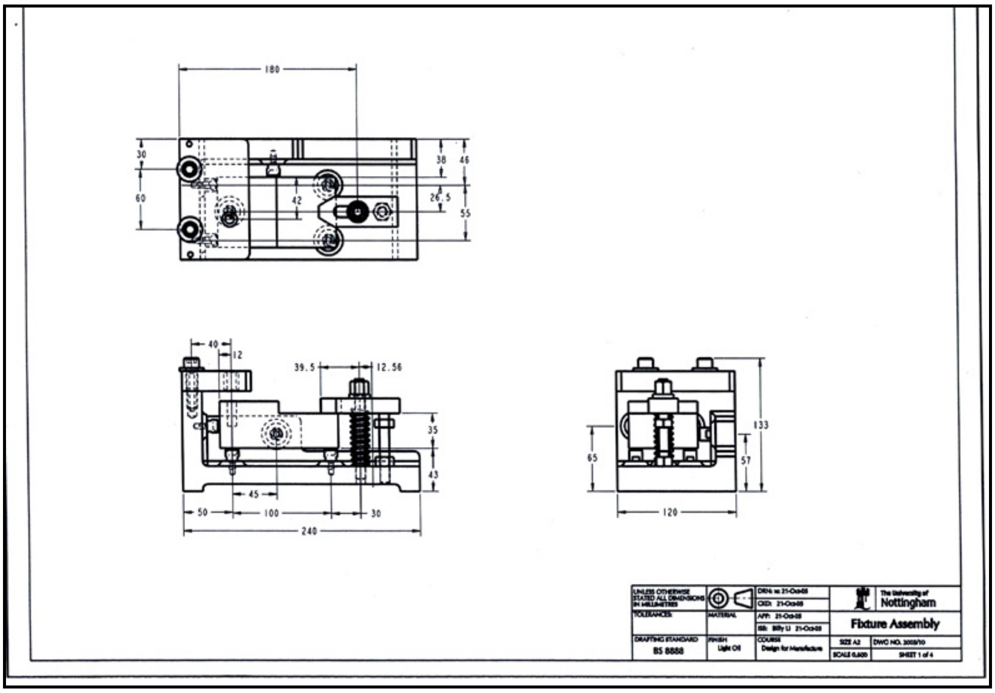


Fig.5-2. The Mechanical Drawings with Detailed Dimensions

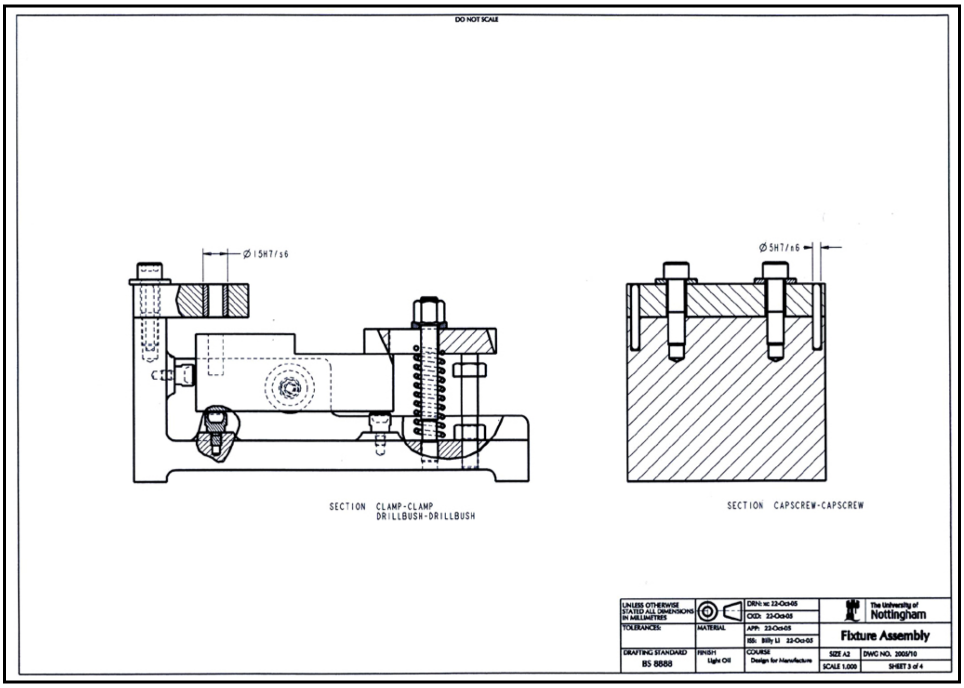


Fig.5-3. The Cross Section Drawings of the Fixture Assembly

This fixture assembly was used to hold the prismatic workpiece **(Fig.3-29)** in position to carry out a machining operation via which the workpiece is drilled a 25mm depth, 4.5mm radius, vertical hole on the top of the surface. The example of this simple fixture assembly was extracted from the year two undergraduate curriculum named 'Design for

Manufacturing' module. Based on this fixture design example, the evaluation process of the VFDAS system conducted by engineering students would be facilitated. The engineering students are regarded as the junior users of the VFDAS. This simple example from the taught course is easily grasped and accepted by the engineering students so that they pay more attention on the aspect of system usability and evaluation rather than the example of fixture design itself.

In addition, the fixture design for this example that is used for the VFDAS implementation was constructed in accordance with the 3-2-1 principle. Using this 3-2-1 principle, both locating and clamping surfaces/points upon the prismatic workpiece are determined. More details with regard to the 3-2-1 principle were described in previous section 2.2.2.

Most of the element models in this fixture design were created using Pro/ENGINEER such as: prismatic workpiece, fixture base, spring and so on. Other element models in the fixture assembly were directly obtained from the standard modular fixtures provided by the commercial tool company such as: locator pin, stud, clamp etc. Finally, all these fixture elements are used to generate the final fixture assembly model using Pro/ENGINEER.

5.3.2 Pre-defined VR Library of Fixture Elements Module

To satisfy the requirement of the prompt fixture element selection in VFDAS, an interactive VR library module is developed in the system. In this VR library, all fixture elements that are contained in the simple fixture assembly presented in last section, are applied in the well-organized fixture element database to provide the decision making support for constructing of fixture configuration. The VR library module comprises 15 pre-defined fixture elements **(Fig.5-1)** at present, which are directly derived from the simple fixture assembly. More existing element models provided by the commercial company will be gathered and included into the VR library module in order to gradually create a systematic modular fixture database which is equipped within the VR library of VFDAS. This database is used to provide the fixture designers with more alternatives to simplify the process of fixture element selection.

Moreover, a convention for specifying the original orientation and axis related to each fixture element from the VR library has been established. A series of samples of the fixture elements from the interactive VR library module are shown in **Fig.5-4**.

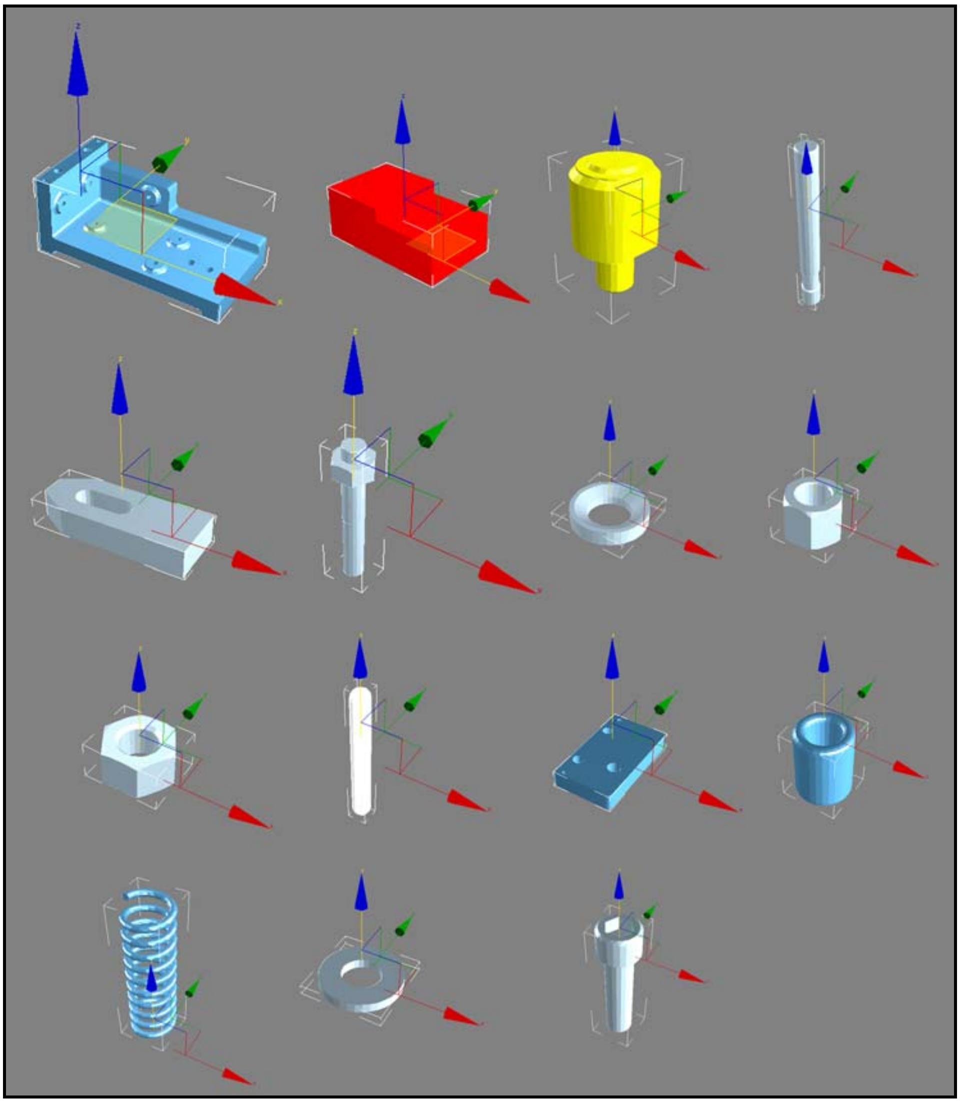


Fig.5-4. The Samples of Fixture Elements from the VR Library Module

Besides, a particular feature of the pre-defined VR library module is that each fixture element is associated with physical properties as soon as they are selected and loaded to the virtual design environment of VFDAS. These physical properties require to be taken into account during the actual fixture planning and assembly evaluation. These physical properties associated with fixture elements refer to their own mass, gravity, friction, elasticity, 3D real-time collision detection, applied force and so on, which realistically demonstrates the physical fixture behaviour in use as if in the real physics world. This feature of the VR library is quite special in comparison with other parametric fixture databases and pre-developed CAD libraries established in various CAFD systems which are reported in the existing research literature [7] [11] [13] [18].

Access to the pre-defined VR library for the input of the fixture elements is through the pop-up menus on the graphical user interface (GUI) of the VFDAS system. These menus not only demonstrate essential information related to the fixture elements, but also combine a 3D graphical representation of the fixture element from the needed viewpoints. Therefore, the essential information related to the fixture elements is often their geometric dimensions and mechanical characteristics etc.

The predefined models of fixture elements saved in *.NMO format are appropriately classified and stored on computer hard disk in the different folders according to the different fixture element types. These comprise the final fixture element database in VFDAS. As a result, six major types of fixtures in the final fixture element database are fixture bases, locators, clamps, supporters, adaptors and connecting accessories, For example: the folder named 'locators' included in the fixture database contains all locating elements such as: the locator pin. The selected fixturing elements associated with physical properties are retrieved and imported from the systematic fixture database into the VFDAS system once the corresponding buttons on the menus are pressed by users. The detailed VR behaviour programming in relation to the VR library module construction will be presented in the section 6.3.1.

To develop a more comprehensive VR library for fixture elements in the future that will be used to efficiently maintain the fixture related information and data, several suitable literature is worthwhile being addressed with regard to the research aspect of fixture feature recognition and classification. This research aspect is the important factor for constructing a successful fixture element database.

A CAFD system for comprehensive modular fixtures was reported by Hou and Trappey in 2001 [6], which is made up of three key modules, i.e. Fixture Data Management, Fixture Element Selection and Fixture Layout Design. To logically maintain fixture information, a comprehensive fixture database associated within the Fixture Data Management module is established, in which essential features relating to each type of fixtures should be represented. The essential features from the commercial fixture catalogues such as: geometric dimensions and mechanical features are firstly classified into categories. In addition, fixture elements can also be grouped by their functions, sizes and shapes so that the parametric database is created. Fixture related data can be easily accessed and browsed after scientific grouping and classification. In this system, the fixture elements are initially grouped by their mechanism i.e. mechanical or hydraulic. Thereafter, in each type of fixture, the elements are classified according to their functions e.g. locators, supports and clamps. Lastly, the dimension and shape are considered for classification [6]. As result, the classification structure for the fixture elements is illustrated in **Fig.5-5**.

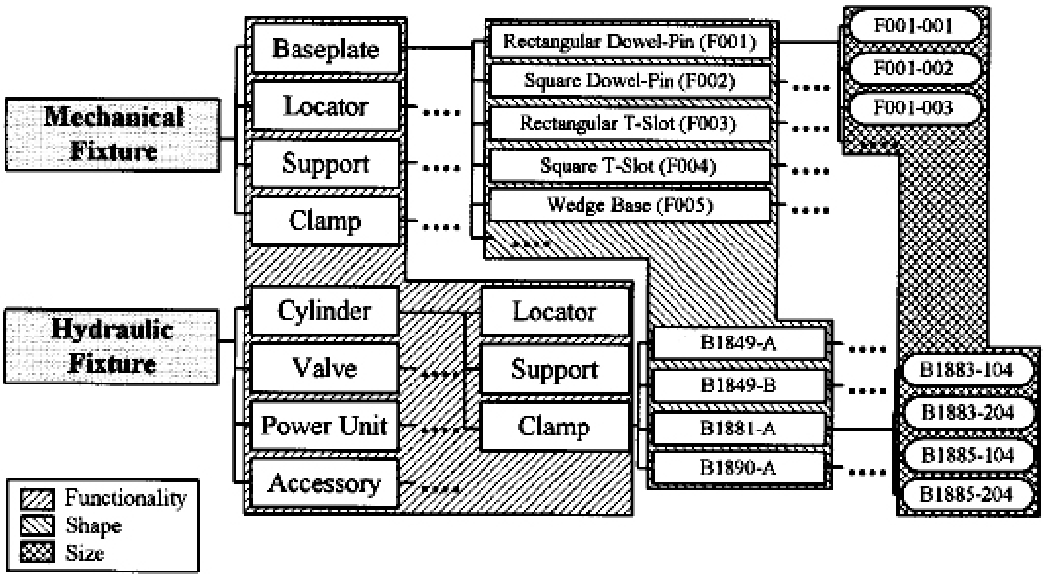


Fig.5-5. The Classification Structure of Fixture Elements

Dai et al [11] developed a method in 1997 to create the modular fixture element database as well as construct the fixturing tower database (i.e. subassemblies). Using this method, the modular fixture elements are divided into four categories **(Fig.5-6)** according to their functions: (a) baseplate elements, where overall locating, supporting, clamping elements and the workpiece are fastened. (b) locating and supporting elements, used for locating and supporting the workpiece. (c) clamping elements, used for clamping the workpiece to constrain its movement during the machining operation. (d) accessory elements, the supplementary elements providing the connection between the baseplate and the locators, supports and clamps to reach at the required height of the fixture tower. The main goal of classifying the fixture elements is to establish the element database for the rapid retrieval and assembly of fixtures [11].

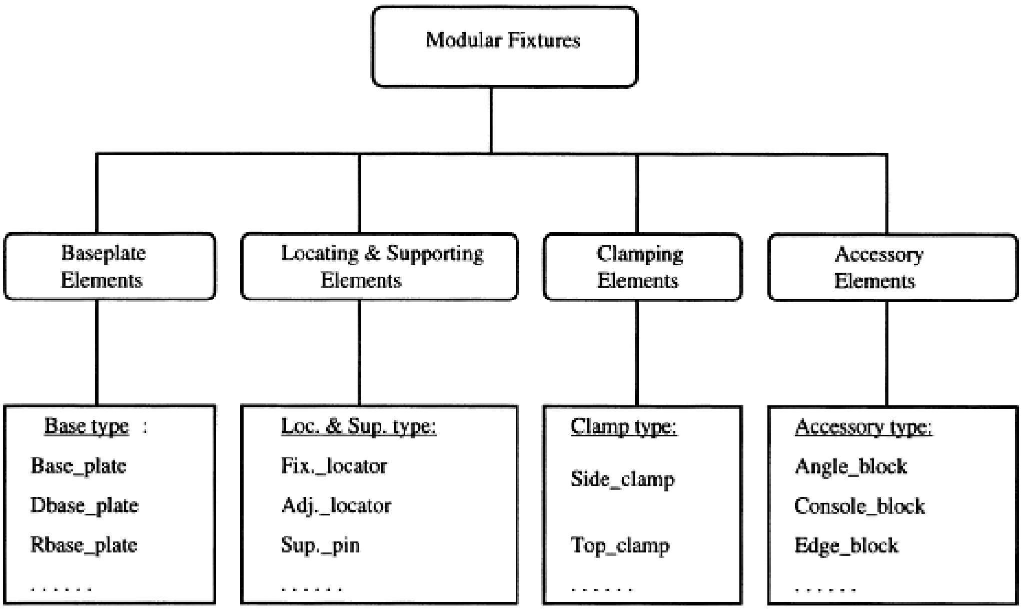


Fig.5-6. The Classification of Modular Fixturing Elements

5.3.3 Physicalization Module

Unlike other existing CAFD or CAMFD (Computer Aided Modular Fixture Design) systems available from similar research [5] [6] [7] [18], an important Physicalization Module was developed within the VFDAS system. Using this Module, the substantial VR advantage of simulating the physical properties and like-life behaviours with interactivity is integrated into the system architecture of VFDAS. The main functions of this Physicalization Module are to monitor and manage the essential physical properties and behaviours associated with each fixture element within the interactive design environment of VFDAS. These elements are selected and input from the pre-defined VR library module to form the final fixture design.

Four operational utilities are comprised in the Physicalization Module to efficiently manage and control the physics properties associated with every fixture element. In general, the four operational utilities are the ‘Unmovable Utility’, ‘Movable Utility’, ‘Accurate Manipulation Utility’ and ‘Unphysicalized Utility’. Accordingly, they are associated with the four corresponding keys on the computer keyboard, which respectively are key 7, key 8, key 9 and key 0.

In addition, each fixture element is assigned with the physical properties as soon as they are loaded from the VR library into the VFDAS system. These input fixture elements can be selected and manipulated by fixture designers using the keyboard in order to place them in the proposed location and orientation according to the requirement of the fixture design.

Once the fixture designers interactively locate the selected fixture element in the position and orientation as the final decision, the Unmovable Utility is applied to fasten and immobilize the selected fixture element. This fixture element will still have the function of accurate collision detection although it can no longer be spatially moved after using the Unmovable Utility. Movable Utility is used to re-assign the physical properties to the selected fixture elements e.g. their own mass, gravity, friction, elasticity, collision detection, applied force, reaction force etc., so that the immobilized fixture elements can be manipulated again according to the operational need of users. Unphysicalized Utility is employed to unphysicalize the selected fixture elements so that they no longer have any physical property Thereafter, they are able to spatially penetrate and intersect with other 3D physical objects in VFDAS without the functions of collision detection and physics behaviour simulation. Finally, the Accurate Manipulation Utility is smartly designed to tackle the difficulty of interactively manipulating the physicalized 3D objects in accuracy. Accurate Manipulation Utility is also used to carry out the accurate fixture assembly process. Using this Utility, the selected fixture elements are accurately manipulated and placed to the position and orientation fixture designers propose although they are associated with a series of physical properties and interactive behaviours. The specific VR behaviour programming relating to the Physicalization Module and four constituent utilities will be explained in section 6.3.3.

5.3.4 Interactive Design Module

The main role of the Interactive Design Module developed in the VFDAS system aims to provide fixture designers with the functionality by which they can interactively manipulate and position the selected fixture element one by one around the target workpiece according to the location and orientation proposed by them.

The operation of VFDAS system normally begins by loading the 3D solid model of the target workpiece which is created using a CAD toolkit such as: Pro/ENGINEER. As soon as the workpiece model is loaded into the system, a suitable baseplate needs to be selected and retrieved from the VR library. Using Interactive Design Module, this baseplate is manipulated and placed to the appropriate location and orientation. Thereafter, other relevant fixture elements such as: the locating, supporting and clamping elements are then selected and input from the systematic fixture database affiliated to the pre-defined VR library module. These elements will be virtually assembled one by one to make contact with the target workpiece on a specified surface and at a specific location in order to form the complete fixture configuration. In addition, the 3-2-1 locating and clamping principles are integrated to this Interactive Design Module to determine the correct fixture layout.

The interactive design process in VFDAS is achieved in the physics simulation world. To support the interactive design process, using the Interactive Design Module sometimes needs to jointly cooperate with the functions of the Physicalization Module so that each fixture element associated with physical properties is assembled one by one around a workpiece. Using the Physicalization Module attempts to promptly assign or remove the physical properties associated with the selected fixture elements whenever the fixture designers need to do so.

To achieve the interactive manipulation in the Interactive Design Module, a group of keys are specified to realize the functions of 3D object translation and 3D object rotation in the VFDAS, e.g. moving the selected fixture element forward/backward, left/right, up/down and respectively rotating it by its local axis X, Y, Z. The keys used for these functions are specified as the keys of W, A, S, D, Z, X as well as the keys of 1, 2, 3, 4, 5, 6 on the computer keyboard. More specifically, the key W is used to move the selected fixture element model forward according to the world coordinate system of the VE. The key S is used to move the element model backward. The key A is used to move the element towards the left direction. The key D is used to move the element toward the right direction. The key Z is used to move up the fixture element along the Y axis of the world coordinate. The key X is used to move down the element along the Y axis of the world coordinate. The function of element selection is simply achieved by clicking the left mouse button on the target element.

As a result, the appropriate location and orientation of each fixture element are determined through using these functions of interactive manipulation to form the final fixture configuration. The selected fixture elements are fastened by using the Unmovable Utility associated within the Physicalization Module once fixture designers confirm their final decision with regard to the location and orientation of assembling the element. The fastened fixture elements will still have the function of accurate collision detection which is ready for the assembly interference detection while other fixture elements are being mounted on the baseplate.

5.3.5 Design Modification Module

The primary function of the Design Modification Module in VFDAS is to provide the fixture designers a flexible modification opportunity. Using this Module, the generated fixture configuration can be further changed and improved if the design result created by the Interactive Design Module does not meet the design requirement the fixture designers have proposed. In addition, this useful Design Modification Module is made up of several operational utilities. They are employed to fulfil a set of modification operations that the fixture designers require to carry out during the fixture planning and evaluation process. The three operational utilities are respectively 'Add Utility', 'Delete Utility' and 'Replace Utility'.

The Add Utility is used to produce a fixture element or fixture tower set towards a new determined fixturing point on the surface of the target workpiece. The Add Utility is also used to duplicate some fixture elements that will be automatically associated with physical properties if necessary. The computer key for this operation is specified as the key C on the keyboard. The Delete Utility aims to provide the function of removing a fixture element or tower from the design environment of VFDAS. The computer key for the operation is specified as the key Del on the keyboard. In addition, the Replace Utility is applied to replace the current fixture element used in the existing fixture configuration by another candidate fixturing solution users have proposed.

5.3.6 Accurate 3D Real-time Collision Detection Module

The realization of accurate collision detection is often a critical aspect of VR simulation system during any virtual manufacturing process [3]. Nevertheless, the importance of the role that the real-time collision detection acts as in physically-based modelling is focused on VR applications where the motion of an object is restricted by collision with other objects or dynamic constraints [86]. Although the great progress has been made in the development of efficient, accurate collision algorithms for convex objects, the slow progress is reported in producing collision detection algorithms for nonconvex objects [87]. In other words, the difficulty in developing the real-time 3D collision detection in accuracy for concave objects still remains unsolved.

In order to narrow this application gap, the Accurate 3D Real-time Collision Detection Module established in the VFDAS system is efficiently applied to provide the real-time 3D collision detection in accuracy for each fixture element or tower involved for the fixture development. These fixture elements include both 3D convex and concave objects. The function of collision detection in VFDAS is also used to avoid the potential design interference between fixturing elements and the workpiece to ensure a collision-free fixture design.

The accurate real-time 3D collision detection in VFDAS is closely associated with the physical property simulation and considered as the important function towards the development of the VFDAS. This function enhances the realism of the VEs and supports the fixture planning and assembly evaluation. Thus, the behaviour programming process related to the construction of this Accurate 3D Real-time Collision Detection Module will be explained in section 6.5.1.

5.3.7 Group Assembly Module

A fixture tower is considered as a combination of fixture elements to provide the connection between the baseplate and a workpiece. In addition, the fixture tower is also considered as the subassembly of a modular fixture, which comprises the locating, supporting and clamping towers [11]. The number of fixture elements in a tower is one or more elements. However, there should be only one element in the tower served as a locator, supporter or clamp. The primary goal of the modular fixture assembly is to select the locating, supporting, clamping and accessory elements that are used to create the fixture towers in order to fasten the workpiece to the baseplate [11].

To fulfil the function of assembling a fixture tower that includes a group of fixture elements, the Group Assembly Module was developed and incorporated into the VFDAS. This Module is employed to collectively manipulate a set of fixture elements or a fixture tower to the determined fixturing location so as to generate the fixture configuration. To facilitate the interactive manipulation for fixture towers by using this Group Assembly Module, a set of keys on the keyboard are specified to realize the translation of a set of fixture elements and grouping a set of fixture elements. Using this Module also defines the members of the selected group in the VFDAS.

The keys for the function of grouping a set of fixture elements are associated with the key Shift on the keyboard and the Left Mouse Button. They work jointly to implement the function of grouping in this Module. The keys for translating a set of fixture elements that have been included into one group are specified as the keys of T, F, G, H, V, B. In particular, the specified Key T is used to move the grouped fixture tower model forward according to the world coordinate system. The Key G is used to move the grouped fixture tower backward. The Key F is used to move the grouped fixture tower towards the left direction. The Key H is used to move the grouped fixture tower towards the right direction. The Key V is used to move up the grouped fixture tower along the Y axis of the world coordinate. The Key B is used to move down the grouped fixture tower along the Y axis of the world coordinate.

In addition, the Group Assembly Module also allows the fixture designers to generate their customised fixture tower structures. These tower structures are saved into the fixture element database associated with the VR library module so that they can be retrieved to facilitate the similar fixture design cases in the future.

5.3.8 Design Viewpoints Control Module

VR has advantages of high-quality 3D visualisation and spatial perception. These advantages are used to improve the capability of VFDAS and support the fixture design process. The VFDAS system aims to integrate these advantages into the final system framework. Therefore, the Design Viewpoints Control Module was developed to provide fixture designers an effective assistance of 3D graphical representation for the fixture development. This Module allows fixture designers to select different perspectives, viewpoints and distances during the fixture design and assembly process.

The Design Viewpoints Control Module in VFDAS system is divided into two fundamental sub-modules which are the Camera Orbit Unit and the Viewpoint Change Unit. The Camera Orbit Unit is employed to flexibly change the design perspectives and distances according to the visual requirement of users. Using this Camera Orbit Unit, the fixture design and assembly process is facilitated throughout the interactive design process of VFDAS. In addition, the Viewpoint Change Unit is used to rapidly switch between several different orthographic viewing modes i.e. Front Viewpoint, Right Viewpoint, Top Viewpoint and Perspective Viewpoint. Fixture designers are allowed to select and switch the appropriate orthographic viewpoint according to their needs during the different design stages in order to support the fixture development process in the VFDAS system. Each orthographic viewing mode associated with this Viewpoint Change Unit provides users the function of zooming in or out the camera to ensure the suitable orthographic viewpoint is identified.

To manage the functions of the Camera Orbit Unit, a set of keys on the keyboard are assigned. These keys are used to support the interactive task of flexibly fine-tuning the 3D design perspective around the target fixture element and adjusting the distances between the scene camera and the virtual assembly workplace. Thus, the fixture designers are allowed to identify the suitable viewpoint and distance for the interactive manipulation according to the visual requirement of users. The keys for the Camera Orbit Unit are Up and Down Arrows, Left and Right Arrows, Page Up, Page Down and Home. The keys of Up and Down Arrows are used to rotate the scene camera around the view X axis of the design environment in VFDAS. The keys of Left and Right Arrows are used to rotate the scene camera around the view Y axis. The key Page Up is used to realize the zoom-in function of the scene camera. The key Page Down is used to realize the zoom-out function of the scene camera. Finally, the key Home is used to accelerate the movement speed of the scene camera in 2 times. In addition, the scene camera is applied to carry out the major operations for the fixture assembly in VFDAS and this camera is specified as a 3D perspective camera.

In terms of the Viewpoint Change Unit, a set of keys on the keyboard are also assigned to carry out the function of switching between different orthographic viewing modes during the interactive design process of the VFDAS system. The keys for Viewpoint Change Unit are specified as the sequential keys of F9, F10, F11 and F12. The key F9 is used to switch on the orthographic viewing mode of ‘Front Viewpoint’. The key F10 is used to switch on another orthographic viewing mode of ‘Right Viewpoint’. The key F11 is used to switch on the orthographic viewing mode of ‘Top Viewpoint’. The key F12 is used to switch on the viewing mode of ‘Perspective Viewpoint’.

5.3.9 Set Initial Location Module

The main function of the Set Initial Location Module is to specify and retrieve the original axis and orientation for each fixture element in the VFDAS system. The role of the Set Initial Location Module is regarded as an assistant tool of the Interactive Design Module presented above, which is applied to the interactive design process of manipulating and positioning the fixture elements to the determined fixturing location.

Using the function of the interactive manipulation provided by the Interactive Design Module, the selected fixture elements are appropriately rotated and positioned according to the locations and orientations users have proposed. Sometimes the fixture designers can not immediately manipulate the fixture elements to the required position and orientation at one goal. They need to undo what they have done in error and restore the target element’s position and orientation using the functions provided by the Interactive Design Module and this may be time-consuming.

Using the Set Initial Location Module, the original axis, orientation and location of every single fixture element in the design environment of VFDAS can rapidly be restored by using the key Ctrl on the keyboard after the target fixture element is selected. Subsequently, the fixture designers are allowed to repeat the interactive assembly process for this element until the satisfactory fixturing location and orientation are determined. Therefore, the fixture design and assembly process is accelerated by the function provided by Set Initial Location Module. The shortage of the Interactive Design Module is also improved by using the Set Initial Location Module.

5.3.10 Interface Menu Manager Module

In order to support the use of the Pre-defined VR Library Module presented above, the main purpose of the Interface Menu Manager Module aims to provide an access to retrieve and input the suitable fixture element from the well-organised fixture element database which is closely associated with the Pre-defined VR Library Module.

The Interface Menu Manager Module comprises a series of pop-up menus according to the reasonable classification structure. These menus are employed to not only display essential information related to the fixture elements from VR library, but also to associate a 3D graphical representation for each element from various viewpoints needed for a fixture design. Using the Interface Menu Manager Module, fixture designers can easily browse and evaluate the fixture element data and drawings to make a final decision for the rapid fixture element selection. In addition, the menus in the Interface Menu Manager Module can be hidden or displayed using the Right Mouse Button according to the users' convenience during the fixture development process.

When fixture designers have made the decision of the element selection by using the menus in this Module, an input request is generated by the VFDAS system and promptly passed to the related logical database routines. Through these routines, the selected fixture element model is assigned with its relevant physical properties before it is imported from the fixture element database to the interactive design environment of VFDAS. Subsequently, the physical fixture design and assembly process is carried out.

5.3.11 Carousel Layout VR Library Module

To compare with the Pre-defined VR Library Module explained in the section 5.3.2, the Carousel Layout VR Library Module aims to tentatively develop a new VR library layout. This new library layout may be applied to provide the users more flexible and intuitive means for the rapid fixture element selection in the VFDAS system. The up to date VR library established in this Module is metaphorized to the style of 'Carousel Layout' **(Fig.5-7)** because every group of 8 fixture elements in this library are arrayed around a circle shape which is similar to a Carousel arrangement. Within the Carousel Layout VR Library, the fixture designers are able to interactively observe and view the candidate fixture elements by using left/right interpolation control so as to identify and select the appropriate fixture element.

The primary advantages of the Carousel Layout VR Library Module are to display the fixture elements in perspective views with a strong 3D visualization and to interactively manipulate a group of fixture elements at the same time. Thus, the users can realistically visualize and understand what the element geometries look like and promptly scan through a large amount of the candidate elements. After narrowing down the range of fixture element selection, the detailed fixture element data and drawings are also available for further refining the selection e.g. geometric dimensions and mechanical characteristics until the final suitable fixture element is determined.

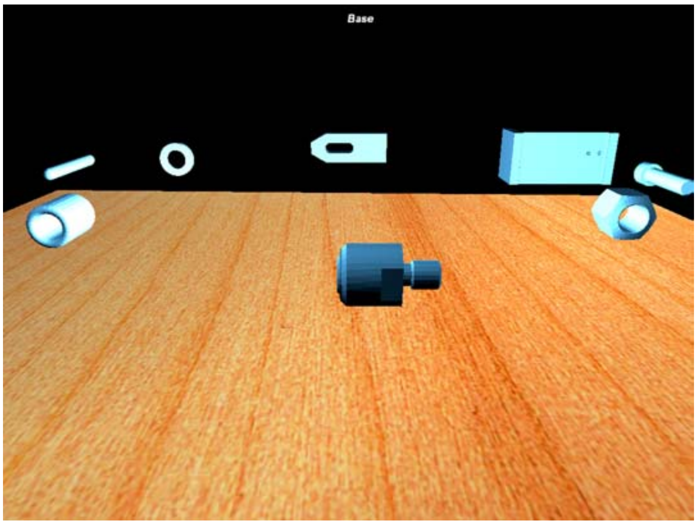


Fig.5-7. The Screenshot of Carousel Layout VR Library Module

5.3.12 Assembly Process Planning and Evaluation Module

The main role of the Assembly Process Planning and Evaluation Module is to provide the functional support of recording the animation for the virtual fixture design and assembly process, i.e. selecting the fixture elements from the VR library and positioning them to generate the final fixture assembly. This fixture design process is interactively carried out by the VFDAS users. Finally, an interactive assembly animation with regard to the fixture development process is produced. Using this assembly animation, a suitable fixture assembly sequence, procedure and trajectories for each fixture element are easily identified and analysed.

Using this Module, the fixture designers can perform the fixture assembly planning and evaluation until the optimal fixture design and assembly plan are determined. The Assembly Process Planning and Evaluation Module allows the assembly animation to replay, pause and even repeat recording by using the control panel placed on the user interface of the VFDAS system. If necessary, the interactive assembly animation recorded by the Module can be exported and stored into the fixture assembly database, which is used for future reference and evaluation. In addition, this module may also facilitate a communication between fixture designers and people from different fields to discover the potential assembly interference through the observation of the recorded assembly animation.

The physics property and behaviour simulation is achieved during the fixture assembly process in VFDAS e.g. real-time 3D collision detection in accuracy. Thus, the interactive assembly animation generated by the Assembly Process Planning and Evaluation Module also demonstrates the physical constraints for the fixture assembly in relation to each fixture element. The physical limitation in use is identified for some fixture elements as if the assembly process is performed in the real physics world. It is helpful for the users who carry out the evaluation with regard to the assembly process.

5.3.13 Save and Restore Assembly Module

The Save and Restore Assembly Module in VFDAS system aims to provide the fixture designers a function by which the uncompleted fixture assembly model that is being developed in progress can be temporarily saved in the computer memory of VFDAS system. Thereafter, the saved uncompleted assembly can be restored according to the users' needs. The fixture designers sometimes make minor mistakes during the fixture design process and require to return the design to the original construction state before the complete assembly model is externally exported to the fixture assembly database by the Design Configuration Output Module, which will be presented in section 5.3.15.

Using the Save and Restore Assembly Module, the original construction state of fixture assembly are saved and restored. The location and orientation of the fixture elements involved in the fixture construction can be conveniently restored to the saved construction state for the original fixture assembly. However, the physical properties associated with these fixture elements will be removed while the operation of restoring the construction state is carried out. The users will have flexible options to re-assign their physical circumstance according to the requirement of fixture design by using the Physicalization Module. Moreover, the control panel with regard to the Save and Restore Assembly Module can be accessed through the GUI of the VFDAS system. This control panel has two sub-functions which are 'Save State' and 'Restore State'.

5.3.14 Machining Interference Check Module

In order to avoid the potential collision and design interference between the proposed machine tool, fixture elements and workpiece, the main function of the Machining Interference Check Module developed in the VFDAS is to extrude and generate essential cutting path envelopes based on a workpiece. The cutting path envelope also refers to the cutting path swept volume. The cutting path swept volume in VFDAS is associated with physical properties. Thus, the solid cutting path swept volume generated by the Machining Interference Check Module has the function of the real-time 3D collision detection in accuracy.

As a result, the fixture elements that are associated with physical properties can not be placed in the solid swept volume or intersected with each other because of the accurate physical collisions. The potential design interference is removed and prevented. Using this Module, the design interference check is achieved in VFDAS to further guarantee collision–free fixture design. This Machining Interference Check Module is the efficient and innovative method that is used to realize the capability of interference avoidance for a fixture design in the VFDAS.

5.3.15 Design Configuration Output Module

In order to generate the integration of the VFDAS system and CAD systems as well as spread the future uses of the VFDAS in industry, the Design Configuration Output Module is developed based on the Virtools CAD Pack to produce the useful outputs for the VFDAS system. The outputs generally are a 3D fixture assembly model with the CAD format *.DXF, its bill of materials (BOM) and an interactive assembly animation. The fixture assembly model is developed during the interactive design process in the VFDAS. The interactive assembly animation comprises the data related to fixture assembly i.e. assembly procedures, assembly sequence and assembly trajectory for each fixture element.

After the fixture design process within VFDAS system, the output of *.DXF fixture assembly model contains essential information related to the spatial relationship and relative position for each element. If necessary, the essential information can be accessed and edited by using the standard CAD system such as: Pro/ENGINEER.

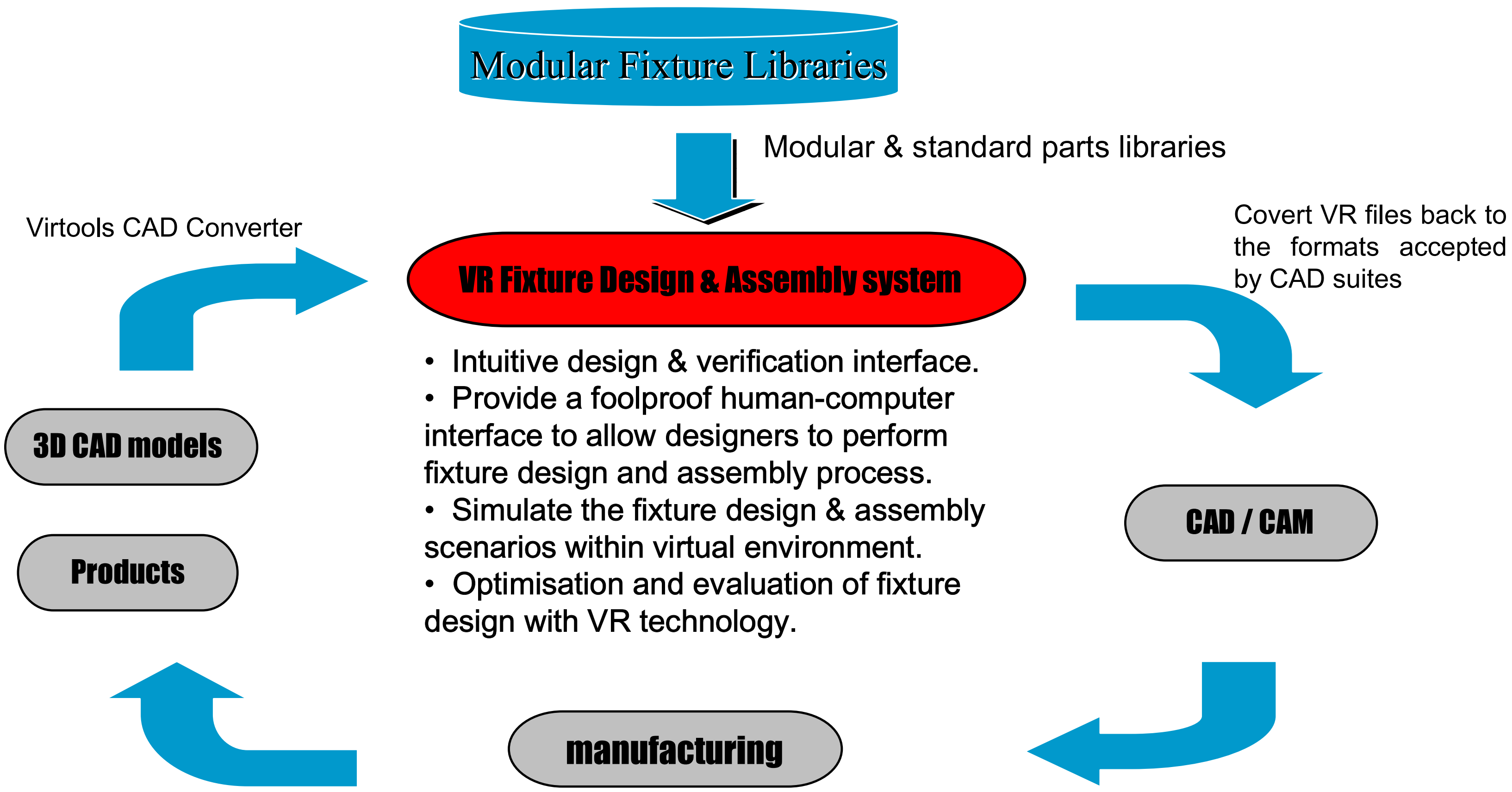
Subsequently, the manufacturing process is carried out according to the essential output obtained from the VFDAS system.

5.4 Application Principles of the VFDAS System

In order to explain how the proposed VFDAS system is applied to the industrial practices, an operational flowchart is created to schematically demonstrate the total application principles of the VFDAS system. This flowchart is shown in **Fig.5-8**.

After a workpiece model is created using a CAD package such as: Pro/ENGINEER, the model of the workpiece is then exported with a portable CAD format assigned by the fixture designers. The workpiece model can be converted to the compatible *NMO format for the VFDAS by using the Virtools CAD Converter. Thereafter, the target workpiece model is ready for the further fixture design process within the VFDAS system. As soon as the workpiece model is loaded into the VFDAS system, a series of interactive operations in relation to the fixture design and assembly evaluation are subsequently carried out using the functionality of VFDAS.

After conducting overall functions provided by the VFDAS system i.e. fixture element selection, fixture layout design, assembly planning and essential analysis, the final fixture configuration design is determined. In addition, the well-organised VR library of modular fixtures in VFDAS system provides design alternatives for the rapid fixture element selection and simplifies the fixture design process. To facilitate the subsequent manufacturing operations, the final optimized fixture/workpiece assembly model associated with the essential data of 3D spatial representation is exported from the VFDAS system as a fixture design output and returned to the standardised CAD/CAM systems that are widely used for the manufacturing process in industry. Finally, the eligible products in terms of the manufacturing requirement are fabricated.


Modular Fixture Libraries
Modular & standard parts libraries
Virtools CAD Converter
VR Fixture Design & Assembly system
Covert VR files back to the formats accepted by CAD suites
3D CAD models
Products
CAD / CAM
manufacturing

- Intuitive design & verification interface.
- Provide a foolproof human-computer interface to allow designers to perform fixture design and assembly process.
- Simulate the fixture design & assembly scenarios within virtual environment.
- Optimisation and evaluation of fixture design with VR technology.

Fig.5-8. The Application Principles of the VFDAS

5.5 The Industrial Role of the VFDAS System

In order to introduce the relationship between the VFDAS system and other production procedures, another operational flowchart is created to schematically demonstrate the primary industrial role that the VFDAS system acts as during the manufacturing process. This flowchart is shown in **Fig.5-9**.

The production orders are given by the industrial customers who intend to produce certain type of mechanical merchandise for the potential market. As soon as the production orders reach the relevant manufacturer, the process planning for manufacturing then is carried out according to the manufacturing requirements provided by the customer. The fixture design acts as an important role throughout the manufacturing process. The fixture designers are required to develop an optimized fixture configuration that can be appropriately applied for the proposed machining operations. This fixture configuration needs to provide an eligible workholding device to locate the workpiece in proper position and orientation as well as prevent the excessive deformation.

The VFDAS system was developed to achieve the fixture design and assembly process for any workpiece provided by the users. The VFDAS system supports the fixture designers to carry out the fixture design process. After the optimized fixture configuration is developed using the VFDAS system, overall modular fixture elements involved in the final fixture configuration may be purchased from the commercial fixture providers. These elements are subsequently used to build up the physical fixture device for the machining operations. As a result, the interactive VFDAS system efficiently performs its role and responsibility for supporting the fixture design during the manufacturing process in industry.

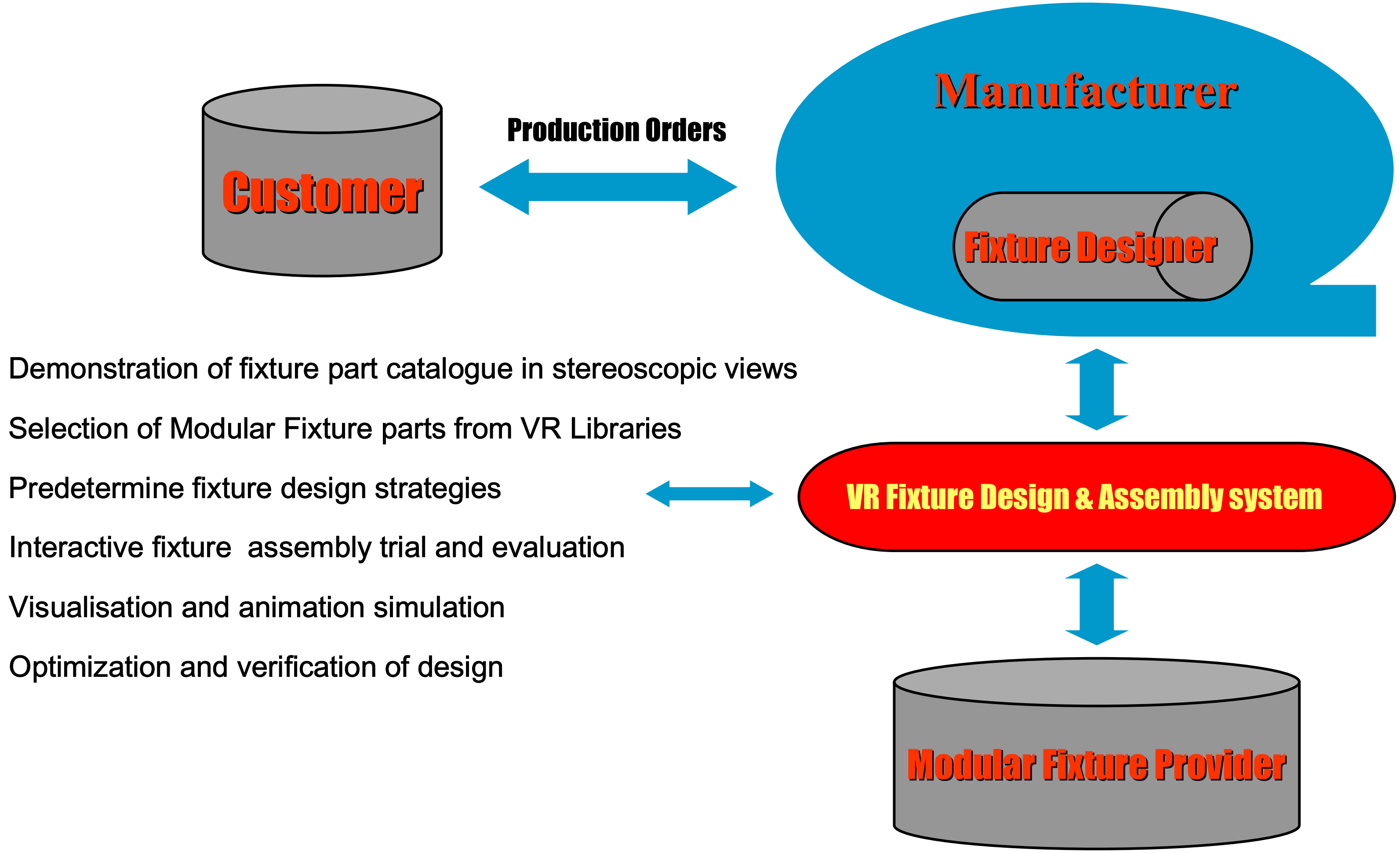

Customer
Production Orders
Manufacturer
Fixture Designer
Demonstration of fixture part catalogue in stereoscopic views
Selection of Modular Fixture parts from VR Libraries
Predetermine fixture design strategies
Interactive fixture assembly trial and evaluation
Visualisation and animation simulation
Optimization and verification of design
VR Fixture Design & Assembly system
Modular Fixture Provider

Fig.5-9. The Industrial Role of the VFDAS System

5.6 Identification of the VFDAS Development Requirements

5.6.1 The Concept of Developing a Modular Fixture

The comprehensive VFDAS system developed in this thesis focused on the research aspect of the modular fixture development because modular fixtures are the most suitable type of fixturing systems which are used for performing a CAFD process so as to produce a greater computerization impact. Accordingly, the concept regarding how to develop a modular fixture must be clearly understood before the development requirements of the proposed VFDAS are determined.

Modular fixtures are commonly developed using the principle of construction sets similar to Lego blocks. The construction sets are usually fastened and assembled together using interchangeable multipurpose fixturing elements which are repeatedly used to hold and constrain various workpieces during a machining operation. A typical modular fixture system comprises a number of standard fixturing elements such as: bases, locators, clamps, and supporting elements. A base element is required upon which the other fixturing elements are selected and mounted. To simplify extension design, T-slots, tenon-slots, dowel or grid-holes and tapped holes are distributed on all facets of the base elements. The supporting, locating and clamping fixture elements are connected together using bolts held in T-nuts or using capped screws [88].

In addition, the main advantages of modular fixture beyond the dedicated fixture are summarized as follows [88] [89]:

(1). Appropriate for a wide variety of different parts and adaptable to changes in design, process plan and machine tool.
(2). Most often used for trial, prototype work and temporary replacement
(3). Reduction of design time by the retrieval of standardised model of fixturing elements from a CAD system
(4). Often applied to deal with small batch and short cycle production
(5). Reduction of the requirement for large storage space because the fixturing elements can be re-used for other fixturing needs once dismantled.

(6). Fixture construction can often be performed without the need for engineering drawings.
(7). Deduction of labour expenditure and maintenance costs
(8). Reduction in the lead-time between fixture design and fabrication
(9). Enable a faster response to customer's needs

Furthermore, a modular fixture is constructed around either a workpiece or a prototype model before a manufacturing operation. A fixture device will be photographed and a list of its fixturing elements will be generated as soon as it is produced. The photographs and element lists are archived for future reference. The fixture can be disassembled while a machining task is completed so that all fixture elements are delivered back for storage. A modular fixture can be built within one day to one week comparing with the production of a dedicated fixture that sometimes requires the period up to two months.

The deficiency of a manual means for producing a fixture is due to the issue of retrieving previous experience and knowledge from massive photographs and element lists. As a result, too much paperwork is often involved and high manual design skills are required to develop fixtures. To overcome these shortcomings, CAD systems were introduced to the modular fixture design [90] [91].

5.6.2 Modular Fixture Design Using CAD Systems

Many commercial and academic CAD systems are developed for fixture design. The main mode of application normally focuses on the feature of manually selecting and positioning of predefined models of fixturing elements around a workpiece to comprise a graphical fixture assembly. The majority of CAD systems for modular fixture design provide a CAD library which contains essential fixturing elements that are given different viewpoints needed for a fixture design. An effective reduction of the fixture design workload for the users is achieved by developing the CAD libraries which include a series of classified modular fixturing elements in order to facilitate the fixture design and assembly process. Thus, several typical CAD systems are integrated with these types of modular fixturing CAD libraries, which will be listed below [7]:

1. Procad with PROREN
2. Isykon with PROREN
3. Cadlink with CIMCAD
4. Nixdorf with Proren
5. Norsk Data with Technovision
6. Strassle with KONSYS
7. Command with M.CAD
8. Prime/CV with MEDUSA

The primary advantages of using a CAD system towards modular fixture development are briefly outlined as follows [7].

1. Deduction of necessary engineering drawings (approximately 90%)
2. Deduction of engineering drawing time involved (approximately 90%)
3. The requirement of more technicians than qualified engineers

However, the main inadequacy of these CAD systems results from their limited ability to select the suitable fixture elements and automatically place them around a workpiece. Due to these fixture models created by using 2D drafting facilities, potential machining and assembly collisions caused during the fixture development process can not be identified very easily. These factors are considered as the main reason why some research gives rise to the evolution of knowledge-base or rule-base fixturing systems with a built-in database that are normally integrated with a 3D solid CAD toolkit.

Using the interactive VFDAS system, the advantages of VR are employed to improve the capability of the traditional CAD systems. These advantages of VR refer to realistic visualization, high-level of interactivity, physics property simulation and real-time 3D collision detection in accuracy etc. The deficiency of conventional CAD fixturing systems related to potential assembly collision is improved by accurate real-time 3D collision detection provided by VFDAS. The development of the VFDAS system aims to explore an intuitive approach of manipulating the models with a VE. Using this approach, the manual skills and experience involved for modular fixture design are reduced and the fixture design process is fulfilled as if in the real physics world.

5.6.3 The Method for Identifying Requirements of VFDAS

In order to establish the rich-functioning VFDAS system, the requirements that arise from the fixture design process need to be determined to facilitate the VFDAS framework development. The potential VFDAS system should be considered as a special type of CAFD system. The requirements that have been used for the development of existing CAFD systems are regarded as guidance which can be refined and converted into the development of VFDAS. Thus, advantages generated by VR are brought to improve the existing CAFD's versatility and functionality.

At the early stage of VFDAS development, the requirements for basic functionality were derived from the existing CAFD systems. Most of fundamental requirements from the traditional CAFD systems have been already reported in a great deal of similar research [7] [60] [91]. Accordingly, it is not essential to explore and identify the development requirements for the VFDAS at a separate research procedure.

According to these requirements outlined from the existing CAFD systems, the most appropriate research objectives with regard to the VFDAS development were identified. More details related to these objectives of VFDAS can be found in previous section 1.4.

5.7 Summary

The specific functionality and system architecture related to VFDAS system were introduced and explained in this chapter. The primary 14 functioning modules associated with the VFDAS system that comprise the fundamental system framework were demonstrated carefully to explain the specified functionality and interaction activity involved in each functional module. Furthermore, the application mechanism and industrial role the VFDAS system has engaged throughout the system implementation process were presented and explained. The method of how to identify the requirement and functionality included in the development of the VFDAS system were also clarified.

In a nutshell, the comprehensive VFDAS system created in this thesis efficiently narrowed the research gap between VR and Fixture Design. The VFDAS system was appropriately used to support the fixture development process so as to invent a novel methodology with regard to fixture design and redefine the conceptualization of fixture development.

Chapter 6. Functionality and System Development of VFDAS

6.1 Introduction

This chapter details the development process of the VFDAS system. This process is theoretically divided into three incremental development phases as shown in **Fig.6-1** so as to systematically construct the system architecture and eventually establish a comprehensive CAMFD system entitled VFDAS. A series of functionality and interaction features were gradually incorporated into VFDAS during these development phases in order to support the conventional fixture development process.

The relevant VR behaviour programming details related to the completion of each necessary function and interaction towards the development of VFDAS are explained in the following sections. In addition, several experimentations in relation to the VR behaviour programming of physics interaction simulations that comprise the Pendulum Simulation, the Spring Physics Mechanism and Mechanical Component Showcase were carried out to explore how to appropriately realize the physics properties and interactive behaviour simulations within a VR environment. Examples of these experimentations can be seen in Appendix. B.

6.2 The Navigation of the VFDAS Development Process

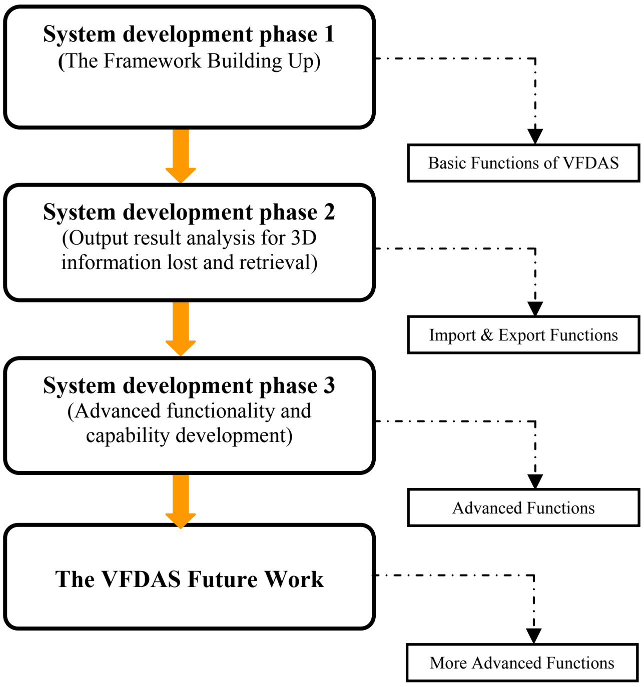

System development phase 1
(The Framework Building Up)
Basic Functions of VFDAS
System development phase 2
(Output result analysis for 3D information lost and retrieval)
Import & Export Functions
System development phase 3
(Advanced functionality and capability development)
Advanced Functions
The VFDAS Future Work
More Advanced Functions

Fig.6-1. Three Development Phases of the VFDAS and Future Work

A. System Development Phase I (Primary framework building up)

- Fixture element library construction (drop menu layout, classification of fixtures, restoration of saved fixture tower, 3D fixture element models import & storage issues)
- Basic manipulation function for spatial design, fixture layout design and assembly within VE
- Control panel design for the function of assigning and removing physics properties of the selected 3D objects

B. System Development Phase II (the VFDAS output result analysis for 3D information lost and retrieval)

The related 3D information evaluation via the fixture model conversion process from the VFDAS system to Pro/ENGINEER are efficiently carried out throughout the 2nd VFDAS development phase, such as: physical properties, geometric information, spatial relationship data of virtual assembly and tolerance information etc.

C. System Development Phase III (Advanced function development associated with the VFDAS system)

After building the fundamental functions in VFDAS that can carry out the simple fixture design process, the development of other advanced functions with regard to the system upgrade is performed through this development phase to further develop the VFDAS system. Therefore, a number of functions in terms of fixture design are developed and integrated into the VFDAS. These functions are summarized as follow:

- Physics properties and behaviour simulation for 3D fixture elements
- Accurate collision detection and interference analysis
- Interactive operation and manipulation
- Assembly animation recording and interactive VR simulation
- Machining strategy design
- Modular fixture element library development
    - Classification
    - 3D Data storage
    - 3D Data retrieval
    - Bill of materials
    - Cost analysis
    - Input menu layout , specification description and graphical representation for each fixture element
    - Carrousel type of VR library
- Assembly state saving and restoration
- 3D Design perspective control and orthographic view changes
- Grouped objects function and translation
- Set initial condition for fixture elements
- Menu manger design and layout

In addition, the potential applications in VFDAS that may be achieved in the future work are explored and discussed. Consequently, the research scope in relation to future VFDAS development is expanded and the possibility of future research domains is identified. These potential research domains for future development of VFDAS are presented below:

- Computer Semi-automated or automated fixture design in VFDAS
- Advanced kinematics analysis
- Human factors and safety assessment
- Tolerance analysis
- Finite element analysis
- Intuitive manipulation and interaction (external device employment e.g. haptic data gloves)
- Full immersion and presence design platform used in VFDAS (external devices e.g. Head mounted display or Cyber glasses)
- Fixture data management and rule-based knowledge database

6.3 System Development Phase One

The method for developing the VFDAS system is considered as an incremental development process. To develop the primary VFDAS framework, three functional objectives were proposed at development Phase One. In the VFDAS framework, the basic functions for performing the interactive fixture design and assembly process are achieved. These three functional objectives at development Phase One were the Pre-defined VR Library of Fixture Elements, the function of Interactive Selection and Manipulation and the function of Assigning and Changing the Physics Modes. Therefore, the following sections describe how each functional objective was developed according to the specific behaviour programming process using Virtools.

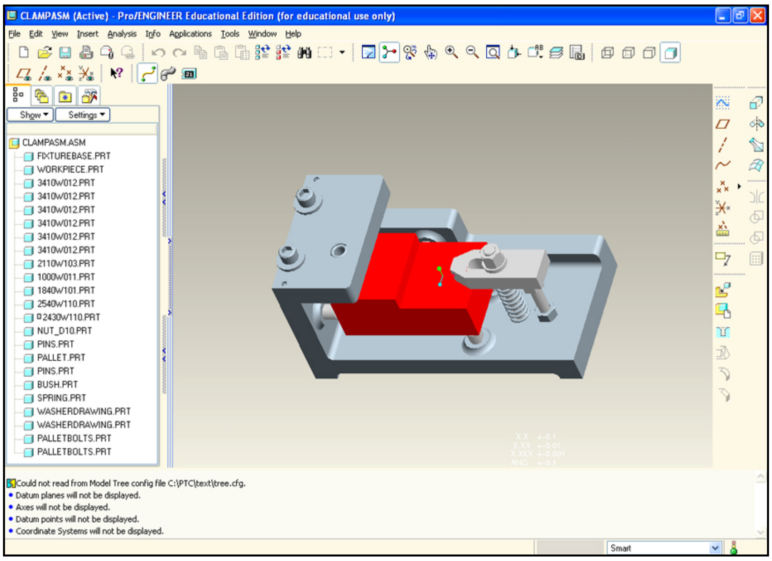
6.3.1 The Functional Objective of Developing the Pre-defined VR Library

The first functional objective was to develop a built-in pre-defined VR library within the interactive VFDAS system. This VR library contains a series of well-classified fixture elements to provide fixture designers more alternatives for a fixture design and realize the rapid fixture element selection as an important functional module in VFDAS system. More details with regard to this functional module can be seen in previous section 5.3.2.

In order to develop useful functions included in the Pre-defined VR Library Module, the detailed behaviour programming process will be presented and explained in the following sections.

6.3.1.1 The Creation of CAD Models for Fixture Assembly

A demonstration example of a simple fixture assembly **(Fig.6-2)** is used to perform a fixture design and assembly process within the VFDAS system so as to demonstrate how the VFDAS system works within an interactive VR physics environment. More details with regard to this example can be seen in previous section 5.3.1.


CLAMPASM (Active) - Pro/ENGINEER Educational Edition (for educational use only)
File Edit View Insert Analysis Info Applications Tools Window Help
Show
Settings
Smart

CLAMPASM.ASM
FIXTUREBASE.PRT
WORKPIECE.PRT
3410W012.PRT
3410W012.PRT
3410W012.PRT
3410W012.PRT
3410W012.PRT
3410W012.PRT
2110W103.PRT
1000W011.PRT
1840W101.PRT
2540W110.PRT
2430W110.PRT
NUT_D10.PRT
PINS.PRT
PALLET.PRT
PINS.PRT
BUSH.PRT
SPRING.PRT
WASHERDRAWING.PRT
WASHERDRAWING.PRT
PALLETBOLTS.PRT
PALLETBOLTS.PRT

Could not read from Model Tree config file C:\PTC\text\tree.cfg.
- Datum planes will not be displayed.
- Axes will not be displayed.
- Datum points will not be displayed.
- Coordinate Systems will not be displayed.

Fig.6-2. The Fixture Assembly Model Created in Pro/ENGINEER

Each fixture element model that forms the simple fixture assembly was either created using Pro/ENGINEER or derived from commercial modular fixture catalogues. As soon as these fixture elements were chosen, the fixture assembly model was created using Pro/ENGINEER. Subsequently, this fixture assembly model was converted into Virtools using the integration method between VR and CAD systems that has already been presented in previous section 3.3.

After the conversion process, the behaviour programming process for developing the VFDAS system was continuously carried out in Virtools to realize the interactivity, functionality, physical property simulation and realistic visualization in the VFDAS.

6.3.1.2 The Behaviour Programming for Loading Buttons

According to the requirement of a quick access to the pre-defined VR library, a set of loading buttons affiliated to the pop-up menus on the graphical user interface (GUI) of the VFDAS are created. These loading buttons are pressed by the users to select and input the suitable fixture elements from the systematic fixture database which was constructed as a part of the VR library. To explain how this functional feature of a loading Button is achieved, the specific behaviour programming content that was assigned to a typical loading button named ‘Clamp’ **(Fig.6-3)** is illustrated in **Fig.6-4**.

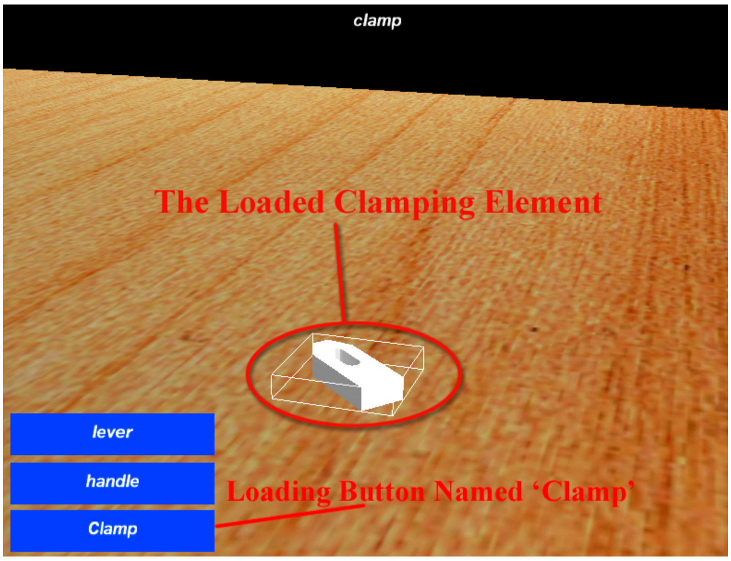


Fig.6-3. The Loading Button for the Clamping Element

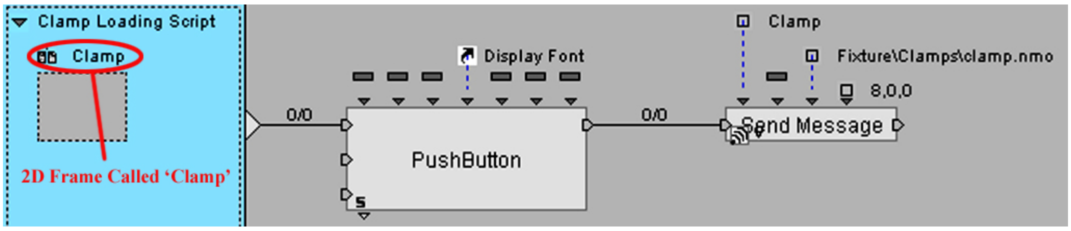


Fig.6-4. The Behaviour Programming Script Applied to 2D Frame Called 'Clamp'

This loading button is built on a 2D Frame called 'Clamp' in Virtools. The 2D Frame is a type of design element, which is used as a place holder while creating a user interface within Virtools environment. In order to fulfill the loading button depended on the 2D Frame, two primary BBs are applied to the target 2D Frame named 'Clamp' during this behaviour programming process **(Fig.6-4)**.

One important BB is named 'Push Button' using which a 2D Frame is converted into a push button. The introductory graph of 'Push Button' BB is indicated below **(Fig.6-5)**. More details about this 'Push Button' BB can be found in Appendix. C.

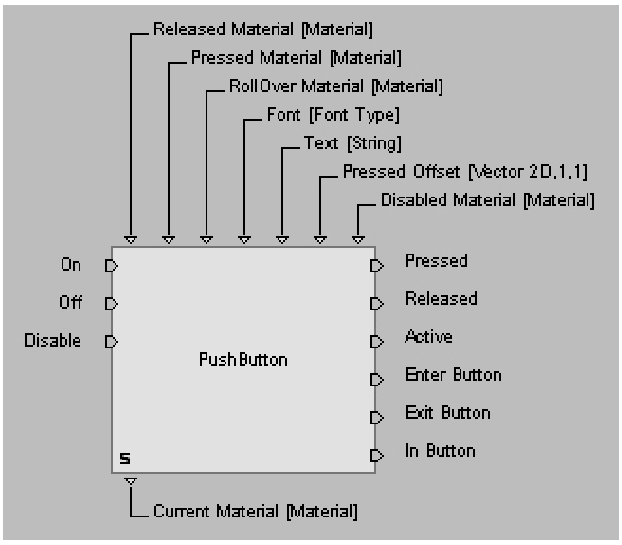

Released Material [Material]
Pressed Material [Material]
RollOver Material [Material]
Font [Font Type]
Text [String]
Pressed Offset [Vector 2D,1,1]
Disabled Material [Material]
On
Off
Disable
PushButton
Pressed
Released
Active
Enter Button
Exit Button
In Button
Current Material [Material]

Fig.6-5. The Introductory Graph of The BB named 'Push Button'

Another BB named 'Send Message' is used to send a message to the target 3D Entity, via which the relevant data bound with the message are simultaneously carried to the designated recipients. The BB of 'Send Message' is also used to trigger the BBs while a particular event takes place on a different Script rather than the current one. Furthermore, these bound data can be retrieved by using the "Get Message Data" BB if applicable. The introductory graph of ‘Send Message’ BB is indicated below **(Fig.6-6)**. More details about this ‘Send Message’ BB can be found in Appendix. C.

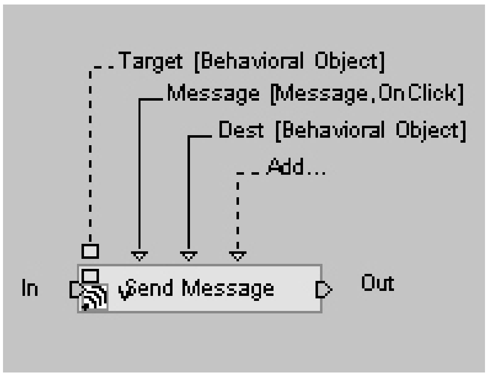


Fig.6-6. The Introductory Graph of The BB named ‘Send Message’

To explain the logic and correlation for the behaviour programming between these two BBs, the total process of specifying and connecting these logical BBs to a workflow has to be clarified in detail. Through this process, the relevant Parameters are specified as well as input and output ports of these BBs are connected by the Behaviour Links (bLink) or Parameter Links (pLink) so as to propagate data values within this visual scripting environment. More details about how to adjust and connect these two BBs can be seen in Appendix. D.

As a result, the logic and correlation of this specific behaviour programming semantics for these two BBs are understood as follows. The ‘Push Button’ BB is used to transform the 2D Frame named ‘Clamp’ into a push button which becomes the latter ‘Clamp’ loading button. As soon as the ‘Clamp’ loading button is released by the system users, an activation flow will be transmitted to trigger the next ‘Send Message’ BB. Thereafter, the ‘Send Message’ BB is able to propagate an important message to the 3D Entity named ‘Level’ as a recipient. This message is bound with the relevant input data with regard to the fixture element called ‘Clamp’ e.g. the input location and retrieval directory according to the fixture database. Subsequently, the Script applied to ‘Level’ will obtain the input data encapsulated in the message so as to retrieve and input the ‘Clamp’ fixture model according to the requirement. More details with regard to this Script will be presented in the next section.

Due to the similar principle of the VE programming, the specific behaviour programming applied to the 'Clamp' loading button described above was efficiently used to create a host of other loading buttons existing in the pre-defined VR library of the VFDAS using the same programming principle.

6.3.1.3 The Behaviour Programming for the Input Interaction of a Fixture Element

To complete the input interaction of retrieving a fixture element from the fixture database to the virtual design environment of VFDAS after the loading button is pressed by the users, a group of BBs are applied to perform this specific behaviour programming process **(Fig.6-7)**. These BBs are all assigned to the 3D Entity named 'Level' during the process of developing this functional feature. Due to 'Level' acting as a global container, it holds everything that is in a Virtools composition. The BBs and a Behaviour Graph involved to this behaviour programming process **(Fig.6-7)** are the 'Wait Message' BB, the 'Get Message Data' BB, the 'Object Load' BB, the 'Set Position' BB, the 'Identity' BB and the Behaviour Graph called 'Parts Collision'. This Behaviour Graph is used to associate a set of physical properties with the element model that is just retrieved from the fixture database by the Script named 'Object Load'.

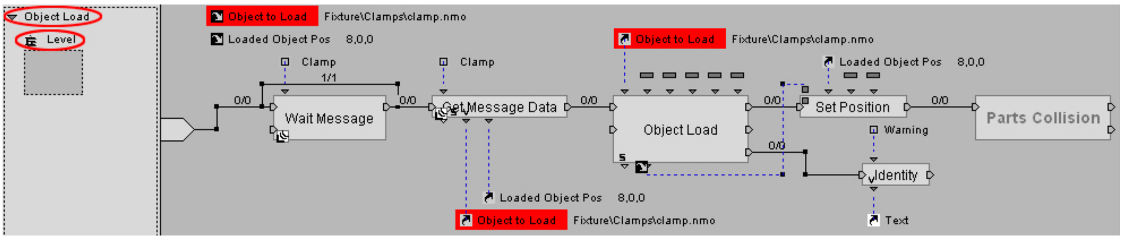


Fig.6-7. The Behaviour Programming Script Assigned to the 'Level'

To realize the functional feature of input interaction and retrieve the fixture element model named 'Clamp' from the fixture database, the BBs assigned to the 3D Entity named 'Level' require to be explained in detail concerning their individual function. The BB named 'Wait Message' is used to wait the receipt of a target message, by which a useful activation output can be generated once the message is received. The BB named 'Get Message Data' is used to interpret and retrieve the data which were bound with the selected message. The BB named 'Object Load' is applied to load the designated *.NMO file that includes one or more 3D objects. The BB named 'Set Position' is used to set the position of a 3D Entity with in a VE. The BB named

‘Identity’ is applied to output the Input Parameters while it is activated. The introductory graphs with regard to these five BBs are presented according to the five following pictures so as to clarify more specification about them **(Fig.6-8) (Fig.6-9) (Fig.6-10) (Fig.6-11) (Fig.6-12)**.

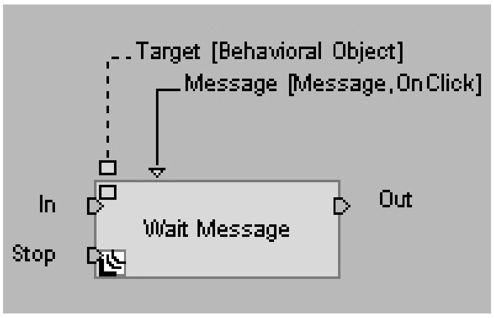


Fig.6-8. The Introductory Graph of the BB Named ‘Wait Message’

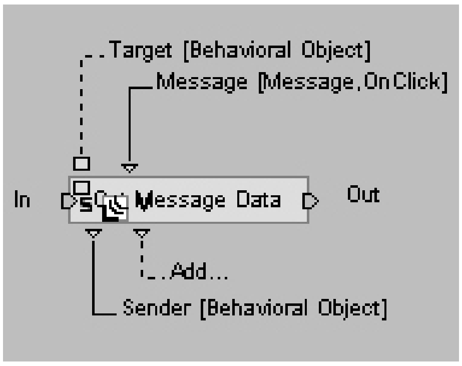

Target [Behavioral Object]
Message [Message,OnClick]
In
Message Data
Out
Add...
Sender [Behavioral Object]

Fig.6-9. The Introductory Graph of the BB Named ‘Get Message Data’

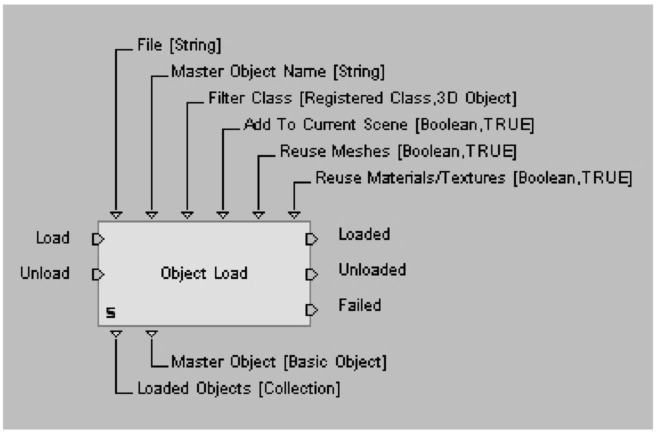

File [String]
Master Object Name [String]
Filter Class [Registered Class,3D Object]
Add To Current Scene [Boolean,TRUE]
Reuse Meshes [Boolean,TRUE]
Reuse Materials/Textures [Boolean,TRUE]
Load
Unload
Object Load
Loaded
Unloaded
Failed
Master Object [Basic Object]
Loaded Objects [Collection]

Fig.6-10. The Introductory Graph of the BB Named ‘Object Load’

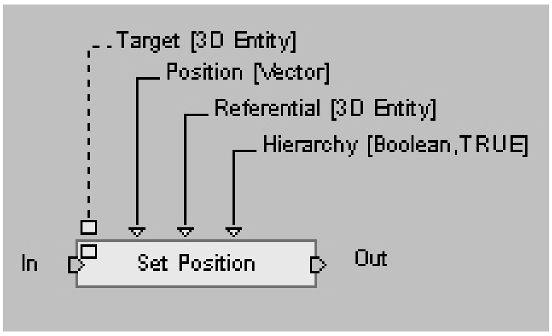


Fig.6-11. The Introductory Graph of the BB Named ‘Set Position’

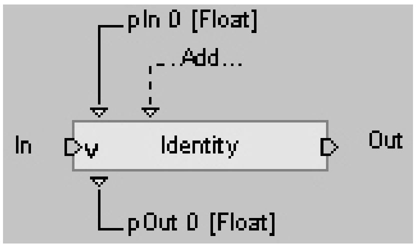


Fig.6-12. The Introductory Graph of the BB Named ‘Identity’

More details about how to specify and connect these five BBs and one Behaviour Graph can be seen in Appendix. E.

The logic and correlation of this behaviour programming for these five key BBs are as follows. Once the message named ‘Clamp’ generated by the ‘Clamp Loading Script’ Script described in previous section in 6.3.1.2 is passed to the specified recipient of ‘Level’, the following Script named ‘Object Load’ presented in this section will be activated. The ‘Wait Message’ BB is used to generate a useful activation until the message named ‘Clamp’ is received. Once the activation of ‘Wait Message’ BB is passed to next ‘Get Message Data’ BB, the relevant input data encapsulated with this ‘Clamp’ message are retrieved by this ‘Get Message Data’ BB. These input data related to the fixture element called ‘Clamp’ are its input location and retrieval directory according to the fixture database. According to the input data of the retrieval directory, the ‘Object Load’ BB then loads the ‘Clamp’ fixture element model with the *.NMO format from the fixture database. According to another input data of the input location, the loaded ‘Clamp’ model is placed to the specified position in a VE by using the ‘Set Position’ BB. The ‘Identity’ BB displays the warning message in the GUI of VFDAS if the ‘Object Load’ BB fails to retrieve the ‘Clamp’ model from the fixture database.

Therefore, the functional feature of input interaction with respect to the fixture element 'Clamp' is finally achieved.

6.3.1.4 The Behaviour Programming for Physicalizing a Fixture Element

As soon as the fixture element named 'Clamp' is loaded from the fixture database to the VFDAS by the five BBs related to the input interaction, the Behaviour Graph called 'Parts Collision' is continuously used to associate a series of physical properties with the element named 'Clamp'. This Behaviour Graph activates the physicalization process while an activation flow generated by the 'Set Position' BB is received. To present how this functional feature of physicalizing a fixture element is achieved, the behaviour programming content that was involved in the 'Parts Collision' Behaviour Graph is shown in **Fig.6-13**.

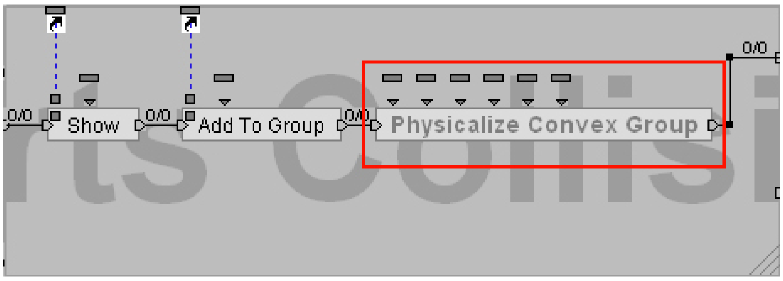


Fig.6-13. The Expanded Behaviour Graph Named 'Parts Collision'

A set of different BBs are applied to this behaviour programming process, which are all included in the Behaviour Graph named 'Part Collision'. The primary BBs and Parameter Operations used to the behaviour programming process in the 'Parts Collision' Behaviour Graph are the BB named 'Show', the BB named 'Add To Group', the BB named 'Group Iterator', the BB named 'Physicalize', the Parameter Operation called 'Get Current' and the Parameter Operation called 'Get Name'. In addition, another Behaviour Graph named 'Physicalize Convex Group' is affiliated into the 'Parts Collision' Behaviour Graph.

Furthermore, the 'Physicalize Convex Group' Behaviour Graph comprises the BB named 'Group Iterator', the BB named 'Physicalize', the Parameter Operation called 'Get Current' and the Parameter Operation called 'Get Name'. The expanded 'Physicalize Convex Group' Behaviour Graph is shown in **Fig.6-14**.

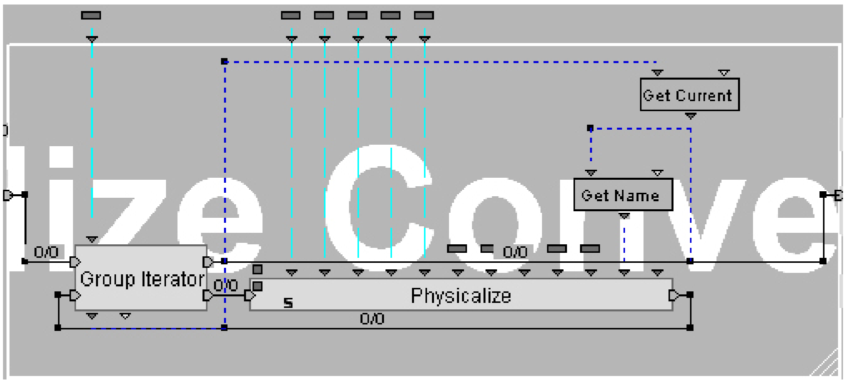


Fig.6-14. The Expanded Behaviour Graph Named 'Physicalize Convex Group'

The BB of 'Show' is used to show a 2D/3D Entity, a Mesh or a Group to ensure that the target object is visible in the VE scene. The BB of 'Add To Group' is used to interpolate the object to an arranged Group. The BB of 'Group Iterator' is used to retrieve each element of a group. The BB named 'Physicalize' is considered as the important BB during the behaviour programming process. This BB is applied to define a 3D object becoming a part of the physics world. The introductory graphs with regard to the four BBs are presented according to the following pictures in order to clarify more their specifications **(Fig.6-15) (Fig.6-16) (Fig.6-17) (Fig.6-18)**.

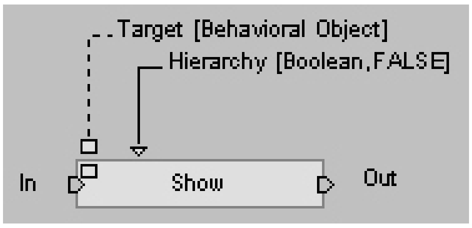


Fig.6-15. The Introductory Graph of the 'Show' BB

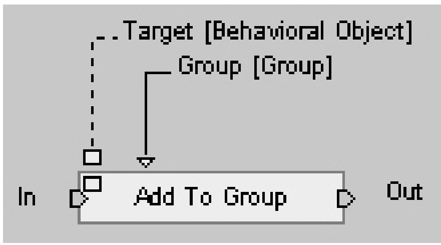


Fig.6-16. The Introductory Graph of the 'Add To Group' BB

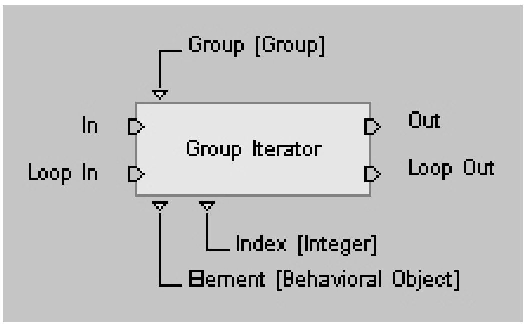


Fig.6-17. The Introductory Graph of the ‘Group Iterator’ BB

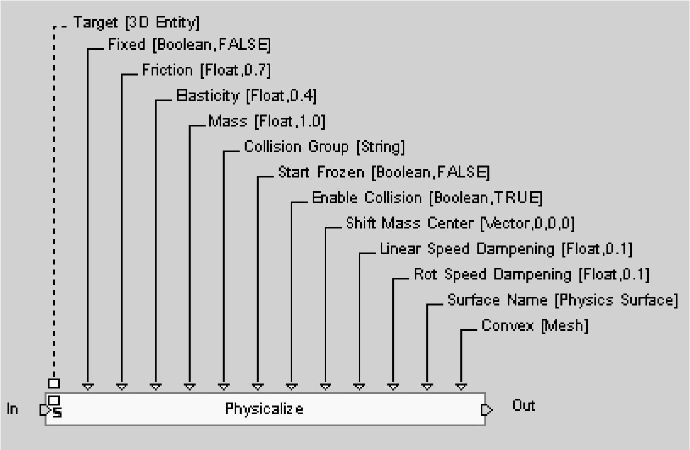
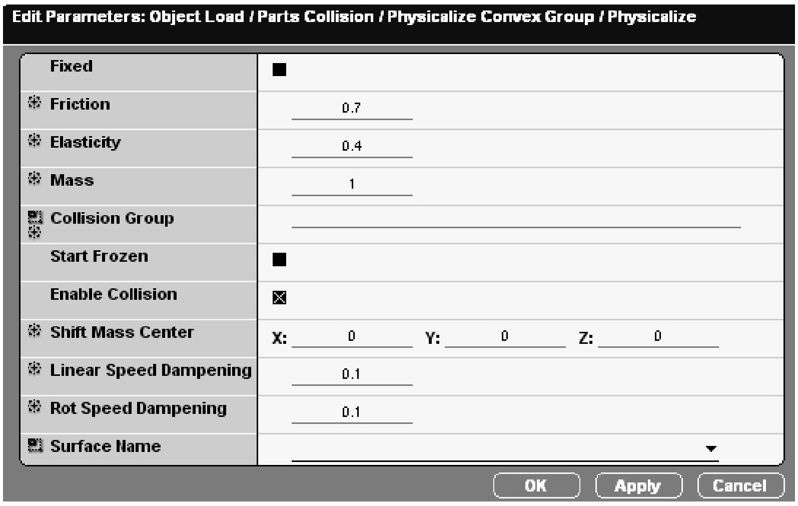

Target [3D Entity]
Fixed [Boolean,FALSE]
Friction [Float,0.7]
Elasticity [Float,0.4]
Mass [Float,1.0]
Collision Group [String]
Start Frozen [Boolean,FALSE]
Enable Collision [Boolean,TRUE]
Shift Mass Center [Vector,0,0,0]
Linear Speed Dampening [Float,0.1]
Rot Speed Dampening [Float,0.1]
Surface Name [Physics Surface]
Convex [Mesh]
In
Physicalize
Out

Fig.6-18. The Introductory Graph of the ‘Physicalize’ BB

In particular, the specification of ‘Physicalize’ BB is carefully explained according to the introductory graph in **Fig.6-18**. The Behaviour Input in this BB indicates that the BB process will be triggered once an activation flow is received. The Behaviour Output indicates that it will be activated while the BB process is completed. The Parameter ‘Fixed’ denotes that the object is considered as unmovable if checked. The term of ‘unmovable’ is defined as the gravity and force will no longer affect on it. The Parameter ‘Friction’ is used to specify the friction properties of the physics material assigned with the object. The Parameter ‘Elasticity’ is used to define the bouncing properties of the physics material associated with the object. The Parameter ‘Mass’ is used to define how heavy the object should be. The Parameter ‘Collision Group’ is considered as the filter string that determines the collision rules used for this object. The Parameter ‘Start Frozen’ means that if checked, the object will not be affected by gravity until some event activates it e.g. collision with another object or an impulse etc. The Parameter ‘Enable Collision’ means that if checked the collision between this object and other physical objects is enabled. The Parameter ‘Shift Mass Center’ defines the mass centre of this object. The Parameter ‘Linear Speed Dampening’ defines the dampening on linear speed (Object’s Translations) using which the air resistance is appropriately simulated. The Parameter ‘Rot Speed Dampening’ defines the dampening on rotation speed (Object‘s Rotations). The Parameter ‘Surface Named’ gives a name to the physics surface created for the target object. The Parameter ‘Convex’ means the mesh specified here is considered as a convex, and if it is not, it is only considered as its convex hull to generate the physics object’s topology. The overall Parameters of ‘Physicalize’ BB that are described above can be specified through Edit Parameters as shown in **Fig.6-19**.


Edit Parameters: Object Load / Parts Collision / Physicalize Convex Group / Physicalize

Fixed	■
Friction	0.7
Elasticity	0.4
Mass	1
Collision Group	
Start Frozen	■
Enable Collision	☒
Shift Mass Center	X: 0 Y: 0 Z: 0
Linear Speed Dampening	0.1
Rot Speed Dampening	0.1
Surface Name	

Fig.6-19. The Edit Parameters for the ‘Physicalize’ BB

The logic and correlation with regard to the behaviour programming that is included in the ‘Parts Collision’ Behaviour Graph are as follows **(Fig.6-13)**. The ‘Parts Collision’ Behaviour Graph will activate the physicalization process as soon as an activation flow is received. The first BB ‘Show’ in this Behaviour Graph is used to ensure the loaded fixture element 3D object named ‘Clamp’ is visible in the VE scene of the VFDAS. Once the activation flow is received, the next BB ‘Add To Group’ will add this ‘Clamp’ 3D object into a target Group named ‘Physics’ that is created by the VE developer. The target Group is used to facilitate the further group physicalization process. Thereafter, the BB ‘Group Iterator’ in ‘Physicalize Convex Group’ Behaviour Graph is used to retrieve each member of the ‘Physics’ Group that contains the ‘Clamp’ 3D object. The ‘Physicalize’ BB associates physical properties with every member of the ‘Physics’ Group that is parsed by the ‘Group Iterator’ BB. The data value related to the parsed member of the ‘Physics’ Group is generated by the ‘Group Iterator’ BB and passed by the Parameter Links to the ‘Physicalize’ BB through the Parameter Operations of both ‘Get Current’ and ‘Get Name’**(Fig.6-14)**. The 3D object ‘Clamp’ that is a member of the ‘Physics’ Group is physicalized. Finally, this functional feature of physicalizing a fixture element in the pre-defined VR Library of the VFDAS is achieved.

6.3.2 The Functional Objective of Interactive Selection and Manipulation

Another functional objective proposed in the development Phase One of the VFDAS was to develop the function of interactive selection and manipulation for the chosen fixture elements. According to the function of interactive selection and manipulation, the selected fixture elements are interactively manipulated and placed to the proposed location and orientation around the workpiece one by one so as to form the final fixture assembly model. The function of interactive selection and manipulation is provided by the Interactive Design Module in VFDAS system. More details with regard to this functional Module can be found in previous section 5.3.4.

The specific behaviour programming process for this functional objective is presented in the following sections.

6.3.2.1 The Behaviour Programming for the Interactive Selection

In order to achieve the interactive manipulation and assembly with regard to the fixture elements, the functional feature of interactive selection that is used to indicate and select the target fixture element must be developed first.

The behaviour programming content for the functional feature of selecting and indicating the target fixture element is presented in **Fig.6-20**. A set of BBs are chosen and used to this behaviour programming process, which are all assigned to the ‘Level’ to build up the Script named ‘Object Picking’. The BBs applied to this behaviour programming process are the ‘Mouse Waiter’ BB, the ‘2D Picking’ BB, the ‘Identity’ BB, the ‘Show’ BB and the ‘Show Object Information’ BB.

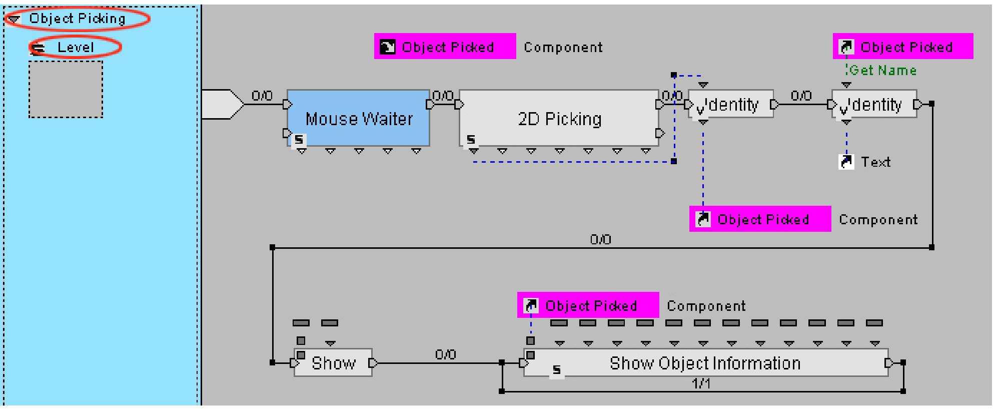


Fig.6-20. The Behaviour Programming for the Interactive Selection

These five BBs are obtained from the BBs standard library according to the directory and then assigned to the Script named ‘Object Picking’, such as: ‘Mouse Waiter’ (Controllers/Mouse), ‘2D Picking’ (Interface/Screen), ‘Identity’ (Logics/Calculator), ‘Show’ (Visuals/Show-Hide), ‘Show Object Information’ (Visuals/Show-Hide). The BB named ‘Mouse Waiter’ is used to trigger different outputs in terms of mouse actions. The BB named ‘2D Picking’ is used to return the 3D Entity, the normal and the exact location of the 2D mouse picking. The BB named ‘Identity’ is used to output the relevant Input Parameters while activated. The BB named ‘Show’ is used to show a 2D/3D Entity, a Mesh or a Group. The BB named ‘Show Object Information’ is used to display the bounding box and the normals of a target 3D Entity. The introductory graphs with regard to these BBs are presented according to the following pictures to clarify more their specifications **(Fig.6-12) (Fig.6-15) (Fig.6-21) (Fig.6-22) (Fig.6-23)**.

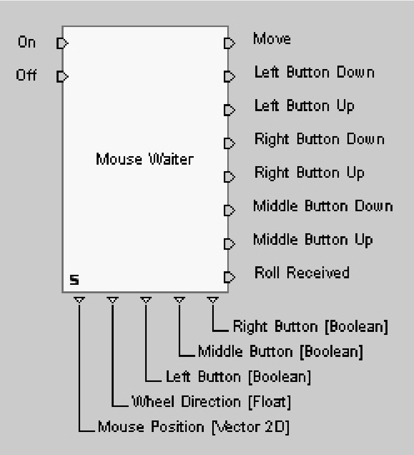


Fig.6-21. The Introductory Graph of the ‘Mouse Waiter’ BB

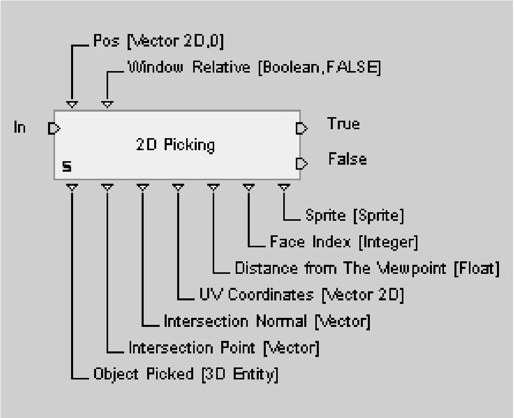

Pos [Vector 2D,0]
Window Relative [Boolean,FALSE]
In
2D Picking
S
True
False
Sprite [Sprite]
Face Index [Integer]
Distance from The Viewpoint [Float]
UV Coordinates [Vector 2D]
Intersection Normal [Vector]
Intersection Point [Vector]
Object Picked [3D Entity]

Fig.6-22. The Introductory Graph of the ‘2D Picking’ BB

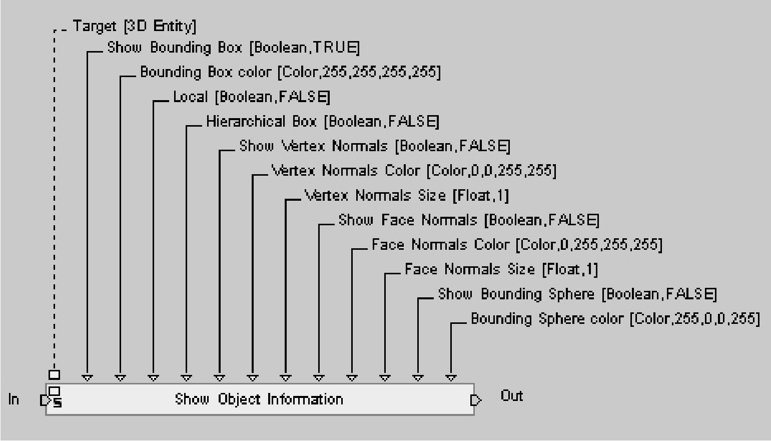

Target [3D Entity]
Show Bounding Box [Boolean,TRUE]
Bounding Box color [Color,255,255,255,255]
Local [Boolean,FALSE]
Hierarchical Box [Boolean,FALSE]
Show Vertex Normals [Boolean,FALSE]
Vertex Normals Color [Color,0,0,255,255]
Vertex Normals Size [Float,1]
Show Face Normals [Boolean,FALSE]
Face Normals Color [Color,0,255,255,255]
Face Normals Size [Float,1]
Show Bounding Sphere [Boolean,FALSE]
Bounding Sphere color [Color,255,0,0,255]
In
S
Show Object Information
Out

Fig.6-23. The Introductory Graph of the ‘Show Object Information’ BB

The logic and correlation of this behaviour programming for the functional feature of interactive selection are as follows. The ‘Mouse Waiter’ BB is used to define the left mouse button as the user input method for interaction **(Fig.6-20)**. Once the left mouse button is released, the activation flow will trigger the ‘2D Picking’ BB that is used to identify the first object in Z Axis order as the selected object according to the picking position of the 2D mouse. As soon as the 3D object is selected by the mouse, the data value related to ‘Object Picked’ will be produced and passed to the first ‘Identity’ BB through the Parameter Link between the ‘2D Picking’ BB and the ‘Identity’ BB **(Fig.6-20)**. Thereafter, the ‘Identity’ BB outputs the Input Parameter related to ‘Object Picked’ value to create a Parameter Output that is specified as a purple Parameter Shortcut called ‘Object Picked’. This purple Parameter Shortcut indicates the selected 3D entity and it will be conveniently applied to the further behaviour programming process for the functional feature of interactive manipulation later on. Once the activation flow activates the second 'Identity' BB, the associated name of the 'Selected 3D Entity' is retrieved and output to another Parameter Shortcut named 'Text' using which the name of the selected 3D object is displayed on the GUI of the VFDAS. Besides, the 'Show' BB is used to ensure that the selected 3D object is visible in the VE. The 'Show Object Information' BB is used to show the bounding box around the selected objected, which is specified as the white color **(Fig.6-24)**.

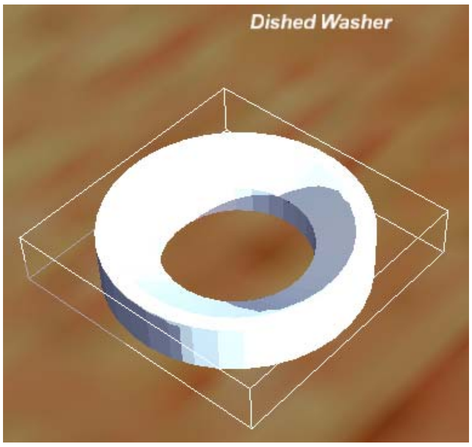


Fig.6-24. The Selected 'Dished Washer' with the White Bounding Box

As soon as the functional feature of interactive selection is achieved, the functional feature of the interactive manipulation and assembly in VFDAS starts to be developed. This functional feature allows that the selected fixture elements are interactively manipulated to the appropriate location and orientation around the target workpiece.

The fixture elements involved for the interactive assembly process are classified into three different physics Modes according to their own physical status e.g. 'Unphysicalized Mode', 'Movable Mode' and 'Accurate Manipulation Mode'. Three different behaviour programming processes were separately applied to the three different Modes. These three behaviour programming processes will be explained in the following sections in terms of each Mode.

6.3.2.2 The Behaviour Programming for the Interactive Manipulation in Unphysicalized Mode

The fixture elements in 'Unphysicalized Mode' do not have any physical property. The specific behaviour programming content for manipulating and locating the selected fixture elements that belong to the 'Unphysicalized Mode' is shown in **Fig.6-25**. Several BBs for this behaviour programming process are applied to the 'Level'. These BBs are the 'Switch On Key' BB, the 'Parameter Selector' BB, the 'Op' BB, the 'Per Second' BB, the 'Translate' BB and the 'Rotate' BB. Besides, two types of Behaviour Graphs that are 'PerSecondTranslate' and 'PerSecondRotate' are also applied to this behaviour programming process. The Behaviour Graph named 'PerSecondTranslate' comprises the 'Op' BB, the 'Per Second' BB and the 'Translate' BB. The Behaviour Graph named 'PerSecondRotate' comprises the 'Op' BB, the 'Per Second' BB and the 'Rotate' BB. The expanded 'PerSecondTranslate' and 'PerSecondRotate' Behaviour Graphs are shown in the following pictures **(Fig.6-26) (Fig.6-27)**.

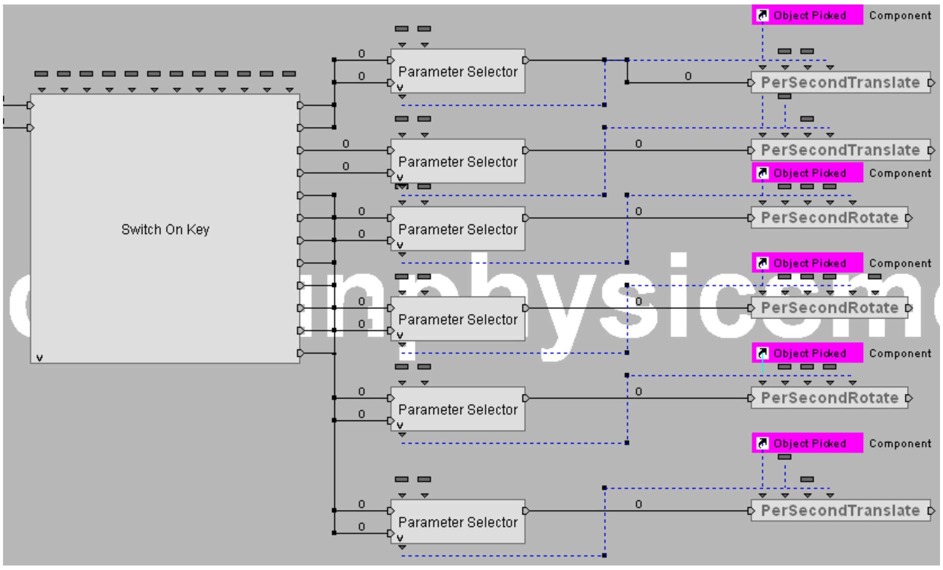


Fig.6-25. The Behaviour Programming for Interactive Manipulation and Assembly in Unphysicalized Mode

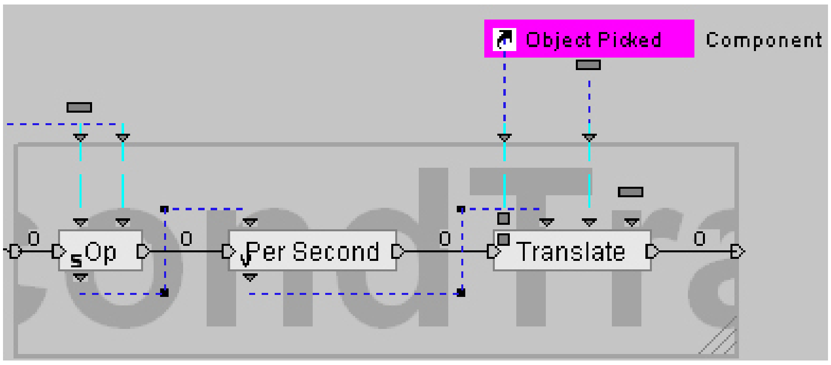


Fig.6-26. The Expanded Behaviour Graph Named 'PerSecondTranslate'

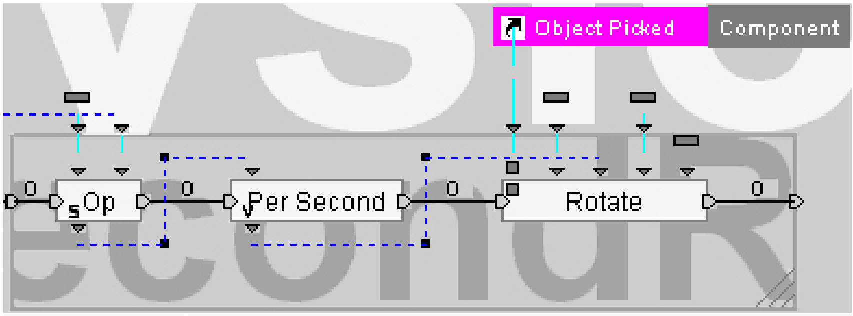


Fig.6-27. The Expanded Behaviour Graph Named 'PerSecondRotate'

The 'Switch On Key' BB is used to trigger the relevant output while receiving a specified Key. The 'Parameter Selector' BB is used to route the corresponding Input Parameter to the Output Parameter according to the Input activated. The 'Op' BB is used to process any valid Parameter Operation. The 'Per Second' BB is used to calculate a progression (Y) according to a given velocity (X): Y=X*Elapsed Time during one Frame. X can be any type of parameter as long as this parameter is only made of float values or derived type. The 'Translate' BB is used to translate a target 3D Entity. The 'Rotate' BB is used to rotate the target 3D Entity. To clarify more their specifications, the introductory graphs with regard to these six BBs are presented according to the following pictures below **(Fig.6-28) (Fig.6-29) (Fig.6-30) (Fig.6-31) (Fig.6-32) (Fig.6-33)**.

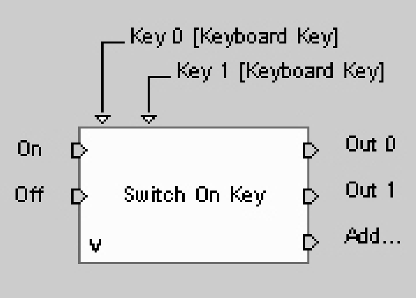


Fig.6-28. The Introductory Graph of the 'Switch On Key' BB

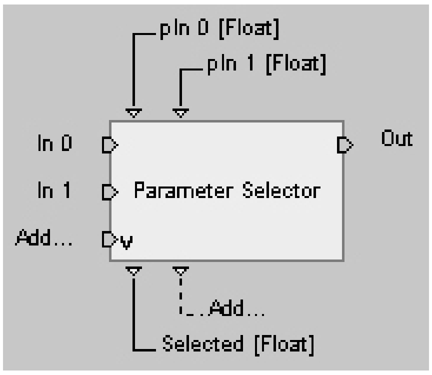


Fig.6-29. The Introductory Graph of the ‘Parameter Selector’ BB

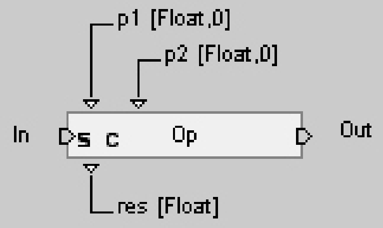


Fig.6-30. The Introductory Graph of the ‘Op’ BB

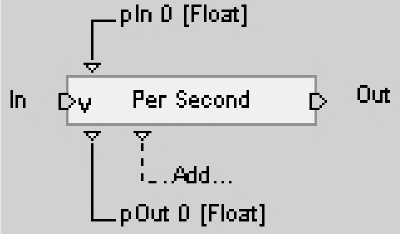


Fig.6-31. The Introductory Graph of the ‘Per Second’ BB

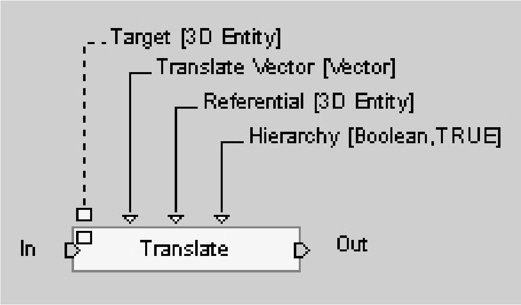


Fig.6-32. The Introductory Graph of the ‘Translate’ BB

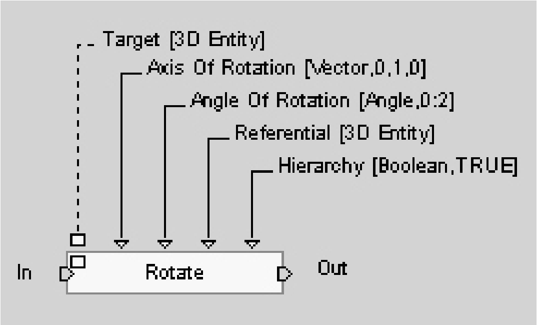
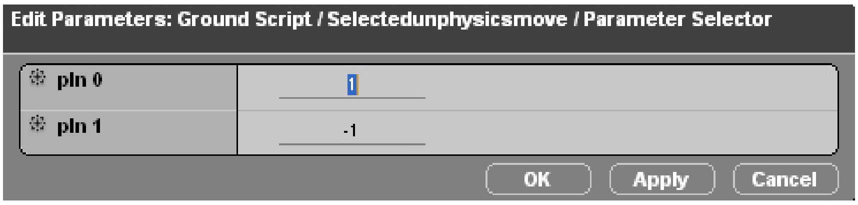


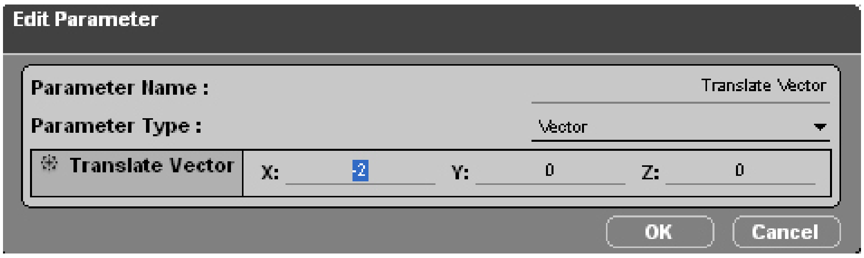
Fig.6-33. The Introductory Graph of the 'Rotate' BB

The logic and correlation of this behaviour programming for the functional feature of the interactive manipulation in Unphysicalized Mode are as follows. The 'Switch On Key' BB is employed to specify the keys of W, A, S, D, Z, X on the keyboard for the interactive translation of the selected 3D object as well as the keys of 1, 2, 3, 4, 5, 6 for the interactive rotation of the selected 3D object. Furthermore, the purple Parameter Shortcut 'Object Picked' here specifies the selected 3D Entity and has been introduced in previous section 6.3.2.1 regarding the function of interactive selection. Each 'Parameter Selector' BB provides two Float types of Parameters that are respectively defined as 1 and -1 **(Fig.6-34)**. These two Parameters are used to determine the direction of translation or Rotation for the selected 3D objects i.e. positive or negative; clockwise or anti-clockwise. In terms of Behaviour Graph 'PerSecondTranslate', the value of translation vector per second is specified by the Parameter 'Translate Vector' **(Fig.6-35)** associated with the 'Op' BB. In terms of 'PerSecondRotate' Behaviour Graph, the value of rotation vector per second is specified by the Parameter 'Angle Per Second' **(Fig.6-36)** associated with the 'Op' BB. The rotation axis for the selected object is specified by the Parameter named 'Axis of Rotation' **(Fig.6-37)** associated with the 'Rotate' BB.


Edit Parameters: Ground Script / Selectedunphysicsmove / Parameter Selector

pIn 0	1
pIn 1	-1

Fig.6-34. The Two Parameters of 'Parameter Selector' BB



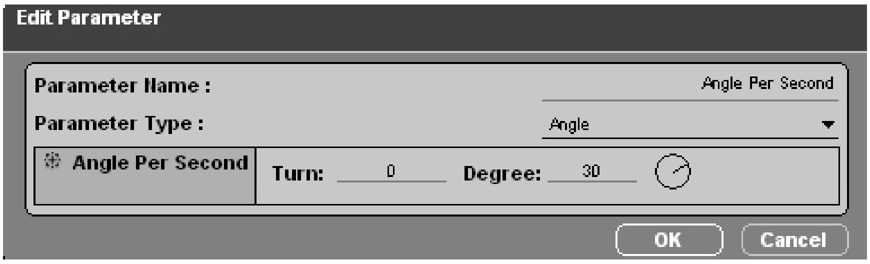
Edit Parameter

Parameter Name : Translate Vector

Parameter Type : Vector

Translate Vector X: -2 Y: 0 Z: 0

Fig.6-35. The Parameter named ‘Translate Vector’ of ‘Op’ BB



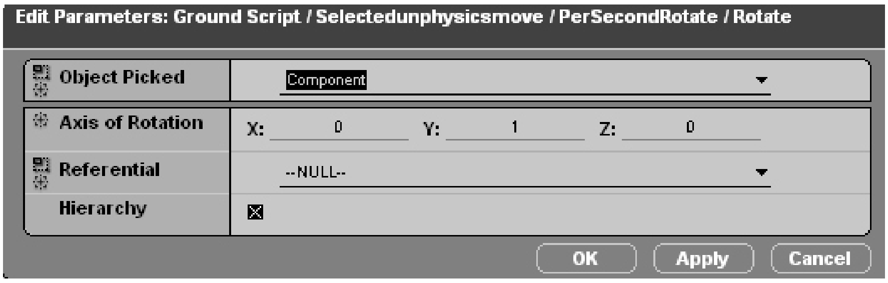
Edit Parameter

Parameter Name : Angle Per Second

Parameter Type : Angle

Angle Per Second Turn: 0 Degree: 30

Fig.6-36. The Parameter named ‘Angle Per Second’ of ‘Op’ BB



Edit Parameters: Ground Script / Selectedunphysicsmove / PerSecondRotate / Rotate

Object Picked: Component

Axis of Rotation X: 0 Y: 1 Z: 0

Referential: --NULL--

Hierarchy ☒

Fig.6-37. The Parameter named ‘Axis of Rotation’ in ‘Rotate’ BB

6.3.2.3 The Behaviour Programming for the Interactive Manipulation in ‘Movable Mode’

The fixture elements in ‘Movable Mode’ are associated with physical properties. The behaviour programming content described in **Fig.6-25** related to ‘Unphysicalized Mode’ can not realize the functional feature of interactive manipulation with regard to the fixture elements that belong to the ‘Movable Mode’.

Consequently, the behaviour programming content for manipulating and locating the selected fixture elements that belong to the ‘Movable Mode’ is shown in **Fig.6-38**. Several BBs for this behaviour programming process are assigned to the ‘Level’. These BBs are the ‘Switch On Key’ BB, the ‘Physics Impulse’ BB and the ‘Physics Torque’ BB. The ‘Switch On Key’ BB is used to activate the corresponding output while receiving a specified key. The ‘Physics Impulse’ BB is used to generate an impulse or apply a constant force to the physics object in a VE. The ‘Physics Torque’ BB is used to produce a rotation impulse or apply a constant rotation impulse to the physics object. The introductory graphs related to these BBs are presented according to the following pictures to clarify more their specifications **(Fig.6-28) (Fig.6-39) (Fig.6-40)**.

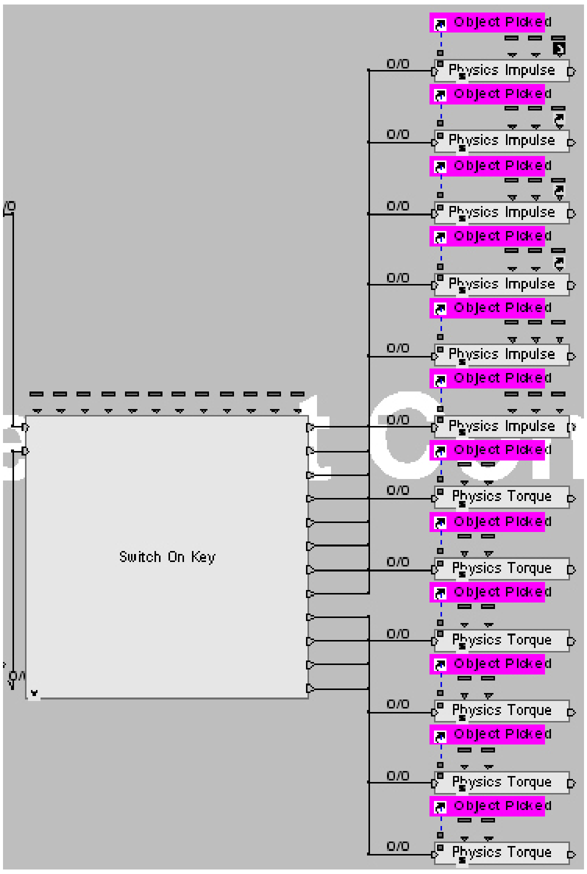


Fig.6-38. The Behaviour Programming for Interactive Manipulation and Assembly in Movable Mode

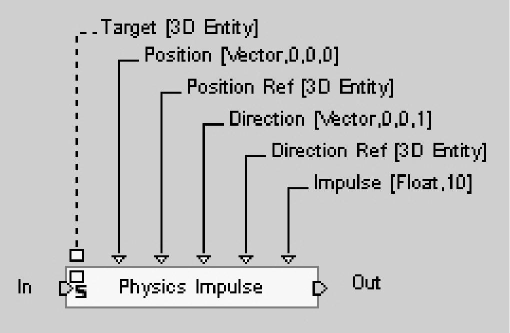


Fig.6-39. The Introductory Graph of the ‘Physics Impulse’ BB

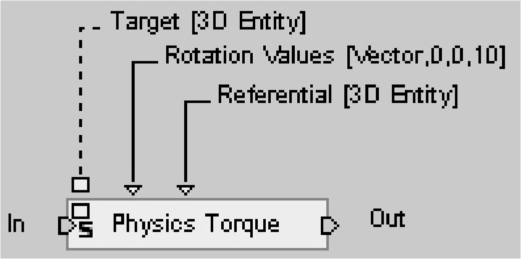
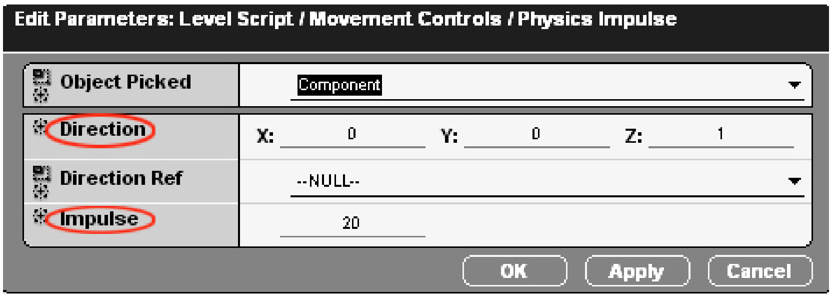


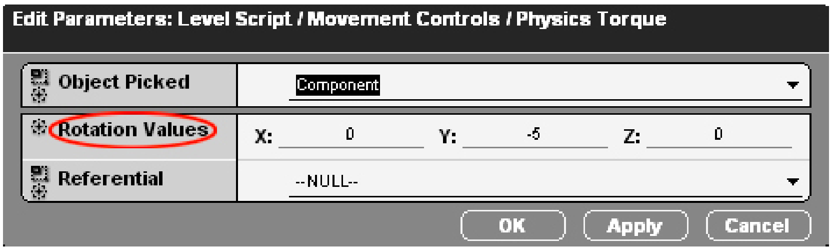
Fig.6-40. The Introductory Graph of the 'Physics Torque' BB

The logic and correlation of this behaviour programming for the functional feature of the interactive manipulation in Movable Mode are as follows. The 'Switch On Key' BB is used to specify the keys of W, A, S, D, Z, X for the interactive translation of the selected 3D object as well as the keys of 1, 2, 3, 4, 5, 6 for interactive rotation of the selected 3D object. The key settings are same as the interactive manipulation in 'Unphysicalized Mode'. This means that both unphysicalized and physical fixture elements are interactively manipulated using the same keys on the keyboard in spite of their physical status. The purple Parameter Shortcut called 'Object Picked' in the Script **(Fig.6-38)** determines the selected 3D Entity, which is defined by the function of interactive selection addressed in section 6.3.2.1. The fixture elements that belong to 'Movable Mode' are associated with physical properties as a part of the physics world. The method of manipulating them is using an impulse or constant force upon them to change their spatial position and orientation. The magnitude and direction of the force applied to the fixture elements in Movable Mode are specified by the Parameters associated with 'Physics Impulse' BB **(Fig.6-41)**. Besides, the magnitude and rotating axis of a rotation force applied to the fixture elements in Movable Mode are specified by the Parameters associated with 'Physics Torque' BB **(Fig.6-42)**.


Edit Parameters: Level Script / Movement Controls / Physics Impulse

Object Picked	Component
Direction	X: 0 Y: 0 Z: 1
Direction Ref	--NULL--
Impulse	20

Fig.6-41. The Parameters Associated with 'Physics Impulse' BB



Edit Parameters: Level Script / Movement Controls / Physics Torque

Object Picked: Component

Rotation Values X: 0 Y: -5 Z: 0

Referential: --NULL--

Fig.6-42. The Parameters Associated with 'Physics Torque' BB

6.3.2.4 The Behaviour Programming for the Interactive Manipulation in 'Accurate Manipulation Mode'

To accurately manipulate the fixture elements with physical properties in a fixture design process, the selected fixture elements need to be assigned to the 'Accurate Manipulation Mode'. Once the selected fixture elements are assigned to the 'Accurate Manipulation Mode', they can be accurately manipulated to the proposed position and orientation by using the same key settings of W, A, S, D, Z, X for the interactive translation and 1, 2, 3, 4, 5, 6 for interactive rotation.

The behaviour programming content for manipulating and locating the selected fixture elements that belongs to the 'Accurate Manipulation Mode' is presented in **Fig.6-43**. This behaviour programming content is similar to the Script described in **Fig.6-25.** The main difference is that the target 3D Entity is a 3D Frame named 'Master Frame For Red Player' instead of the purple Parameter Shortcut 'Object Picked' in **Fig.6-25**. This 3D Frame is used to control a physics object.

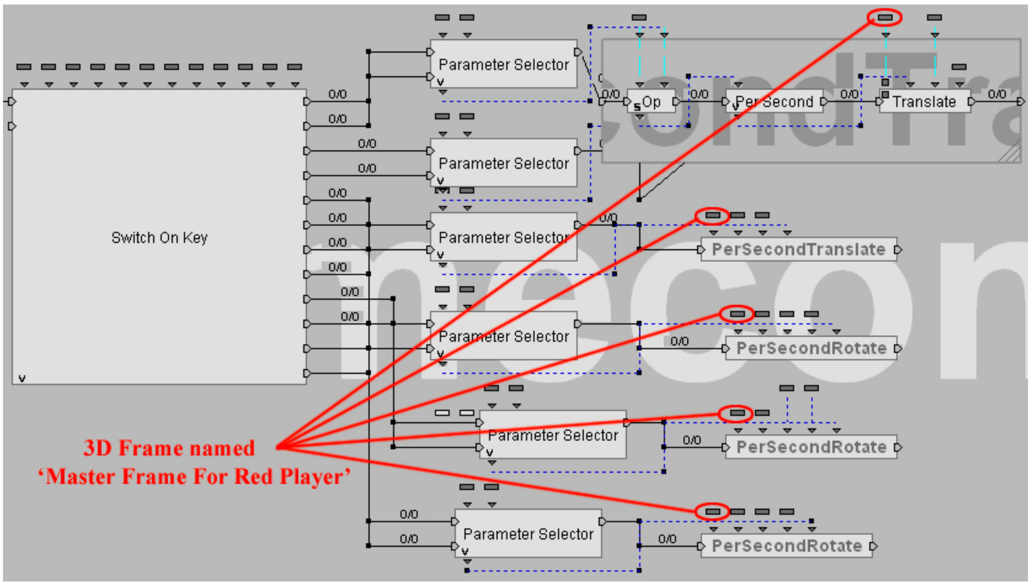
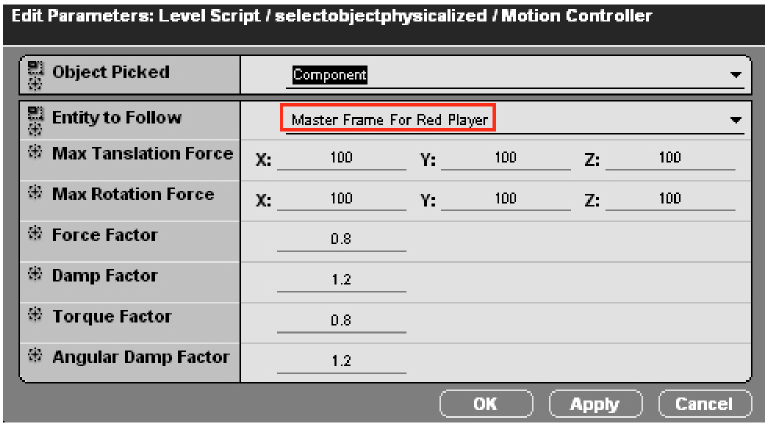


Fig.6-43. The Behaviour Programming for Interactive Manipulation and Assembly in Accurate Manipulation Mode

To manipulate a physics object in accuracy, another BB named 'Motion Controller' is applied to this behaviour programming process and assigned to the 3D Frame 'Master Frame For Red Player' **(Fig.6-44)**. The 'Motion Controller' BB is used to control a physics object by making it follow another 3D Entity. Using this BB, the fixture elements in 'Accurate Manipulation Mode' can follow the specified 3D Frame named 'Master Frame For Red Player' so that the process of interactive manipulation in 'Accurate Manipulation Mode' is actually carried out by controlling the 3D Frame 'Master Frame For Red Player'. The introductory graph with regard to the 'Motion Controller' BB is presented in **Fig.6-45** to state its specification.


Edit Parameters: Level Script / selectobjectphysicalized / Motion Controller

Object Picked	Component					
Entity to Follow	Master Frame For Red Player					
Max Tanslation Force	X:	100	Y:	100	Z:	100
Max Rotation Force	X:	100	Y:	100	Z:	100
Force Factor	0.8					
Damp Factor	1.2					
Torque Factor	0.8					
Angular Damp Factor	1.2					

Fig.6-44. The Parameters of the 'Motion Controller' BB

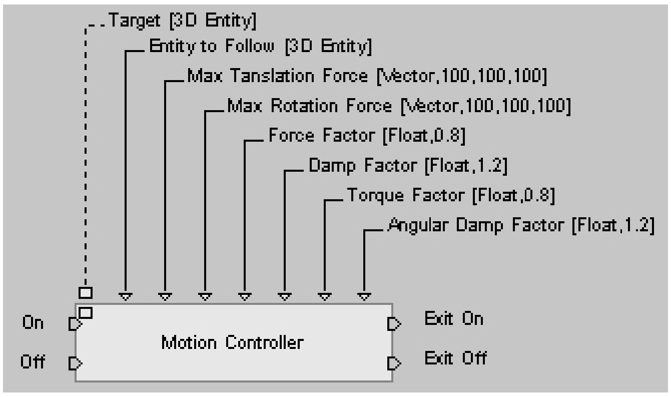

Target [3D Entity]
Entity to Follow [3D Entity]
Max Tanslation Force [Vector,100,100,100]
Max Rotation Force [Vector,100,100,100]
Force Factor [Float,0.8]
Damp Factor [Float,1.2]
Torque Factor [Float,0.8]
Angular Damp Factor [Float,1.2]
On
Off
Motion Controller
Exit On
Exit Off

Fig.6-45. The Introductory Graph of the 'Motion Controller' BB

6.3.3 The Functional Objective of Assigning and Changing the Physics Modes

Another functional objective required in Phase One was to develop the function of assigning and changing the different Physics Modes for the selected fixture elements. These Physics Modes in VFDAS are 'Unmovable Mode', 'Movable Mode', 'Accurate Manipulation Mode' and 'Unphysicalized Mode'. Their functions are sequentially provided by the 'Unmovable Utility', 'Movable Utility', 'Accurate Manipulation Utility' and 'Unphysicalized Utility' in Physicalization Module of the VFDAS. More details with regard to the Physicalization Module can be found in previous section 5.3.3.

To assign and change the different Physics Modes for the selected fixture elements, the behaviour programming process applied to the four different utilities is explained in the following sections.

6.3.3.1 The Behaviour Programming for the Unmovable Utility

Once fixture designers interactively manipulate the selected fixture element to the final location and orientation, the Unmovable Utility is employed to fasten and immobilize the selected fixture element. This fastening task is fulfilled by using the key 7 on the keyboard. The fastened fixture element in 'Unmovable Mode' will also be a fixed physics 3D object without the movement that have the real-time 3D collision detection. In addition, the fixture elements that are assigned in 'Unmovable Mode' are still considered as a part of the physics world but gravity and force can not have any effect on them.

The behaviour programming content for fastening the selected fixture element using Unmovable Utility is presented in the following picture **(Fig.6-46)**. A group of BBs and Parameter Operations that are assigned to the 'Level' are applied to this behaviour programming process. They are the 'Key Waiter' BB, the 'Is In Group' BB, the 'Test' BB, the 'Remove From Group' BB, the 'UnPhysicalize' BB, the 'Identity' BB, the 'Show' BB, the 'Add To Group' BB, the 'Group Iterator' BB, the 'Physicalize' BB, the Parameter Operation called 'Get Current' and the Parameter Operation called 'Get Name'. Moreover, a Behaviour Graph 'Parts Collision' is also applied to this behaviour programming. This Behaviour Graph 'Parts Collision' comprises the 'Show' BB, the 'Add To Group' BB, the 'Group Iterator' BB, the 'Physicalize' BB, the Parameter Operation 'Get Current' and the Parameter Operation 'Get Name' **(Fig.6-13) (Fig.6-14)**.

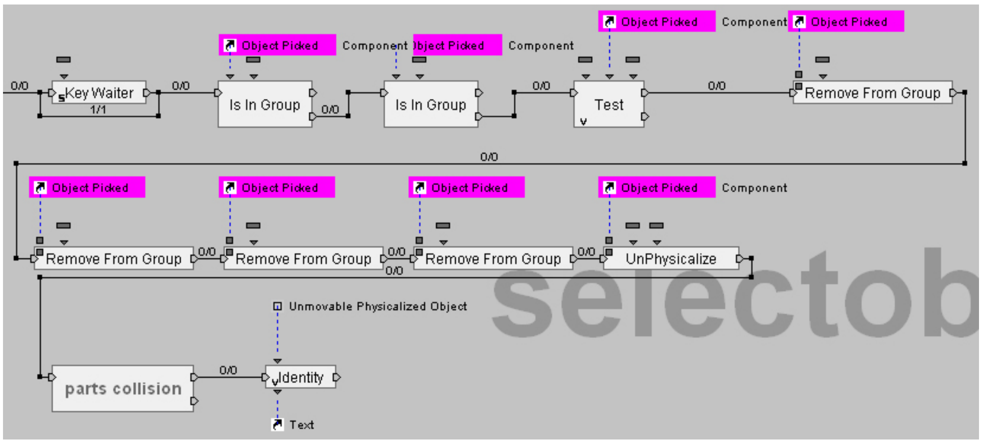


Fig.6-46. The Behaviour Programming for the Unmovable Utility

The 'Key Waiter' BB is used to wait for a key to be pressed. The 'Is In Group' BB is used to check if the specific object is in a Group. The 'Test' BB is used to generate the appropriate output according to the test between A and B. The 'Remove From Group' BB is used to remove the target object from a Group. The 'UnPhysicalize' BB is used to remove a physics object from the physics world. The 'Identity' BB is used to output the input parameters while it is activated. The 'Show' BB is used to show a 2D/3D Entity, a Mesh or a Group to ensure that the target object is visible in the VE. The 'Add To Group' BB is used to add the object to an arranged Group. The 'Group Iterator' BB is used to retrieve each element of a Group. The 'Physicalize' BB is used to define a 3D object becoming a part of the physics world. More details in relation to the 'Parts Collision' Behaviour Graph can be seen in previous section 6.3.1.4. The introductory graphs with regard to these BBs are presented according to the following pictures to clarify more their specifications **(Fig.6-12) (Fig.6-15) (Fig.6-16) (Fig.6-17) (Fig.6-18) (Fig.6-47) (Fig.6-48) (Fig.6-49) (Fig.6-50) (Fig.6-51)**.

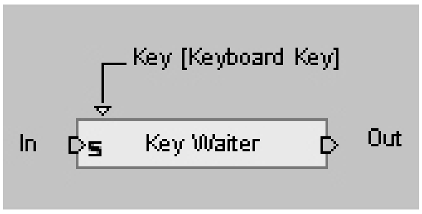


Fig.6-47. The Introductory Graph of the 'Key Waiter' BB

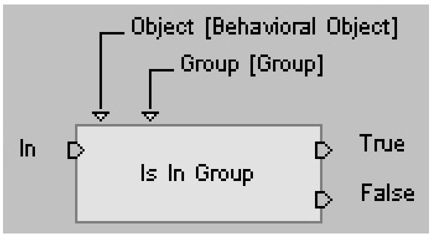


Fig.6-48. The Introductory Graph of the 'Is In Group' BB

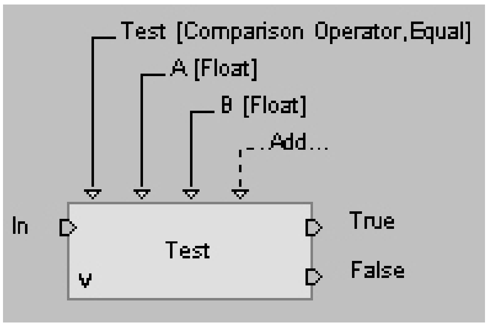


Fig.6-49. The Introductory Graph of the 'Test' BB

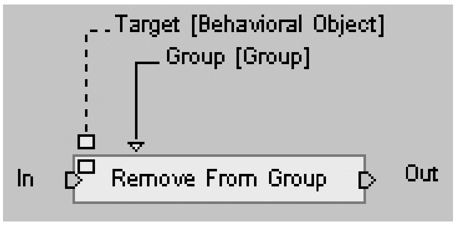


Fig.6-50. The Introductory Graph of the 'Remove From Group' BB

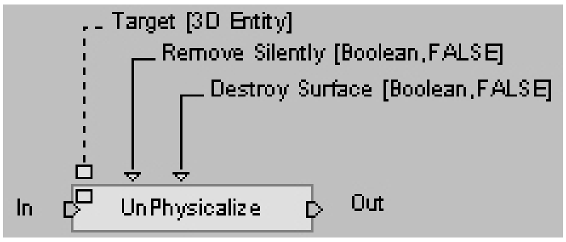
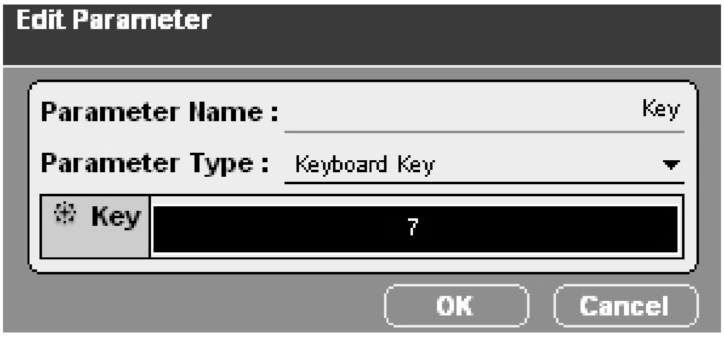

Target [3D Entity]
Remove Silently [Boolean,FALSE]
Destroy Surface [Boolean,FALSE]
In
UnPhysicalize
Out

Fig.6-51. The Introductory Graph of the 'UnPhysicalize' BB

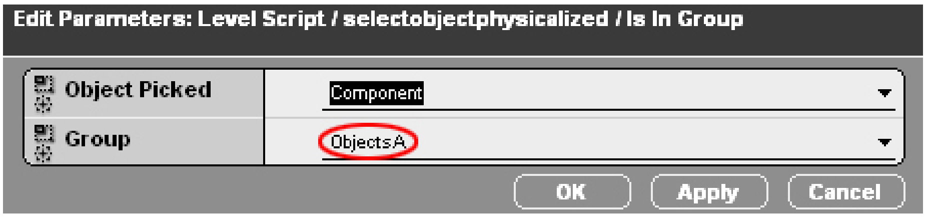
The logic and correlation of this behaviour programming for the function of the Unmovable Utility are as follows. The 'Key Waiter' BB is used to specify the computer key 7 to wait for it being pressed **(Fig.6-52)**. As soon as the key 7 is pressed by the users, the activation flow will trigger the next BB named 'Is In Group'. The 'Is In Group' BB is used to check if the selected object is allocated in the Group called 'ObjectsA' **(Fig.6-53).** This Group 'ObjectsA' comprises a set of 3D objects that do not need to be assigned with physical properties in order to prevent physicalization errors for improper 3D objects. Once the selected object is tested not to be in the 'ObjectsA' Group, the Behaviour Output called 'False' of 'Is In Group' BB will pass an activation flow to the next BB. The next BB is also named 'Is In Group' which is used to check another Group called 'ObjectsB'. The 'ObjectsB' Group comprises another set of 3D objects that also do not need to be assigned with a physical property. Once the selected object is tested not to be in 'ObjectsB' Group, the Behaviour Output 'False' of this BB will pass the activation flow to the next BB named 'Test'. The 'Test' BB is used to test between the selected object and the 3D object named 'Ground' according to the Comparison Operation named 'Not Equal' **(Fig.6-54)** so as to prevent accidentally changing the Physics Mode of the 'Ground' object. Once the selected object is tested not to be the 'Ground' Object, the activation flow will activate the next BB named 'Remove From Group'. These next four 'Remove From Group' BBs are used to remove the selected 3D object from the four different Groups so as to avoid further physicalization errors. Thereafter, the 'UnPhysicalize' BB is employed to unphysicalize the selected 3D object which is expressed by a purple Parameter Shortcut 'Object Picked'. The 'Parts Collision' Behaviour Graph is used to re-assign the physical properties to the selected 3D object according to the physicalization requirement of a fixed physical object **(Fig.6-55)**. The 'Identity' BB is used to display a confirmation message on the GUI of the VFDAS in order to notify the users that the selected 3D object is already assigned to the 'Unmovable Mode'.



Edit Parameter

Parameter Name :	Key
Parameter Type :	Keyboard Key
Key	7

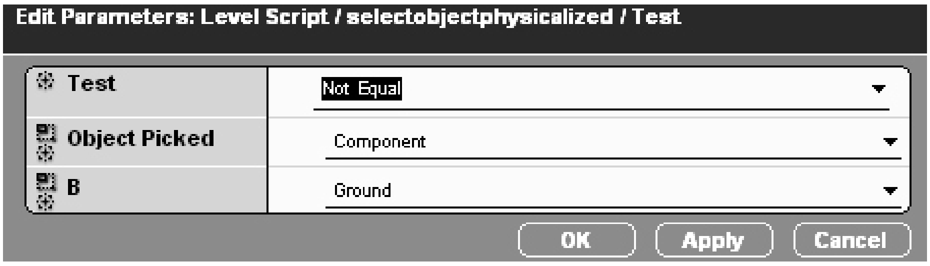
Fig.6-52. The Key 7 Specified for the ‘Key Waiter’ BB



Edit Parameters: Level Script / selectobjectphysicalized / Is In Group

Object Picked	Component
Group	ObjectsA

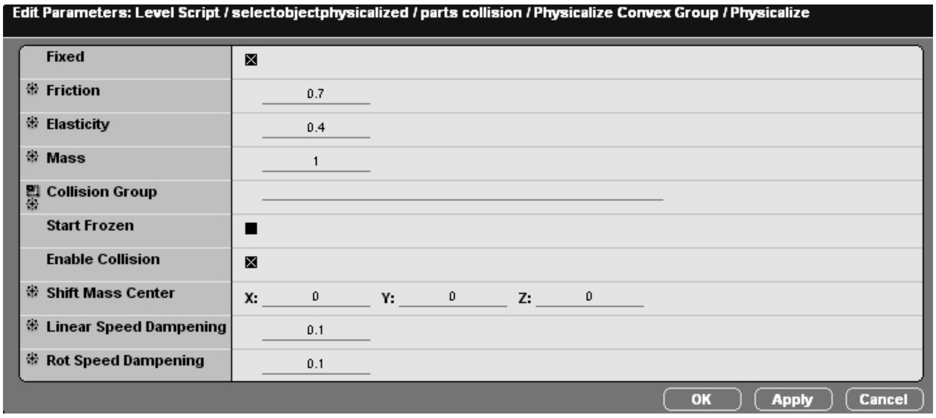
Fig.6-53. The Group Parameter Called ‘ObjectsA’ in the First ‘Is In Group’ BB



Edit Parameters: Level Script / selectobjectphysicalized / Test

Test	Not Equal
Object Picked	Component
B	Ground

Fig.6-54. The Parameters Specified in the ‘Test’ BB



Edit Parameters: Level Script / selectobjectphysicalized / parts collision / Physicalize Convex Group / Physicalize

Fixed	☒
Friction	0.7
Elasticity	0.4
Mass	1
Collision Group	
Start Frozen	■
Enable Collision	☒
Shift Mass Center	X: 0 Y: 0 Z: 0
Linear Speed Dampening	0.1
Rot Speed Dampening	0.1

Fig.6-55. The Parameters Specified in the ‘Physicalize’ BB

6.3.3.2 The Behaviour Programming for the Movable Utility

The Movable Utility is employed to re-assign physical properties to the selected fixture elements, e.g. mass, gravity, friction, elasticity, exact collision detection etc. Thus, the immobilized fixture elements are unfixed by using key 8 on the keyboard. They can then be manipulated by users. The fixture elements in ‘Movable Mode’ have physical properties, e.g. real-time 3D collision detection. They are considered as a part of the physics world so that the method of controlling and manipulating them is applying an impulse or constant force on them to adjust their spatial position and orientation in a VE. The gravity and force take an effect on them in the physics world.

The behaviour programming content for making the immobilized fixture elements manipulable using the Movable Utility is presented in **Fig.6-56**. A few BBs that are assigned to the ‘Level’ are applied to this behaviour programming process. These BBs and Parameter Operations are similar to the behaviour programming applied to the Unmovable Utility according to the picture in **Fig.6-46** except that a few Parameters specified in the BBs are different. They are the ‘Key Waiter’ BB, the ‘Is In Group’ BB, the ‘Test’ BB, the ‘Remove From Group’ BB, the ‘UnPhysicalize’ BB, the ‘Identity’ BB, the ‘Show’ BB, the ‘Add To Group’ BB, the ‘Group Iterator’ BB, the ‘Physicalize’ BB, the Parameter Operation ‘Get Current’ and the Parameter Operation ‘Get Name’. A Behaviour Graph named ‘Parts Collision’ is also applied to this behaviour programming. More details with regard to these BBs and the Behaviour Graph ‘Parts Collision’ can be found in previous section 6.3.3.1.

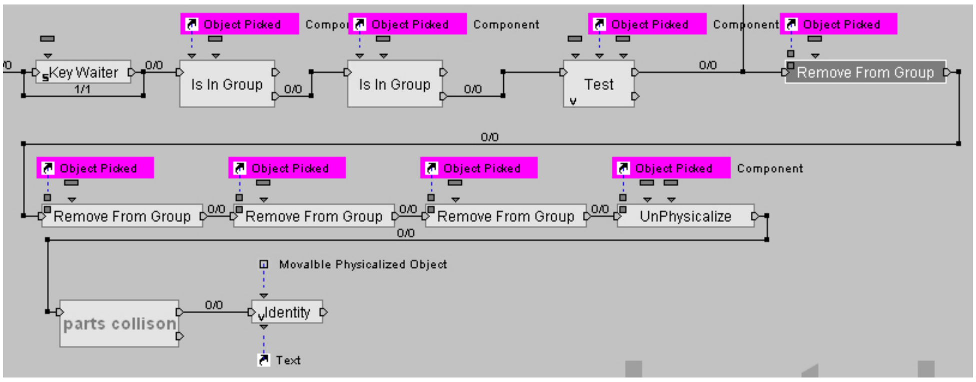
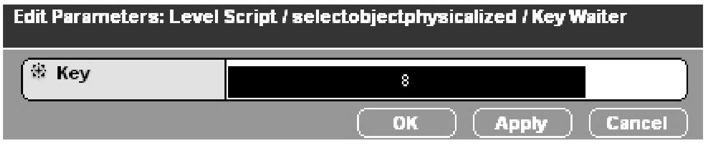


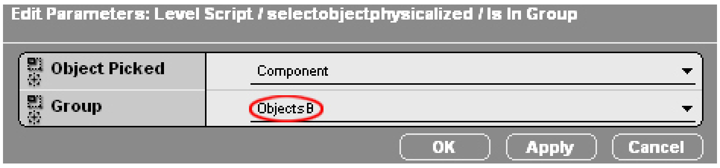
Fig.6-56. The Behaviour Programming for the Movable Utility

The logic and correlation of this behaviour programming for the function of the Movable Utility in VFDAS system are as follows. The 'Key Waiter' BB is used to specify key 8 to wait for it being pressed **(Fig.6-57)**. The activation flow will trigger the next BB named 'Is In Group' once the key 8 is pressed by the users. The 'Is In Group' BB is used to check if the selected object is in the Group called 'ObjectsA' which comprises 3D objects that do not need to be assigned with a physical property. Once the selected object is tested not to be in the 'ObjectsA' Group, the Behaviour Output 'False' of 'Is In Group' BB will then pass an activation flow to the next BB. The next BB is also named 'Is In Group' which is used to check another Group called 'ObjectsB' **(Fig.6-58)**. The 'ObjectsB' Group comprises another set of 3D objects that do not need to be assigned with a physical property. Once the selected object is tested not to be in 'ObjectsB' Group, the Behaviour Output 'False' of this BB will pass the activation flow to the next BB named 'Test'. The 'Test' BB is used to test between the selected object and the 3D object named 'Ground' according to the Comparison Operation named 'Not Equal' **(Fig.6-54)** so as to avoid accidentally changing the Physics Mode of the 'Ground' object. As soon as the selected object is tested not to be the 'Ground' 3D object, the activation flow will activate next BB named 'Remove From Group'. These next four BBs named 'Remove From Group' are used to remove the selected 3D object from the four different Groups in order to avoid physicalization errors. Thereafter, the 'UnPhysicalize' BB is used to unphysicalize the selected 3D object which is indicated by a purple Parameter Shortcut called 'Object Picked'. The 'Parts Collision' Behaviour Graph is used to re-assign the physical properties to the selected 3D object according to the physicalization requirement of a movable physical object **(Fig.6-59)**. The 'Identity' BB is used to show a confirmation message on the GUI of the VFDAS to notify the users that the selected 3D object is already changed to the 'Movable Mode'.


Edit Parameters: Level Script / selectobjectphysicalized / Key Waiter

Key: 8

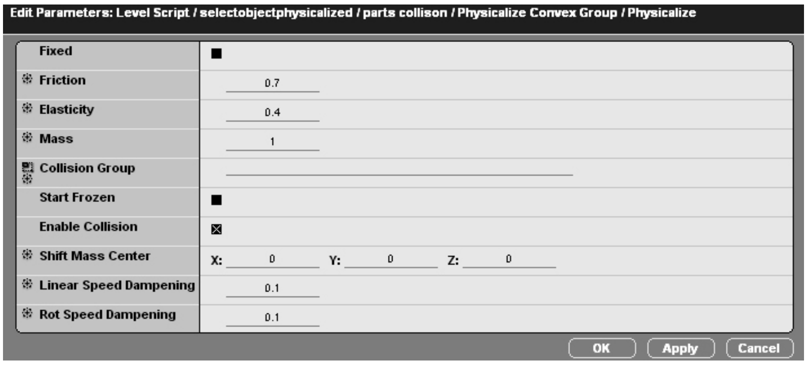
Fig.6-57. The Key 8 Specified for the 'Key Waiter' BB



Edit Parameters: Level Script / selectobjectphysicalized / Is In Group

Object Picked	Component
Group	ObjectsB

Fig.6-58. The Group Parameter Called ‘ObjectsB’ in the Second ‘Is In Group’ BB



Edit Parameters: Level Script / selectobjectphysicalized / parts collison / Physicalize Convex Group / Physicalize

Fixed	■
Friction	0.7
Elasticity	0.4
Mass	1
Collision Group	
Start Frozen	■
Enable Collision	☒
Shift Mass Center	X: 0 Y: 0 Z: 0
Linear Speed Dampening	0.1
Rot Speed Dampening	0.1

Fig.6-59. The Parameters Specified in the ‘Physicalize’ BB

6.3.3.3 The Behaviour Programming for the Accurate Manipulation Utility

To accurately manipulate the physicalized 3D objects in VFDAS, the Accurate Manipulation Utility was developed to carry out the accurate fixture assembly process. Using the Accurate Manipulation Utility, the selected fixture elements are accurately manipulated to the proposed location and orientation even though they are associated with physical properties. The fixture elements in ‘Accurate Manipulation Mode’ have essential physical properties e.g. real-time 3D collision detection. Although they are considered as a part of the physics world, these physical fixture elements are made following with one invisible 3D Frame so that the whole process of interactive manipulation and assembly in ‘Accurate Manipulation Mode’ is actually carried out by controlling the proposed 3D Frame named ‘Master Frame For Red Player’. The selected fixture element is assigned to ‘Accurate Manipulation Mode’ using the key 9 on the keyboard.

The behaviour programming content for assigning the selected fixture elements to ‘Accurate Manipulation Mode’ by using the Accurate Manipulation Utility is presented in **Fig.6-60**. A set of BBs that are assigned to the ‘Level’ are applied to this behaviour programming process. They are the ‘Key Waiter’ BB, the ‘Is In Group’ BB, the ‘Test’ BB, the ‘Remove From Group’ BB, the ‘UnPhysicalize’ BB, the ‘Motion Controller’

BB, the 'Broadcast Message' BB, the 'Identity' BB, the 'Show' BB, the 'Add To Group' BB, the 'Group Iterator' BB, the 'Physicalize' BB, the Parameter Operation called 'Get Current' and the Parameter Operation called 'Get Name'. A Behaviour Graph 'Parts Collision' is also applied to this behaviour programming. In particular, The 'Motion Controller' BB is used to control a physics object by making it follow another 3D Entity. The 'Broadcast Message' BB is used to propagate a Message to every object that is waiting for it. The introductory graphs with regard to these BBs are presented according to the pictures to clarify their specifications **(Fig.6-47) (Fig.6-48) (Fig.6-49) (Fig.6-50) (Fig.6-51) (Fig.6-45) (Fig.6-12) (Fig.6-15) (Fig.6-16) (Fig.6-17) (Fig.6-18) (Fig.6-61)**.

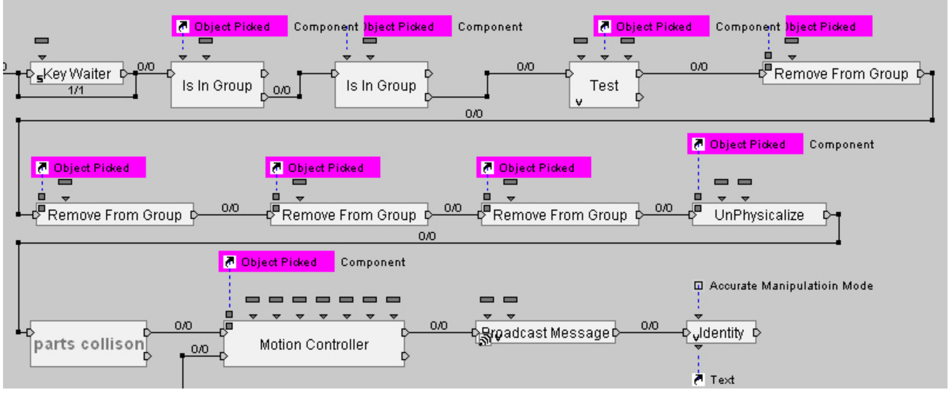


Fig.6-60. The Behaviour Programming for the Accurate Manipulation Utility

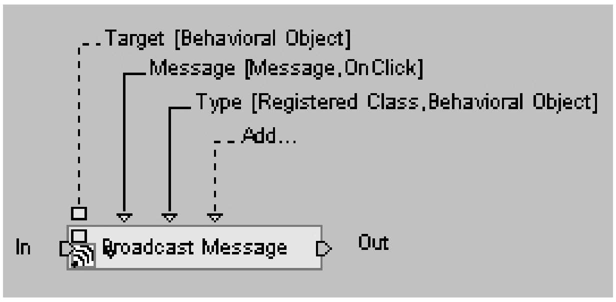
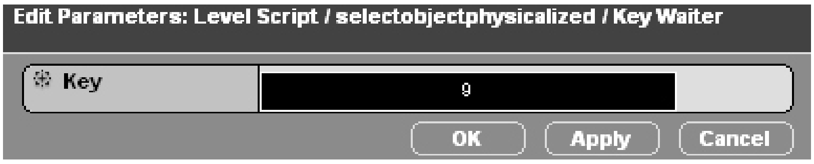

Target [Behavioral Object]
Message [Message,OnClick]
Type [Registered Class,Behavioral Object]
Add...
In
Broadcast Message
Out

Fig.6-61. The Introductory Graph of the 'Broadcast Message' BB

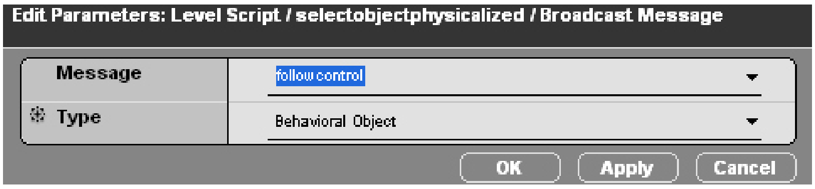
The logic and correlation of this behaviour programming for the function of Accurate Manipulation Utility in the VFDAS are similar to the behaviour programming applied to the Movable Utility addressed in 6.3.3.2. The difference will be explained as follows. The 'Key Waiter' BB is used to specify key 9 to wait for it being pressed **(Fig.6-62)**. The 'Physicalize' BB in the 'Part collision' Behaviour Graph requires to be specified according to the picture shown in **Fig.6-59**. Using the 'Motion Controller' BB, the physicalized fixture elements follow the specified 3D Frame named 'Master Frame For Red Player' **(Fig.6-44)**. The 'Broadcast Message' BB is used to send a Message called 'FollowControl' **(Fig.6-63)** to activate the behaviour programming Script in **Fig.6-43** for the interactive manipulation in 'Accurate Manipulation Mode' and deactivate the Script in **Fig.6-25** for the interactive manipulation in 'Unphysicalized Mode'.



Edit Parameters: Level Script / selectobjectphysicalized / Key Waiter

Key: 9

Fig.6-62. The Key 9 Specified for the 'Key Waiter' BB



Edit Parameters: Level Script / selectobjectphysicalized / Broadcast Message

Message: follow control

Type: Behavioral Object

Fig.6-63. The Sent Message Named 'FollowControl'

6.3.3.4 The Behaviour Programming for the Unphysicalized Utility

The Unphysicalized Utility is used to unphysicalize the selected fixture elements by using key 0 on the keyboard so that they no longer have any physical property. Thus, they are allowed to spatially penetrate and intersect with other physical 3D objects in VFDAS without the collision detection because sometimes the physical properties do not need to be engaged with a virtual fixture design process. The fixture elements in 'Unphysicalized Mode' do not have any physical property but they can still be manipulated by the users to the proposed location and orientation in the VE so as to fulfil an unphysicalized fixture assembly process.

The behaviour programming content for unphysicalizing the selected fixture elements by using Unphysicalized Utility is presented in **Fig.6-64**. Several BBs that are assigned to the 'Level' are applied to this behaviour programming process. They are the 'Key Waiter' BB, the 'Is In Group' BB, the 'Test' BB, the 'Remove From Group' BB, the 'UnPhysicalize' BB, the 'Broadcast Message' BB and the 'Identity' BB. The introductory graphs with regard to these BBs are presented according to the pictures to clarify their specifications **(Fig.6-47) (Fig.6-48) (Fig.6-49) (Fig.6-50) (Fig.6-51) (Fig.6-61) (Fig.6-12)**.

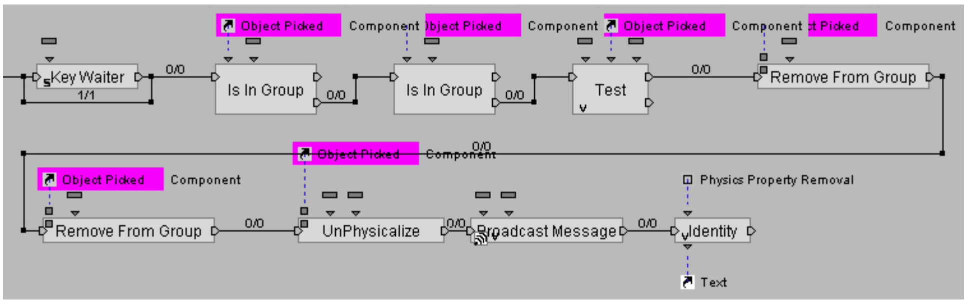


Fig.6-64. The Behaviour Programming for the Unphysicalized Utility

6.4 System Development Phase Two

After three functional objectives for the VFDAS framework are achieved at the development Phase One, the development Phase Two of the VFDAS system is then carried out. The development Phase Two focused on examination of 3D information lost and retrieved during the integration process between the VFDAS and CAD systems. The evaluation of the VFDAS output that is returned to the CAD system was also performed.

The interactive fixture design process is fulfilled by using the various functions of the VFDAS system. The final fixture assembly model developed in VFDAS system is externally exported by the Design Configuration Output Module to other CAD software e.g. Pro/ENGINEER because the CAD software is widely used in industry as the standard tool. Thereafter, the manufacturing operations are carried out.

In order to create the integration between the VFDAS and CAD systems according to the application principles of the VFDAS in **Fig.5-8**, the 3D information associated with the final fixture assembly model is evaluated and analysed during the conversion process from VFDAS system to Pro/ENGINEER. This final fixture assembly model is considered as the main output of the VFDAS system. The 3D information usually refers to physical properties, 3D geometric data, spatial relationship of fixture assembly, tolerance information and so on.

After using the functions of fixture element selection, fixture layout design, assembly planning and essential analysis provided by the VFDAS, the final optimal fixture assembly is determined. The final fixture assembly model is considered as a useful output of the VFDAS that is associated with the CAD format *.DXF. Comparing with other CAD formats, such as: *.STL and *.STEP, the output model of fixture assembly with the *.DXF format contains the usable information related to the spatial relationship and relative position for each fixture element. The information can be accessed and edited by using a standard CAD system such as: Pro/ENGINEER **(Fig.6-65)**. Using Pro/ENGINEER, the fixture designers are allowed to individually adjust the assembly constraint and relationship with regard to each fixture element until the optimal result is created. Afterwards, the fixture design result is efficiently applied to the manufacturing process. As a result, the *.DXF is identified as the appropriate CAD format amongst other CAD formats for the integration process between the VFDAS and CAD systems.

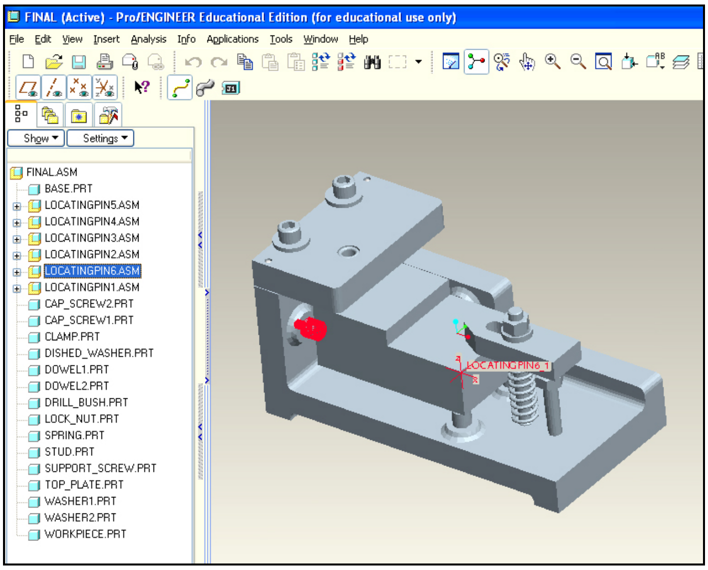


Fig.6-65. The VFDAS Output Model Opened in Pro/ENGINEER

Using *.DXF format, the CAD geometry data, physical properties and the spatial relationship and position data with regard to the output fixture assembly model correctly remain during the conversion process from the VFDAS system to the CAD system in addition to tolerance information. If necessary, the relevant tolerance information for each fixture element can be added once the output fixture assembly model is opened in Pro/ENGINEER. In addition, the physical properties associated with the fixture assembly model that are converted to the Pro/ENGINEER become some static physical parameters associated with each element model. These can not realistically demonstrate the fixture element behaviour in use.

6.5 System Development Phase Three

In the development Phase Three of the VFDAS system, more functions related to the fixture design and assembly were integrated into the existing VFDAS framework. The following sections describe how each functional objective for improving the capability of the VFDAS was developed according to the behaviour programming process based on Virtools.

6.5.1 The Functional Objective of Real-time 3D Collision Detection in Accuracy

6.5.1.1 Literature Study of Collision Detection

Although collision detection algorithms have been improved in the last decade, collision detection is still a major bottleneck in many VR applications. The importance of the role of collision detection acting as in realistic interaction modelling between 3D virtual objects is emphasized in VR applications where the motion of an object is constrained by collision with other objects. The success in developing reliable software for exact collision detection between convex objects was presented by Cohen et al in 1995 [86]. The paper explains a two-level approach which is applied to prune multiple object pairs using bounding boxes and perform exact collision detection between pairs of polyhedral models.

Gottschalk et al addressed the progress of developing a VR software for the collision detection of noncovex objects in 1996 [46]. Although many methods were reported to find a solution regarding the collision detection of nonconvex objects, it is difficult to obtain any method in literature which has capability of handling complex issues. These complex issues e.g. near-miss detection can not be solved by existing collision detection methods.

In 1995, Kuehne and Oliver suggested that the lack of computational efficiency had discouraged VR researchers to introduce collision detection into complex assembly simulation [92]. In terms of the problem of nonconvex objects, an approach is to decompose nonconvex objects i.e. concave objects into convex segments and apply available method on these decomposed convex objects. Nevertheless, this method has shown a few drawbacks and is hence unlikely to be efficient. This research outcome was summarized by Garey et al in 1979 and O'Rourke et al in 1983 [93, 94].

To narrow the research gap of exact collision detection concerning concave 3D objects, the simulation techniques used in the VFDAS system not only provide the fixture designers with real-time 3D collision detection in accuracy but also achieve good physical behaviour simulation based on VR technology.

6.5.1.2 The Behaviour Programming for Accurate Collision Detection

The behaviour programming process for the functional objective of real-time 3D collision detection in accuracy with regard to fixture elements is explained as follows. To take example for the fixture element called 'Base', the specific behaviour programming content for its own accurate collision detection is presented in **Fig.6-66**. Unlike the behaviour programming process shown previously in **Fig.6-14**, the number of convex and concave elements that are used to comprise the physics surface of the physical object requires to be specified via the Edit Settings of the 'Physicalize' BB according to the picture below **(Fig.6-67)**. The physical object here is the fixture element 'Base'. The Concave Count of this 'Physicalize' BB is specified as the number of three, which means that three concave elements are used to form the physics collision surface of the fixture element 'Base'. Thereafter, three different Parameters are created in this 'Physicalize' BB. These three Parameters that define the three separate collision parts of the fixture element 'Base', i.e. the three concave elements, are respectively specified as the Mesh named 'bottombasepart_Mesh', the Mesh named 'frontbasepart_Mesh' and the Mesh named 'baseplateleft_Mesh' **(Fig.6-68)**.

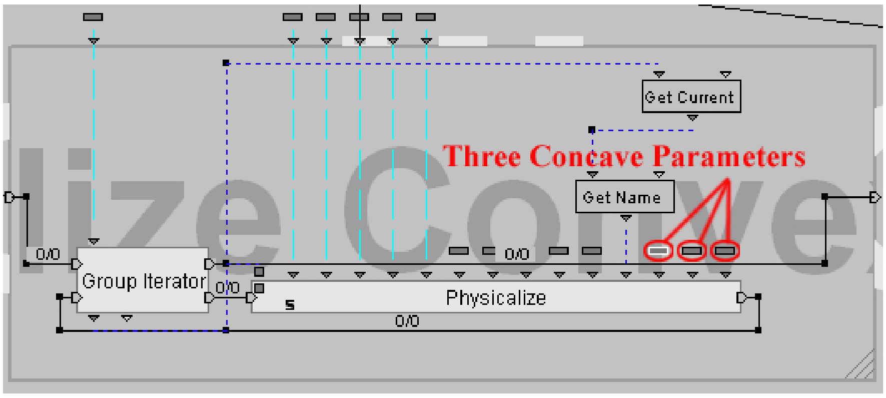
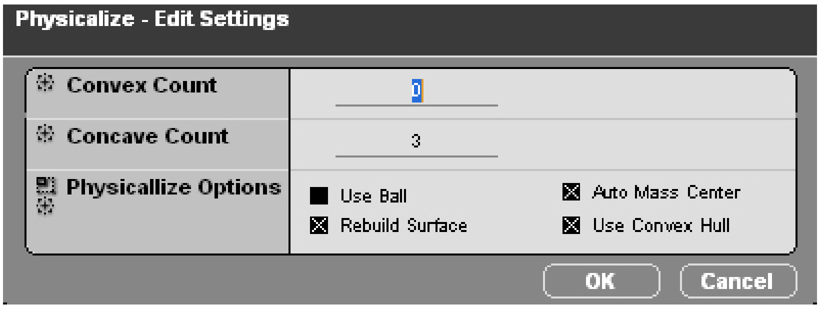


Fig.6-66. The Behaviour Programming for the Accurate Collision Detection of the ‘Base’ Element

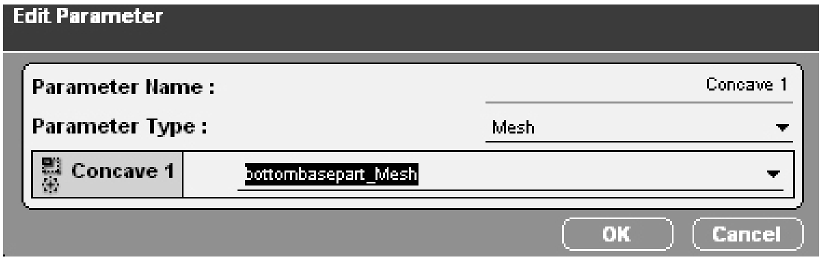

Physicalize - Edit Settings

Convex Count: 0

Concave Count: 3

Physicallize Options:
■ Use Ball ☒ Auto Mass Center
☒ Rebuild Surface ☒ Use Convex Hull

Fig.6-67. The Edit Settings of the ‘Physicalize’ BB


Edit Parameter

Parameter Name : Concave 1

Parameter Type : Mesh

Concave 1: bottombasepart_Mesh

Fig.6-68. The Parameter Specified for One of the Three Concave Elements

These three concave elements that comprise the physics collision surface of the fixture element ‘Base’ are actually three separate 3D Meshes. These Meshes are generated by appropriately pruning and dividing the original 3D model of the ‘Base’ element **(Fig.6-69)** into three separate parts. The 3D geometries related to these three collision Meshes are shown in the following pictures **(Fig.6-70) (Fig.6-71) (Fig.6-72)**.

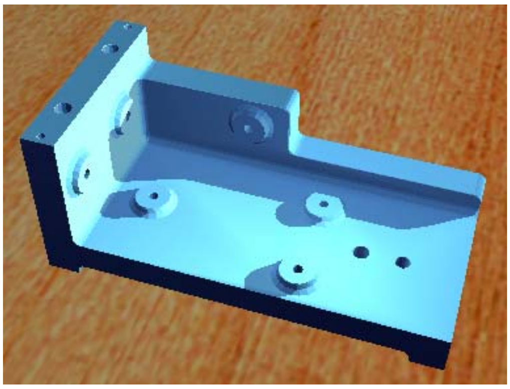


Fig.6-69. The Original 3D Model of the ‘Base’ Element

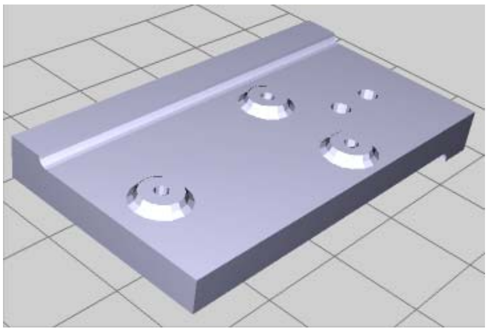


Fig.6-70. The Concave Element Named ‘bottombasepart_Mesh’

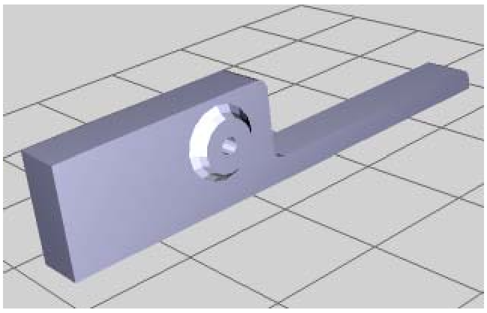


Fig.6-71. The Concave Element Named ‘frontbasepart_Mesh’

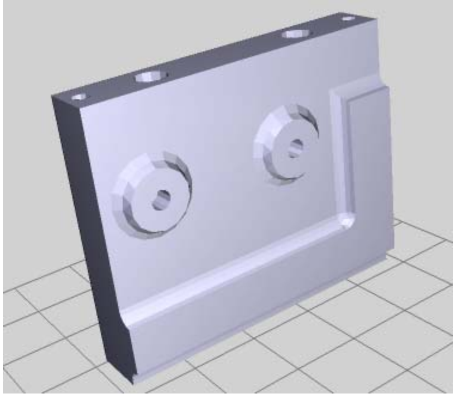


Fig.6-72. The Concave Element Named ‘baseplateleft_Mesh’

Due to the similar principle, the behaviour programming process for the accurate collision detection of the‘Base’ element presented above is efficiently applied to carry out the 3D collision detections for other fixture elements included in Pre-defined VR Library.

6.5.2 The Functional Objective of Recording the Animation of Assembly Process

The useful function provided by the Assembly Process Planning and Evaluation Module is to record the animation for interactive fixture design and assembly process in VFDAS. Using this animation, the process of selecting fixture elements from the Pre-defined VR Library and positioning them around a workpiece to form the final fixture assembly, is demonstrated. This process of a fixture design is normally carried out by the users. More detail with regard to Assembly Process Planning and Evaluation Module can be seen in previous section 5.3.12.

The behaviour programming content for the functional objective of recording the animation for the interactive fixture assembly process is presented in the following pictures **(Fig.6-73) (Fig.6-74) (Fig.6-75) (Fig.6-76)**. Therefore, four essential Scripts are applied to this behaviour programming process.

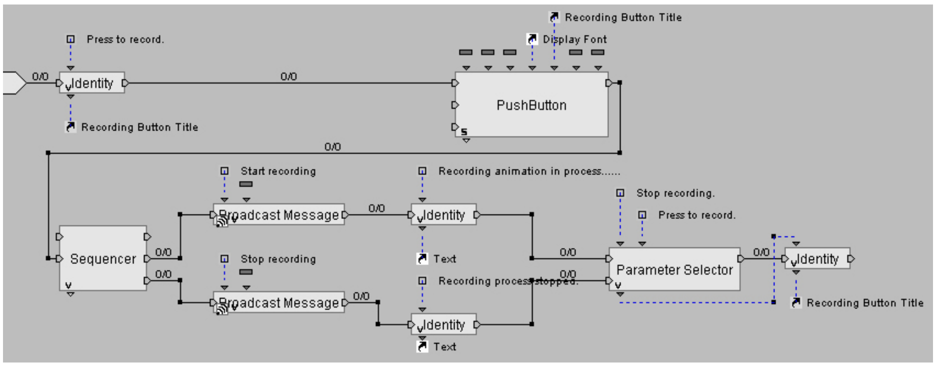


Fig.6-73. The Behaviour Programming Script Named ‘Button recording’

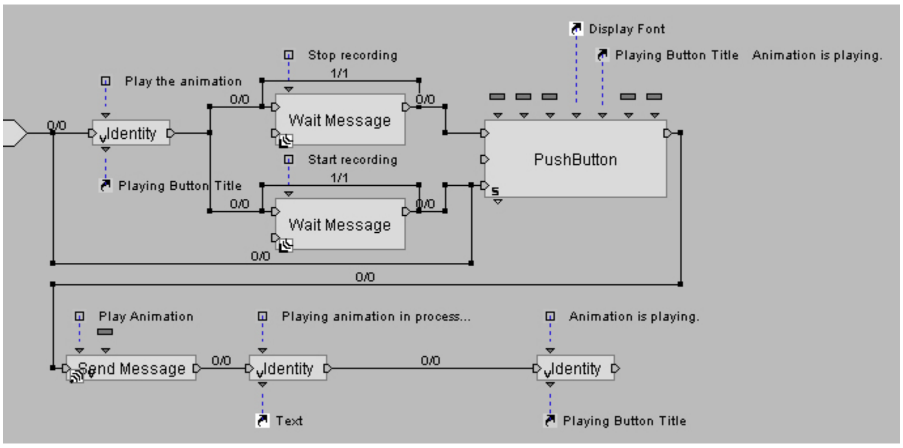


Fig.6-74. The Behaviour Programming Script Named 'Play Animation Button'

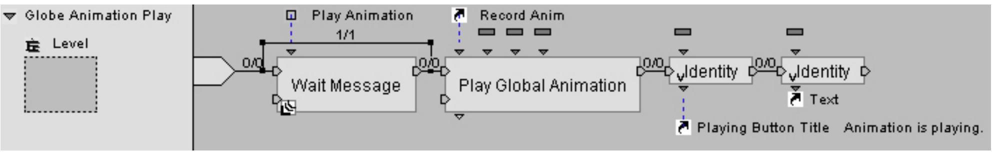


Fig.6-75. The Behaviour Programming Script Named 'Globe Animation Play'

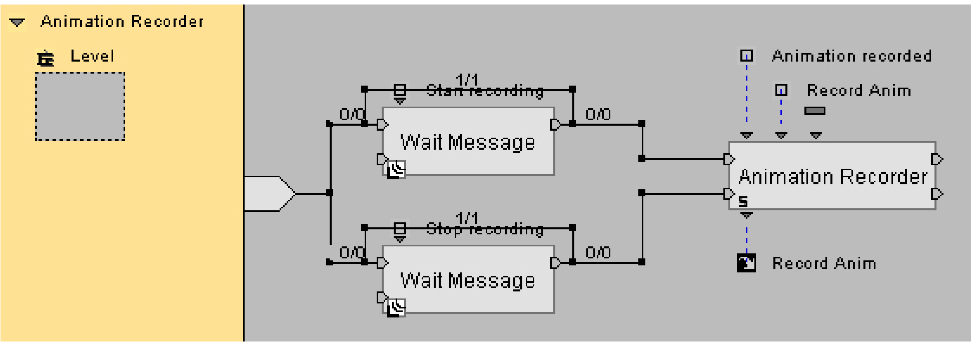


Fig.6-76. The Behaviour Programming Script Named 'Animation Recorder'

The behaviour programming Script named 'Button recording' in **Fig.6-73** is assigned to the push button called 'Animation Recorder'. This Script is used to achieve the functional feature of developing the Control Button by which the users can start or stop recording the animation of the fixture assembly process.

The behaviour programming Script named 'Play Animation Button' in **Fig.6-74** is assigned to the push button called 'Animation Player'. This Script is used to achieve the functional feature of developing the Control Button by which the recorded fixture assembly animation can start or stop playing.

The behaviour programming Script named ‘Globe Animation Play’ in **Fig.6-75** that is assigned to the ‘Level’ is used to achieve the functional feature of playing the recorded animation for the fixture assembly process while the Control Button called ‘Animation Player’ is pressed by the users.

The behaviour programming Script named ‘Animation Recorder’ in **Fig.6-76** that is assigned to the ‘Level’ is used to achieve the functional feature of recording the global animation for the fixture assembly process once the Control Button called ‘Animation Recorder’ is pressed by the users.

According to these four essential Scripts, a set of BBs are applied to this behaviour programming process. The BBs in the Script named ‘Button recording’ are the ‘Identity’ BB, the ‘Push Button’ BB, the ‘Sequencer’ BB, the ‘Broadcast Message’ BB and the ‘Parameter Selector’ BB. The BBs in the Script named ‘Play Animation Button’ are the ‘Identity’ BB, the ‘Wait Message’ BB, the ‘Push Button’ BB and the ‘Send Message’ BB. The BBs in the Script named ‘Globe Animation Play’ are the ‘Wait Message’ BB, the ‘Play Global Animation’ BB and the ‘Identity’ BB. In addition, the BBs in the Script named ‘Animation Recorder’ are the ‘Wait Message’ BB and the ‘Animation Recorder’ BB.

The ‘Identity’ BB is used to output the Input Parameters while it is activated. The ‘Push Button’ BB is used to convert a 2D Frame into a push button. The ‘Sequencer’ BB is used to generate the nth output while ‘In’ is consecutively activated for the nth time. The ‘Broadcast Message’ BB is used to propagate a Message to each object that is waiting for it in a VE. The ‘Parameter Selector’ BB is used to route the corresponding Input Parameter to the Output Parameter according to the Input activated. The ‘Wait Message’ BB is used to wait the receipt of a message, through which a useful activation output is generated once the message is received. Another ‘Send Message’ BB is used to send a message to the target 3D Entity. The ‘Play Global Animation’ BB is used to enable a series of 3D Entities performing a global animation. Finally, the ‘Animation Recorder’ BB is used to record the animation of an object or a Group of objects by storing their Position, Orientation and Scale keyframes during every N frames. The introductory graphs with regard to these BBs are presented according to the pictures to clarify their specifications **(Fig.6-5) (Fig.6-6) (Fig.6-8) (Fig.6-12) (Fig.6-29) (Fig.6-61) (Fig.6-77) (Fig.6-78) (Fig.6-79)**.

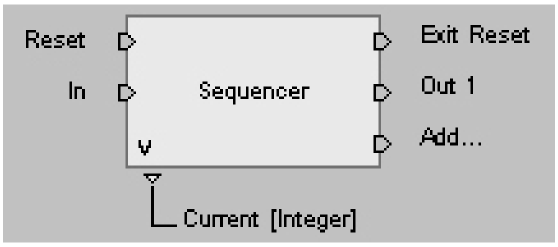


Fig.6-77. The Introductory Graph of the 'Sequencer' BB

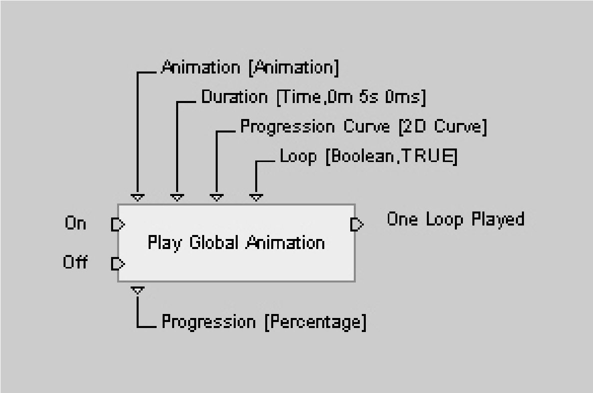


Fig.6-78. The Introductory Graph of the 'Play Global Animation' BB

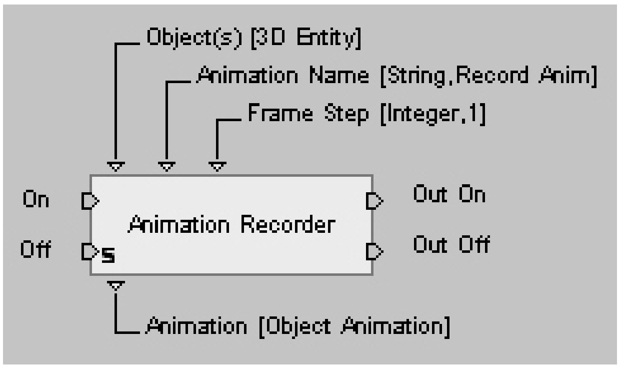


Fig.6-79. The Introductory Graph of the 'Animation Recorder' BB

The logic and correlation of this behaviour programming for the functional objective of recording the fixture assembly animation are as follows. When the Control Button called 'Animation Recorder' on the GUI of the VFDAS is pressed by the users, the behaviour programming Script 'Button recording' **(Fig.6-73)** is executed. Thus, the 'Broadcast Message' BB is used to propagate a Message called 'Start recording' to every object that is waiting for it in the VE. In the meantime, the ‘Wait Message’ BB in the Script named ‘Animation Recorder’ **(Fig.6-76)** waits for receipt of the Message ‘Start recording’ and then passes an activation output to the next ‘Animation Recorder’ BB. Thereafter, the ‘Animation Recorder’ BB starts recording the animation of fixture assembly process with regard to 3D objects included in a Group named ‘Animation recorded’. This Group comprises a series of selected fixture elements involved in the fixture design process. As soon as the Control Button ‘Animation Recorder’ is pressed once more, the Script named ‘Button recording’ **(Fig.6-73)** is activated again so that another ‘Broadcast Message’ BB propagates the Message called ‘Stop recording’ to the objects that are awaiting for it in the VE. Simultaneously, another ‘Wait Message’ BB in the Script named ‘Animation Recorder’ **(Fig.6-76)** passes another activation output to the next ‘Animation Recorder’ BB once the Message ‘Stop recording’ is received. Due to the Behaviour Link being connected with the bIn ‘Off’ of the ‘Animation Recorder’ BB, the ‘Animation Recorder’ BB then stops recording the animation of the fixture assembly process. While the Control Button ‘Animation Player’ on the GUI is pressed by the users, the Script named ‘Play Animation Button’ **(Fig.6-74)** is activated. Thus, the ‘Send Message’ BB is used to send a Message called ‘Play Animation’ to the target 3D Entity named ‘Level’. Meanwhile, the ‘Wait Message’ BB in the Script named ‘Globe Animation Play’ **(Fig.6-75)** waits for receipt of the ‘Play Animation’ Message and can then propagate the activation output to the next ‘Play Global Animation’ BB. The ‘Play Global Animation’ BB then starts playing the recorded animation of fixture assembly process.

6.5.3 The Functional Objective of the Viewpoint Change Unit

The Viewpoint Change Unit associated with the Design Viewpoint Control Module is used to switch between the different orthographic viewing modes: Front Viewpoint, Right Viewpoint, Top Viewpoint and Perspective Viewpoint. Each orthographic viewing mode in this Viewpoint Change Unit provides users with the function of zooming in or out to ensure the suitable orthographic viewpoint is identified. More details with regard to Viewpoint Change Unit can be seen in previous section 5.3.8.

The behaviour programming content for the functional objective of the Viewpoint Change Unit is presented in the following pictures **(Fig.6-80) (Fig.6-81) (Fig.6-82) (Fig.6-83)**. Four essential Scripts are applied to the behaviour programming process.

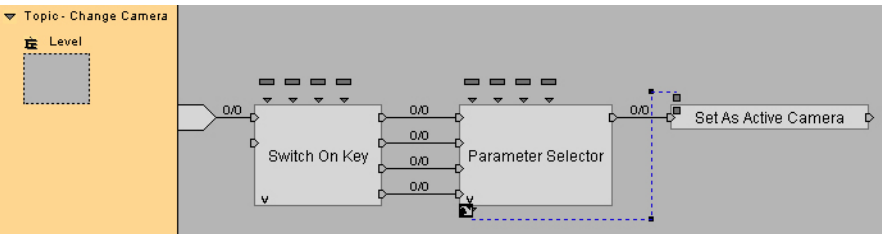


Fig.6-80. The Behaviour Programming Script Named 'Change Camera'

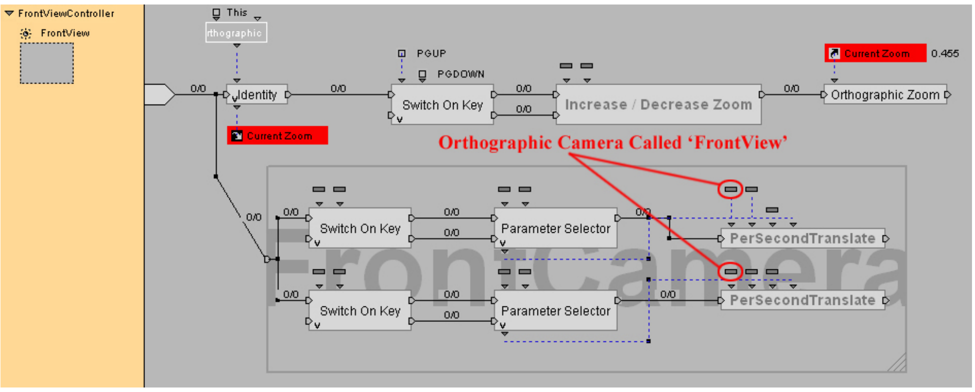


Fig.6-81. The Behaviour Programming Script Named 'FrontViewController'

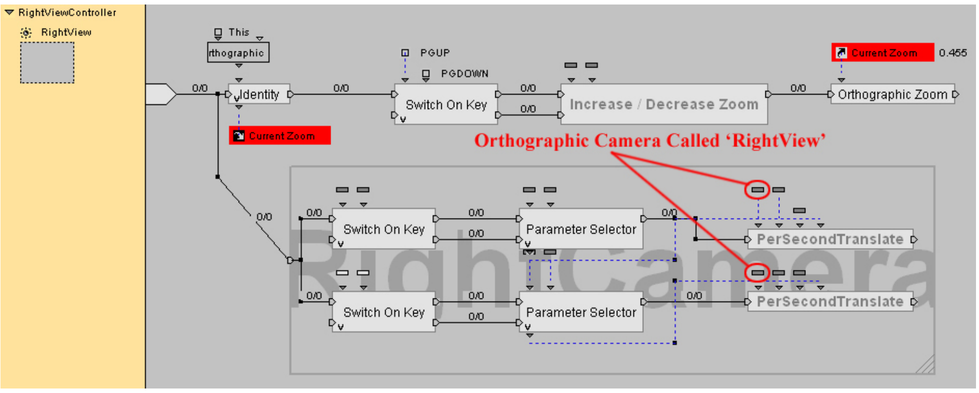


Fig.6-82. The Behaviour Programming Script Named 'RightViewController'

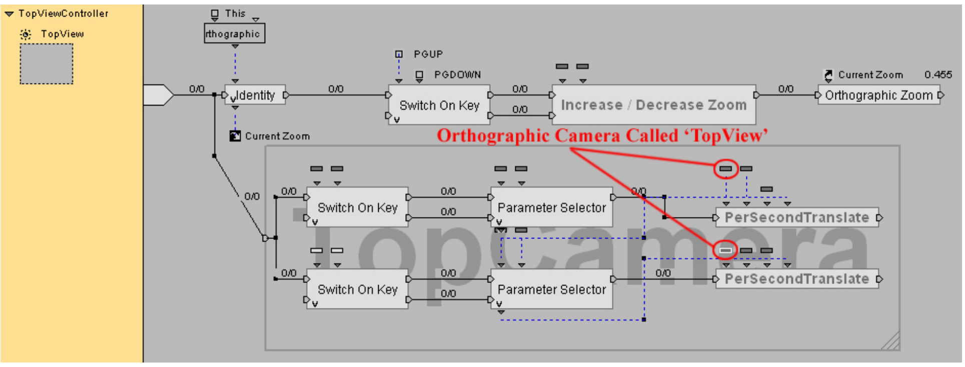


Fig.6-83. The Behaviour Programming Script Named 'TopViewController'

The behaviour programming Script named 'Change Camera' in **Fig.6-80** that is assigned to the 'Level' is used to achieve the functional feature of specifying the computer keys of F9, F10, F11 and F12 with the four corresponding orthographic cameras. These orthographic cameras are respectively named 'FrontView', 'RightView', 'TopView' and 'Main Camera' so as to switch the active Camera in the VE while the corresponding key is pressed.

The behaviour programming Script named 'FrontViewController' in **Fig.6-81** is assigned to the orthographic Camera named 'FrontView'. This Script is used to achieve the functional features of zooming in or out the orthographic 'FrontView' Camera and parallel shifting it alongside the perpendicular plane.

The behaviour programming Script named 'RightViewController' in **Fig.6-82** is assigned to the orthographic Camera named 'RightView'. This Script is used to achieve the functional features of zooming in or out the orthographic 'RightView' Camera and parallel shifting it alongside its perpendicular plane.

The behaviour programming Script named 'TopViewController' in **Fig.6-83** is assigned to the orthographic Camera named 'TopView'. This Script is used to achieve the functional features of zooming in or out the orthographic 'TopView' Camera and parallel shifting it alongside its perpendicular plane.

According to these four Scripts, a set of BBs are applied to this behaviour programming process. The BBs in the Script named 'Change Camera' are the 'Switch On Key' BB, the 'Parameter Selector' BB and the 'Set As Active Camera' BB. The BBs and

Parameter Operation in the Script named 'FrontViewController' are the 'Identity' BB, the 'Switch On Key' BB, the 'Parameter Selector' BB, the 'Op' BB, the 'Threshold' BB, the 'Orthographic Zoom' BB, the 'Per Second' BB, the 'Translate' BB and the Parameter Operation called 'Get Orthographic Zoom'. The BBs and Parameter Operation in the Script named 'RightViewController' are the 'Identity' BB, the 'Switch On Key' BB, the 'Parameter Selector' BB, the 'Op' BB, the 'Threshold' BB, the 'Orthographic Zoom' BB, the 'Per Second' BB, the 'Translate' BB and the Parameter Operation called 'Get Orthographic Zoom'. Finally, the BBs and Parameter Operation in the Script named 'TopViewController' are the 'Identity' BB, the 'Switch On Key' BB, the 'Parameter Selector' BB, the 'Op' BB, the 'Threshold' BB, the 'Orthographic Zoom' BB, the 'Per Second' BB, the 'Translate' BB and the Parameter Operation called 'Get Orthographic Zoom'.

In addition, two types of Behaviour Graphs that are named 'Increase/Decrease Zoom' and 'PerSecondTranslate' are applied to this behaviour programming process. The Behaviour Graph 'Increase/Decrease Zoom' comprises the 'Parameter Selector' BB, the 'Op' BB and the 'Threshold' BB. The Behaviour Graph 'PerSecondTranslate' comprises the 'Op' BB, the 'Per Second' BB and the 'Translate' BB. The expanded 'Increase/Decrease Zoom' and 'PerSecondTranslate' Behaviour Graphs are shown in the following pictures **(Fig.6-84) (Fig.6-85)**.

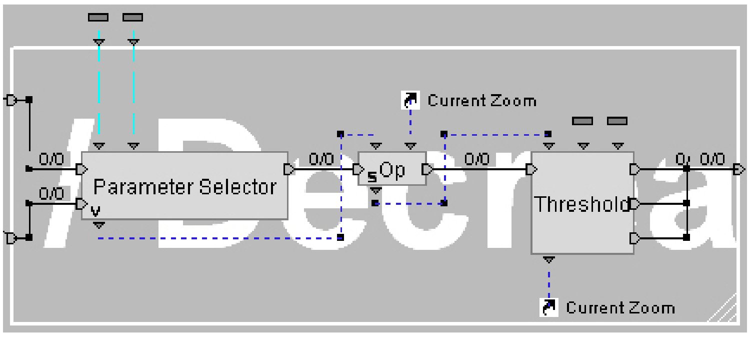


Fig.6-84. The Expanded Behaviour Graph Named 'Increase/Decrease Zoom'

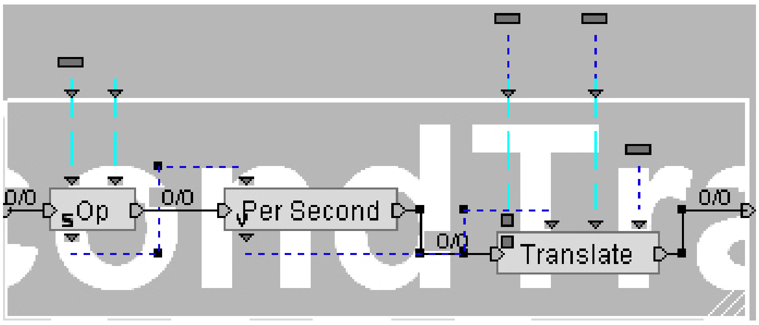


Fig.6-85. The Expanded Behaviour Graph Named 'PerSecondTranslate'

The 'Switch On Key' BB is used to trigger the relevant output while a specified key is received. The 'Parameter Selector' BB is used to route the corresponding Input Parameter to the Output Parameter according to the Input activated. The 'Set As Active Camera' BB is used to change the active camera in the VE. The 'Identity' BB is used to output the Input Parameters while it is activated. The 'Op' BB is used to process any valid Parameter Operation. The 'Threshold' BB is used to bind a variable between two limits (Minimum and Maximum) as well as create the corresponding output according to the comparison result. The 'Orthographic Zoom' BB is used to emulate a zoom while the viewing mode is orthographic. The 'Per Second' BB is used to calculate a progression (Y) according to a given velocity (X): Y=X*Elapsed Time during one Frame. The 'Translate' BB is used to translate a 3D Entity. The introductory graphs with regard to these BBs are presented in the pictures to clarify their specifications **(Fig.6-12) (Fig.6-28) (Fig.6-29) (Fig.6-30) (Fig.6-31) (Fig.6-32) (Fig.6-86) (Fig.6-87) (Fig.6-88)**.

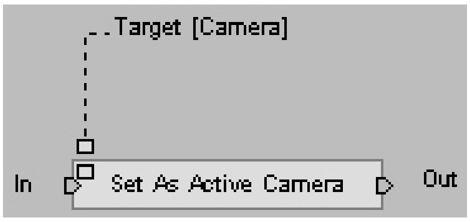


Fig.6-86. The Introductory Graph of the 'Set As Active Camera' BB

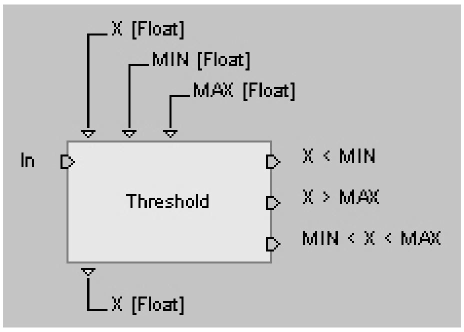


Fig.6-87. The Introductory Graph of the 'Threshold' BB

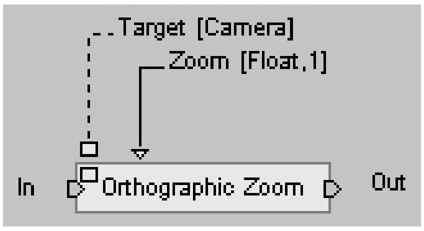
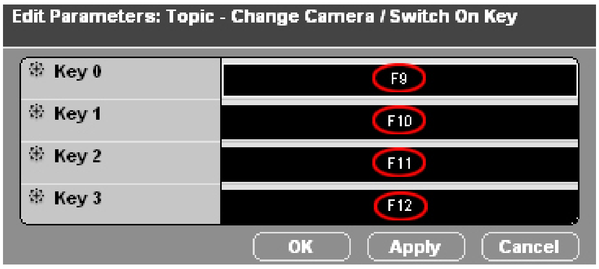


Fig.6-88. The Introductory Graph of the 'Orthographic Zoom' BB

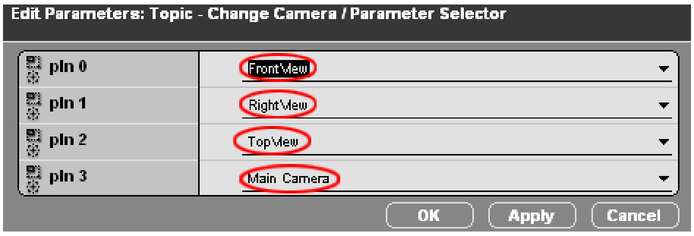
The logic and correlation of this behaviour programming for the functional objective of the Viewpoint Change Unit are as follows. According to the behaviour programming Script named 'Change Camera', the 'Switch On Key' BB **(Fig.6-89)** is used to specify four sequential keys of F9, F10, F11 and F12. These keys are associated with the four corresponding orthographic cameras by using the 'Parameter Selector' BB **(Fig.6-90)**. The four orthographic cameras are respectively Camera 'FrontView', Camera 'RightView', Camera 'TopView' and Camera 'Main Camera' in the VE. Once the specified key is pressed by users, the 'Set As Active Camera' BB will switch the corresponding camera to the active camera so that the corresponding viewing mode is activated in the VE i.e. the Front Viewpoint, the Right Viewpoint, the Top Viewpoint or the Perspective Viewpoint. Furthermore, the behaviour programming Scripts named 'FrontViewController', 'RightViewController' and 'TopViewController' are used to achieve the functional features of zooming in or out the corresponding orthographic Camera and parallel shifting it alongside the perpendicular plane. These cameras are respectively Camera 'FrontView', Camera 'RightView' and Camera 'TopView' so as to ensure the suitable orthographic viewpoint is identified.



Edit Parameters: Topic - Change Camera / Switch On Key

Key 0	F9
Key 1	F10
Key 2	F11
Key 3	F12

Fig.6-89. The Parameters Specified for the 'Switch On Key' BB



Edit Parameters: Topic - Change Camera / Parameter Selector

pIn 0	FrontView
pIn 1	RightView
pIn 2	TopView
pIn 3	Main Camera

Fig.6-90. The Parameters Specified for the 'Parameter Selector' BB

6.6 Summary

Development of VFDAS took place over three incremental phases. Phase 1 established the system framework including the construction of the fixture element library, the function of interactive selection and manipulation and the function of assigning the Physics Modes. Phase 2 focused on transfer of data and information between Pro/ENGINEER and VFDAS. Phase 3 developed more advanced functions to the system including virtual object grouping, collision detection, assembly animation recording, 3D viewpoint selection and interface controls etc. As a result, a fully functioning prototype of VFDAS has been developed that can allow a user to: import an intended 3D workpiece from Pro/ENGINEER, select fixture elements from the VR library, place them onto the workpiece surface, assemble the fixture configuration, record the assembly process, view a pre-recorded animation of the fixture design and assembly process and view the virtual environments from different viewpoints. The VFDAS system was validated in terms of the collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results are presented in the following chapter.

Chapter 7. Technical Validation of the VFDAS System

7.1 Introduction

This chapter will technically discuss and validate the simulation results of the VFDAS system with respect to collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results of VFDAS are demonstrated and quantified by a series of simple examples so as to show what the system can achieve and what the limitations are. In addition, the essential computer platform used for the VFDAS will also be discussed.

7.2 Collision Detection

Within VFDAS system, the virtual 3D objects with different geometry and topology can be classified into two categories: convex objects and concave objects. A hull is a volume as an approximation to the volume inhabited by a 3D object [95]. A hull can be considered as a special type of bounding box that is not restrained to a rectangular volume and whose accuracy can be controlled by the choice of mesh. A hull may be convex or concave.

Collision detection is meant to be full contact of two objects in the virtual environment. However, an unexpected distance between two contacted surfaces still exists in the VFDAS system, which can not be detected by the VFDAS itself. The Pro/ENGINEER software was used as a tool to measure this error of real-time collision detection in VFDAS. This error is completely derived from the VFDAS itself and does not relate to Pro/ENGINEER.

On the other hand, the real-time 3D collision detection in VFDAS can be usually classified into Surface Contact, Line Contact and Point Contact. In order to quantify the final results of collision detection within VFDAS system, the accuracy of collision detection will be reported and discussed according to these three essential contacts in the following section.

7.2.1 The Convex Objects

A convex hull **(Fig.7-1)** represents a virtual object without holes or hollows. Consider stretching a rubber band around an object. The object is convex if the rubber band touches the surface of the object at all points [95].

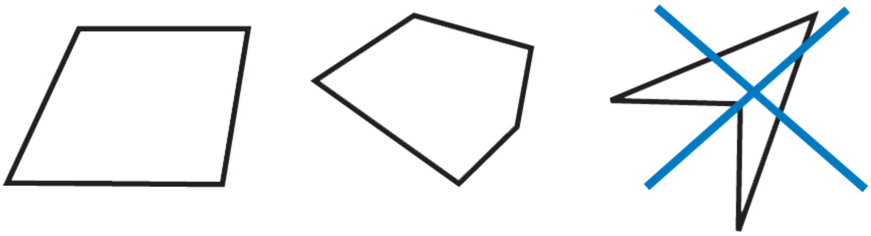


Fig.7-1. The Comparison of Convex Hulls and a Non-Convex Hull

Within the VFDAS system, a typical example of the collision detection between convex objects can refer to two standard cubes with low polygon models **(Fig.7-2)**. These two cubes are statically plied up each other on the ground in the virtual environment under the gravity effect. The dimensions of one cube are 30 mm for its each equal edge and the dimensions of another cube are 50 mm for its each equal edge **(Fig.7-2)**. This example of collision detection represents the type of Surface Contact.

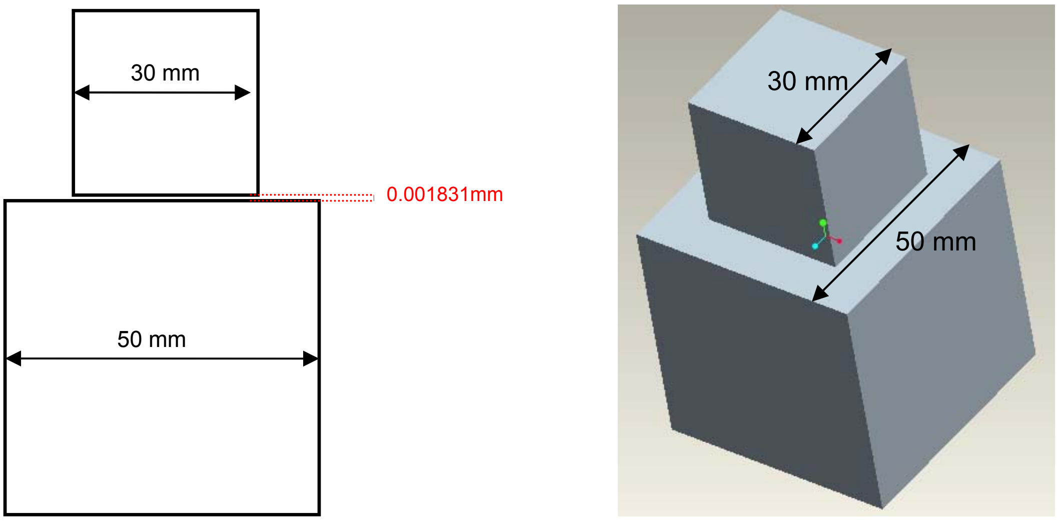


Fig.7-2. The Surface Contact Type of Collision Detection Between Convex Objects

While these two cubes may be in contact and keep static in force equilibrium under the gravity effect in virtual environment according to the collision detection function provided by the VFDAS, these two cube models can be exported into the Pro/ENGINEER software using Design Configuration Output Module of VFDAS. Using the analysis function of 'Measure' in Pro/ENGINEER, an unexpected distance between the two contacted surfaces of these cubes models could be calculated as the value of 0.001831mm. Furthermore, the dimension and scale of these models would always remain the same throughout the conversion process from the VFDAS system to the Pro/ENGINEER vice versa.

In terms of the surface contact type of collision detection, this value indicates the best collision detection result between convex objects within VFDAS because these two cubes are regarded as standard convex hulls without any concave topology representation upon the models. As a result, the accuracy of surface contact type of collision detection between convex objects within VFDAS is 0.001831mm.

According to line contact type of collision detection, another typical example of collision detection between convex objects within the VFDAS system refers to one standard cube and one standard cylinder with low polygon models **(Fig.7-3)**. This cube is statically placed on the ground in the virtual environment under the gravity effect and this cylinder is placed on the top of the cube with curve surface contacting this cube. The dimensions of the cube are 50 mm for its each equal edge. The diameter of the cylinder is 15 mm and the length of this cylinder is 30 mm **(Fig.7-3)**. This example of collision detection can be considered as the type of Line Contact because the actual contact area between the cube and cylinder in this case is only a line shape.

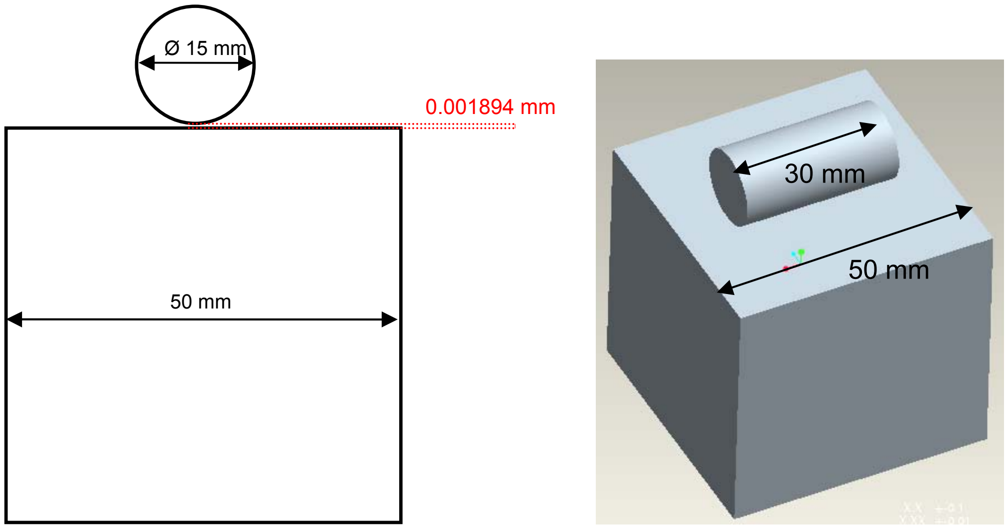


Fig.7-3. The Line Contact Type of Collision Detection Between Convex Objects

Whilst the cube and cylinder can be in contact and keep static in force equilibrium under the gravity effect in the physics virtual environment, these two geometric models can be exported into the Pro/ENGINEER software by the VFDAS. Applying the analysis function of 'Measure' in Pro/ENGINEER, a distance between the cube's contacted surface and the cylinder tangent line that is closest to the surface of the cube could be detected and measured as the value of 0.001894 mm.

This value determines the best collision detection result for a line contact type of collision detection between convex objects within the VFDAS system. As a result, the accuracy of line contact type of collision detection between convex objects within VFDAS is 0.001894 mm which is slightly higher than the value of surface contact type of collision detection results.

According to point contact type of collision detection, one more example of collision detection between convex objects within the VFDAS refers to one standard cube and one standard sphere **(Fig.7-4)**. This cube is statically placed on the ground in the virtual environment under the gravity effect and this sphere is placed on the top of the cube. The dimensions of the cube are also 50 mm for its each equal edge. The diameter of the sphere is 20 mm **(Fig.7-4)**. This example of collision detection can be considered as the type of Point Contact because the actual contact area between the cube and sphere is only a point.

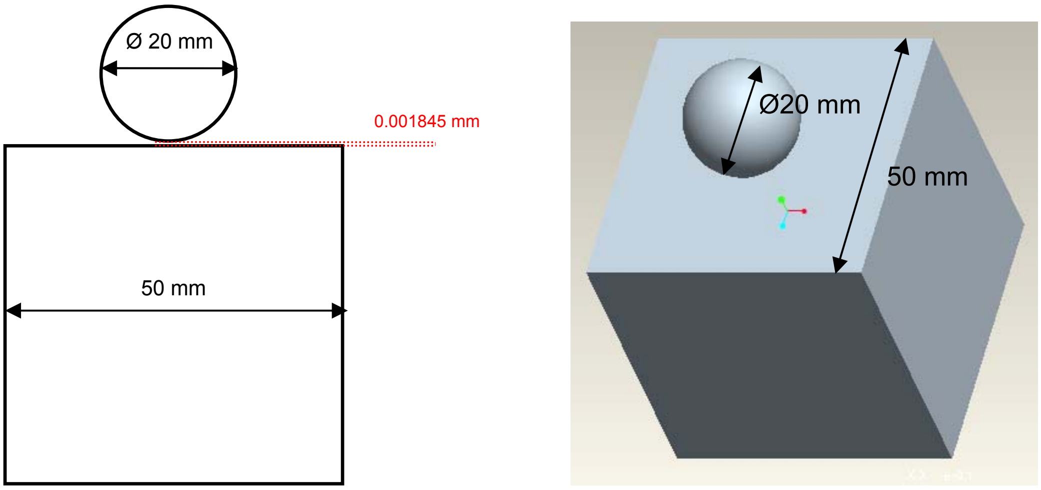


Fig.7-4. The Point Contact Type of Collision Detection Between Convex Objects

While the cube and sphere can be in contact and keep static in force equilibrium under the gravity effect in the physics virtual environment, these two geometric models could be exported into the Pro/ENGINEER software by the VFDAS. Using the analysis function of 'Measure' in Pro/ENGINEER, an unexpected distance between the cube's contact surface and the sphere's tangent point that is closest to the surface of the cube could be calculated as the value of 0.001845 mm.

This value indicates the best collision detection result for point contact type between convex objects within VFDAS system. As a result, the accuracy of point contact type of collision detection within VFDAS is 0.001845 mm which is slightly higher than the value for surface contact type and lower than the value for line contact type.

According to three essential contact types, the comparison of the accuracy of collision detection within the VFDAS system is shown in the table below **(Fig.7-5)**:

VFDAS System	Surface Contact Type	Line Contact Type	Point Contact Type
Accuracy of Collision Detection	0.001831mm	0.001894 mm	0.001845 mm

Fig.7-5. The comparison of collision detection accuracy in three contact types

7.2.2 The Concave Objects

A concave hull **(Fig.7-6)** represents a virtual object with holes or hollows. Consider stretching a rubber band around an object. The object is concave if the rubber band does not touch the surface of the object at all points.

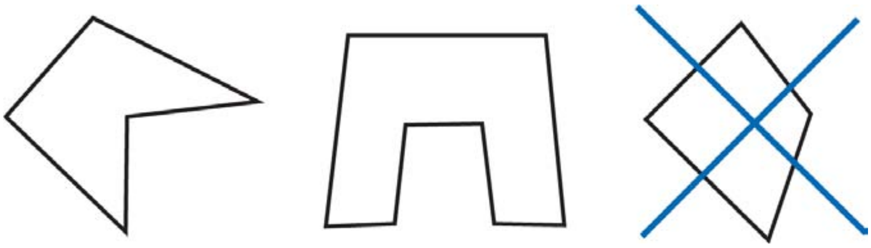


Fig.7-6. The Comparison of Concave Hulls and a Non-Concave Hull

Simulating concave objects can be very processor intensive, particularly if they move in a virtual environment. The collision detection of convex objects is usually faster to simulate than the collision detection of concave objects in VFDAS system. Consequently, a VE developer should represent an object using a convex hull representation as often as possible in order to maintain the real-time collision detection throughout the VFDAS system application. On the other hand, the convex surface representation is normally able to provide higher accuracy for 3D collision detection than the concave surface representation within VFDAS. Therefore, a good solution to simulating a complex concave surface for accurate collision detection is to approximate the concave surface by a group of convex surfaces. This is often referred to as a compound convex surface.

A typical example of the collision detection between one concave object and one convex object within the VFDAS system may refer to one arch shape and one standard cube **(Fig.7-7)**. This arch shape is statically placed on the ground in the physics virtual environment under the gravity effect and the concave hole is facing upwards opposite the ground direction. Using the interactive manipulation function provided by VFDAS, this cube was inserted from the top of the arch and finally placed within the concave part of this arch shape to be static without movement. The dimensions of the cube are 24 mm for its each equal edge. The height of this arch shape is 50 mm and the width of this arch is 75 mm and the depth of this arch shape is 35 mm. The width of the concave hole on the arch is 25 mm and the height of the concave hole upon the arch is 20 mm and the depth of the concave hole is 35 mm **(Fig.7-7)**.

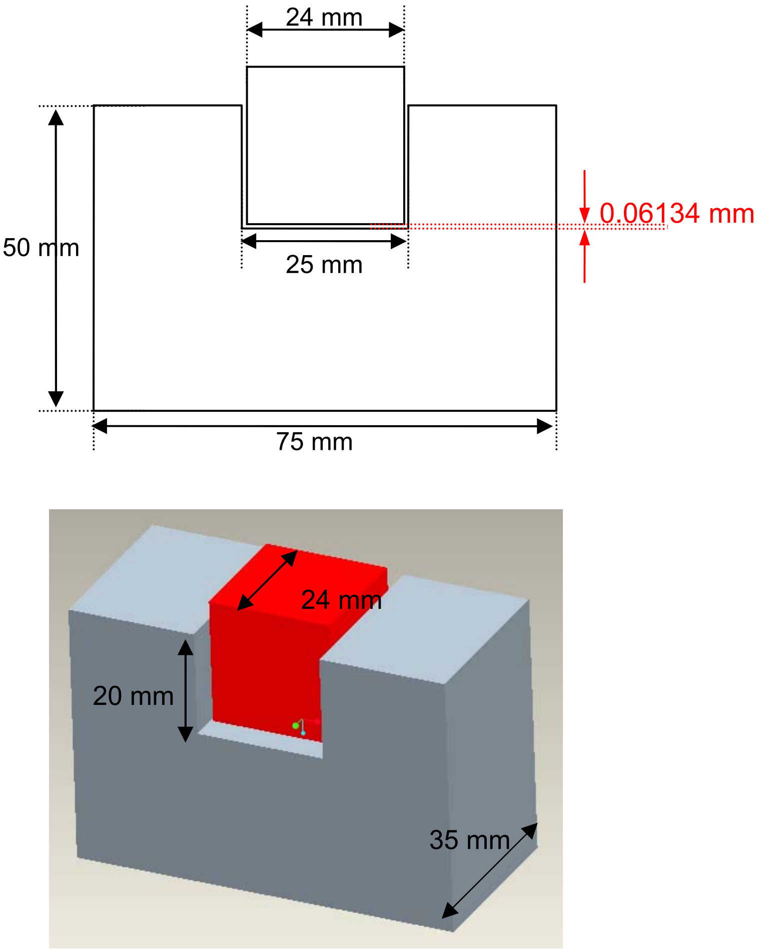


Fig.7-7. Collision Detection Between a Concave Arch and a Convex Cube

Whilst the cube and the arch shape can be in contact and keep static in force equilibrium under the gravity effect in the physics environment, these two models could be exported into the Pro/ENGINEER software by the VFDAS system. Using the analysis function of 'Measure' in Pro/ENGINEER, a greater distance between two contacted surfaces could be detected and measured as 0.06134 mm. Although this example of collision detection can also be considered as the type of Surface Contact, the inaccuracy of the collision detection incurred is much higher than the value of the collision detection result between convex objects, which was reported as 0.001831mm. The main reason for this fact can be identified as the concave surface representation within VFDAS system usually has a lower accuracy of collision detection result than that of the convex surface representation.

In order to increase the accuracy of collision detection between concave objects within VFDAS, the hull may be partitioned by explicitly identifying the convex sub-parts that compose the concave object. Furthermore, the visual representation and physics collision surface of a 3D object within VFDAS are separately defined by two different groups of meshes. The visual representation of an object is controlled by the currently active mesh. The physics collision surface of a 3D object is normally constructed from a group of object's designated meshes which determines the accuracy of collision detection.

In this example, the concave arch can be subdivided into 3 convex sub-parts i.e.3 convex meshes **(Fig.7-8)**. After defining the physics collision surface of this concave arch with these 3 convex meshes, the experiment mentioned above can be repeated one more time within VFDAS. Thereafter, these resulting models are exported into Pro/ENGINEER again by the VFDAS system. Using the analysis function of 'Measure' in Pro/ENGINEER, the distance between the two contacted surfaces of the arch shape and the cube could be measured as 0.001974 mm. Therefore, the accuracy of surface contact type of collision detection between one concave and one convex object was improved approximately 31.1 times in comparison with the previous collision detection result of 0.06134 mm.

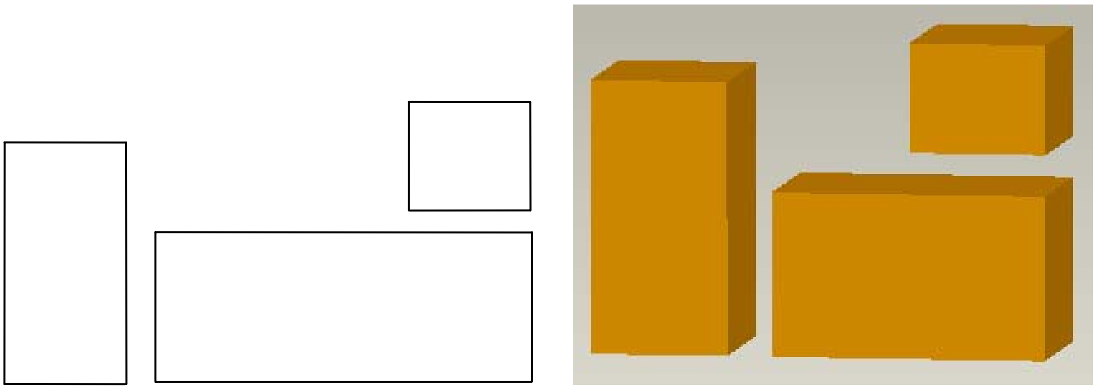


Fig.7-8. Defining Physics Collision Surface of Concave arch with 3 Convex Sub-parts

7.3 Tradeoff Between Accuracy and Rendering Speed

Although the accuracy of collision detection within the VFDAS system may be greatly improved by subdividing the complex concave surface representation into a group of convex sub-parts, it is not an appropriate method of subdividing a complex concave object into hundreds of convex objects to respect this rule. The processing time spent to manage and parse each of the convex surface representation would be finally higher, which normally results in the low rendering rate or unsatisfactory system performance to hinder the requirement of real-time 3D collision detection within VFDAS. Consequently, there is a tradeoff between the accuracy of collision detection and rendering rate which needs to be compromised so that the VFDAS system is able to provide a high accuracy of collision detection as well as operate with a relatively good simulation performance.

A typical example of explaining the tradeoff between accuracy and rendering speed within VFDAS refers to the different collision detection results between a semi-circle arch and one standard cylinder. This cylinder is an immobilized physics object within the VFDAS system and it is vertically placed on the ground in the physics virtual environment. This semi-circle arch is considered as a concave surface representation and the inner surface of concave part **(Fig.7-9)** may keep a contact with the cylinder after the semi-circle arch is moved toward the cylinder using the interactive manipulation function of VFDAS. The diameter of the cylinder is 15 mm and the height of this cylinder is 120 mm. The diameter of the semi-circle arch is 300 mm. The section view of this arch is the 50mm × 50 mm square and revolved by 180 degree according to the centerline **(Fig.7-10)**. This example of collision detection can be considered as the type of Line Contact.

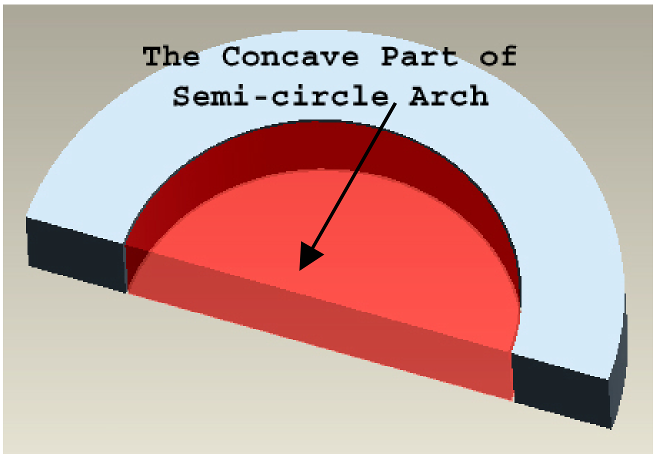


Fig.7-9. Inner surface of Concave Part of the Semi-circle Arch

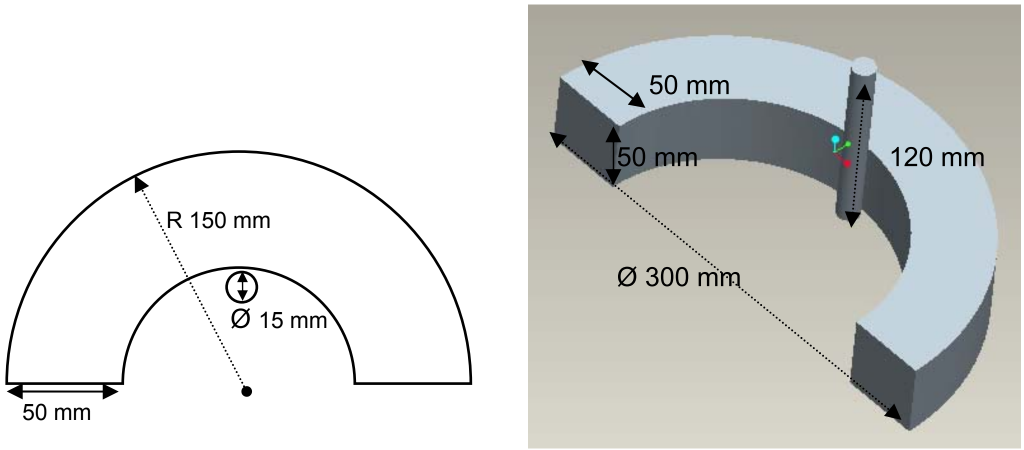


Fig.7-10. Collision Detection Between a Semi-circle Arch and a Cylinder

In the first experiment, this concave semi-circle arch was subdivided into 5 convex sub-parts so that the physics collision surface of this semi-circle arch could be defined by clearly specifying these 5 convex meshes that compose this semi-circle arch shape **(Fig.7-11)**. In fact, these 5 convex meshes still contain a small part of concave surface representation upon them. The more sub-parts the concave semi-circle arch can be divided to, the more each sub-part mesh can be approximate to the geometry of a convex surface representation. In theory, the collision detection results within VFDAS would be more accurate. On the contrary, rendering speed with regard to the system performance would be lower.

While the cylinder can be in contact with inner surface of concave part of semi-circle arch and keep static in force equilibrium under the gravity effect in the physics environment, these two models could be exported into Pro/ENGINEER software by the VFDAS system. Using the analysis function of 'Measure' in Pro/ENGINEER, a distance between the two contacted surfaces of the semi-circle arch and the cylinder could be calculated as 0.005302 mm. As a result, the accuracy of collision detection between the semi-circle arch and the cylinder was 0.005302 mm **(Fig.7-11)**. In addition, the corresponding rendering rate within VFDAS system was averagely 73.1 frames per second which speed may be detected by the Profiler of Virtools.

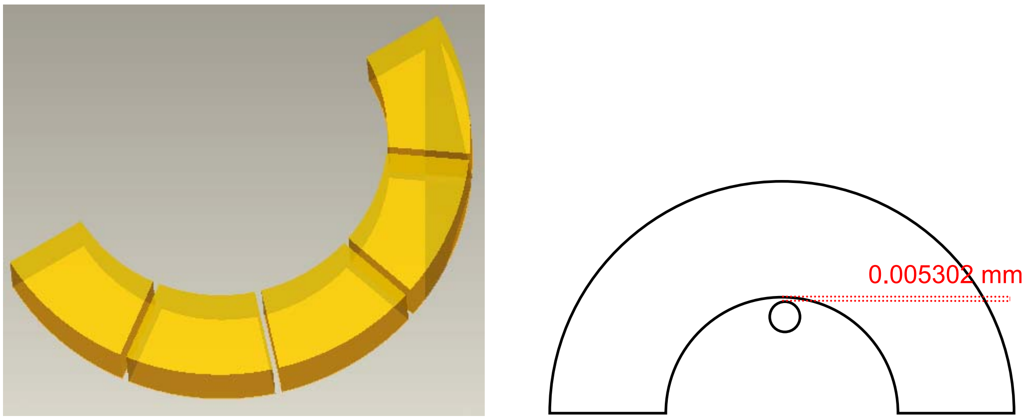


Fig.7-11. The Accuracy of Collision Detection in 5 Convex Sub-parts

In the second experiment, this concave semi-circle arch was subdivided into 30 convex sub-parts so that the physics collision surface of this semi-circle arch could be defined by specifying these 30 convex meshes **(Fig.7-12)**. While the inner surface of concave part of this semi-circle arch can be in contact with the cylinder and keep static in force equilibrium, these two models could be exported into Pro/ENGINEER software by the VFDAS system for measurement. A distance between the two contacted surfaces of the semi-circle arch and the cylinder could be detected and measured as 0.003796 mm. As a result, the accuracy of collision detection between this semi-circle arch and the cylinder was 0.003796 mm **(Fig.7-12)**. In addition, the corresponding rendering rate within the VFDAS system was averagely 28.9 frames per second which rendering speed is much slower than the previous speed of 73.1 frames per second.

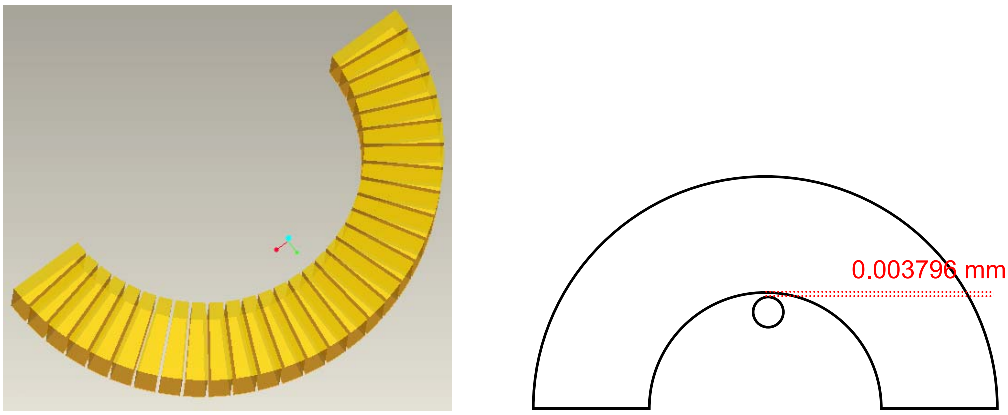


Fig.7-12. The Accuracy of Collision Detection in 30 Convex Sub-parts

Comparing the results of these two experiments, the accuracy of collision detection within VFDAS can be increased by approximately 28.4% from 0.005302 mm to 0.003796 mm by defining the physics collision surface of the semi-circle arch with 30 convex sub-part meshes instead of 5 convex sub-part meshes. In contrast, the rendering speed related to the VFDAS system performance was reduced by approximately 60.4% from 73.1 frames per second to 28.9 frames per second on average.

Using the same method presented above, while the concave semi-circle arch remains the original shape without subdivision, the accuracy of collision detection was determined as 0.07092 mm. The corresponding rendering rate was averagely 81.9 frames per second. While the semi-circle arch was subdivided into 15 convex sub-parts, the accuracy of collision detection was determined as 0.004627 mm. The corresponding rendering rate was averagely 52.6 frames per second.

In general, the minimum rendering frame rate of any VR simulation should be higher than 25 frames per second. The good performance of a VR system normally requires that the rendering frame rate can be higher than 50 frames per second so as to maintain real-time 3D collision detection. On the other hand, the process of slicing the semi-circle arch into a great number of small sub-part meshes using Rhino package is very time-consuming for the VE developer. As a result, the tradeoff between the accuracy of collision detection and rendering speed in this case should not be subdividing this concave semi-circle arch into more than 15 sub-part meshes so that the VFDAS can provide sufficient accuracy of collision detection as well as operate with a good system performance.

In addition, the results of the comparison between the number of divided sub-part meshes, the accuracy of collision detection and rendering speed with regard to the concave semi-circle arch can be seen in the table below **(Fig.7-13)**:

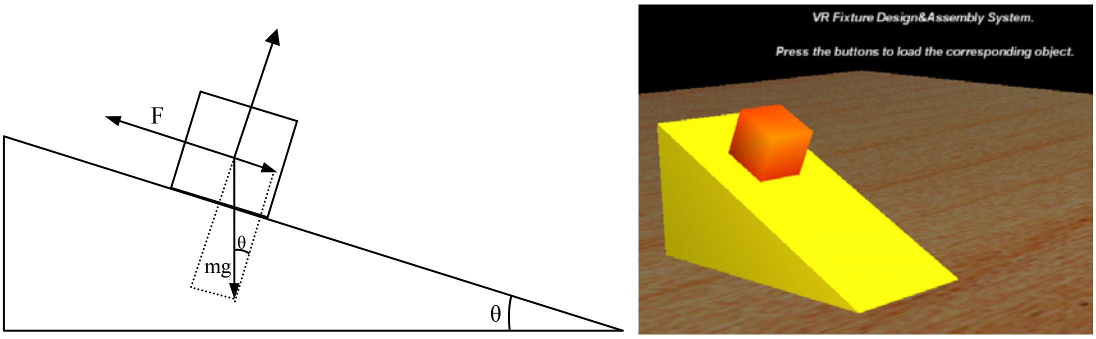
Sub-parts (Meshes)	1	5	15	30
Accuracy (mm)	0.07092	0.005302	0.004627	0.003796
Rendering Speed (FPS)	81.9	73.1	52.6	28.9

Fig.7-13. The Comparison Results for Sub-parts, Accuracy and Frame Rate

7.4 Friction

In order to validate and quantify the simulation result of friction in VFDAS system, the typical example with regard to a standard cube and a triangular prism may be used and presented in this section **(Fig.7-14)**. The triangular prism is a fixed physical object on the ground and the cube is placed on the sloped surface of the triangular prism, which trends to slide down under the gravity effect in the physics virtual environment.

In this example, F represents the friction between the cube and the triangular prism. θ represents the slope angle of the triangular prism. m represents the mass of the cube. g represents the gravitational constant. u represents the coefficient of friction between the cube and triangular prism while these two contacting surfaces are sliding against each other **(Fig.7-14)**. In addition, coefficient of friction is a measure of whether a surface is smooth or rough and defines the friction properties of the physics material associated with the object within VFDAS system.



VR Fixture Design&Assembly System.

Press the buttons to load the corresponding object.

Fig.7-14. The Example of Friction Simulation Result in VFDAS

While θ constantly increases in angle value and reaches a certain value within VFDAS, the cube will start sliding along the slope against the triangular prism surface. Before the cube slides, it is in force equilibrium under gravity, friction and the support force from the triangular prism and then the trigonometric function equation indicated below would be valid according to Newton's law.

$$
\begin{aligned}
&F = m \cdot g \cdot Sin\theta = m \cdot g \cdot Cos\theta \cdot u \\
&Sin\theta = Cos\theta \cdot u \qquad\qquad\qquad (1) \\
&u = Tg\theta
\end{aligned}
$$

Assume u = 0.49, then $\theta = Arctg(u) = Arctg0.49 = 26.105^{\circ}$

Theoretically the cube will start sliding along the slope surface of the triangular prism as soon as the θ is increased over 26.105 degrees.

In behaviour programming process based on Virtools, the final friction coefficient between two objects is defined by two values which have to be separately assigned to these two objects by VR programmers. The resulting value of this friction coefficient should multiply these two values together. As long as the resulting value is the same as realistic coefficient of friction between these two objects, the friction simulation in VFDAS can not tell the difference with the real physical world. The users of VFDAS are not involved in this programming process and they are not supposed to input any value to define the coefficient of friction during their use of VFDAS.

Within the VR environment two friction coefficients have to be assigned to the two contacting surfaces respectively rather than only one friction coefficient as general practices in real life. u_1 represents the coefficient of friction of the cube surface and u_2 represents the coefficient of friction of the triangular prism surface. The final coefficient of friction between the cube and triangular prism follows the formula of $u = u_1 \times u_2$. In this example, the u_1 was assigned as the value of 0.7 and the u_2 was also assigned as the value of 0.7. Therefore, u= 0.7 × 0.7 = 0.49.

In real application of the VFDAS system, while θ = 26.106, 26.107, 26.108 or 26.109, the cube still remains static. After this experiment could be conducted within VFDAS system, the actual result shows the cube does not move until the value of θ is increased to 26.110 degrees. Any angle value > 26.105 and < 26.110 is not recognized by the VFDAS system as the friction simulation is still not accurate enough to handle this realistic situation. As a result, the accuracy of the angle value related to friction simulation in VFDAS is $26.110^\circ - 26.105^\circ = 0.005^\circ$.

7.5 Mass, Gravity and Applied Force

In order to validate and quantify the simulation result of mass, gravity and applied force in the VFDAS system, a typical example with regard to three cuboids can be used and demonstrated in this section **(Fig.7-15)**. The blue cuboid at left hand side is a physical object without movements that is placed on the ground and acts as a fixed wall. The yellow cuboid at right hand side is a movable physical object that is placed on the ground under the gravity effect in the physics virtual environment. The red cuboid in the middle is a moveable physical object that is clamped in position by the other two cuboids.

In this example, F represents the external force applied to the yellow cuboid at the right side. M represents the mass of the red cuboid in the middle. g represents the gravitational constant. G represents the weight of the red cuboid. u represents the coefficient of friction between these cuboids **(Fig.7-15)**. In addition, mass is a quantitative measure of the inertia of an object and also the property of an object that results in gravitational attraction within VFDAS system. The weight of an object is defined as the force of gravity on the object.

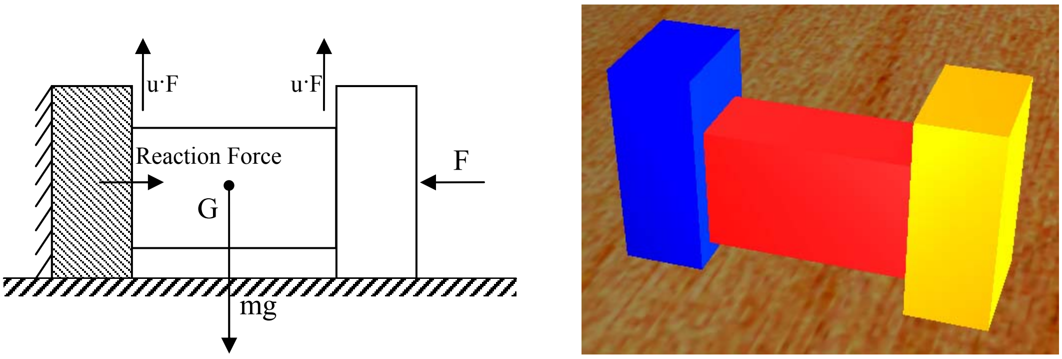


Fig.7-15. The Example Related to Mass, Gravity and Applied Force in VFDAS

In order to clamp this red middle cuboid in position in the physics virtual environment, a constant force needs to be applied to the yellow cuboid at right side so that the generated friction amongst these three cuboids is big enough to work against the gravity effect upon the red cuboid. The coefficient of friction for the ground was set as 0 in VFDAS, which indicates any object that makes a contact with the ground would not produce any friction in between. The minimum force that has to be applied while this red cuboid remains static in external force equilibrium is:

$$2u \cdot F = G = m \cdot g$$

$$F = \frac{m \cdot g}{2u} \qquad (2)$$

Because of u = 0.49, m =1, g = 9.8, so F = 10N

For 1kg mass red cuboid and 0.49 friction coefficient, theoretically the minimum force that has to be applied to the yellow cuboid at right side will be 10N so as to prevent the red middle cuboid sliding down and hold it in position.

Moreover, g was assigned as the default value of 9.8 although this value may be modified by the VE developer if necessary. Within the VR environment u_1 represents the coefficient of friction of the blue cuboid surface; u_2 represents the coefficient of friction of the red cuboid surface; u_3 represents the coefficient of friction of the yellow cuboid surface. The final coefficient of friction amongst these three cuboids follows the formula of $u = u_1 \times u_2$ and $u = u_2 \times u_3$. In this example, the u_1, u_2 and u_3 were all assigned as the value of 0.7. Therefore, u= 0.7 × 0.7 = 0.49.

In real application of the VFDAS system, while F = 10.000, 10.005, 10.010, 10.015 or 10.020, the middle red cuboid still slides down under the gravity effect. After this experiment could be implemented within VFDAS, the actual result demonstrates that the red cuboid can not be clamped in position until the value of F is increased to 10.026N. As a result, the accuracy of the force value applied to this clamping example in VFDAS system is $10.026\text{N} - 10.000\text{N} = 0.026\text{N}$, which is related to the aspects of mass, gravity, friction and applied force.

On the other hand, this example can reflect the real fixture design application where the friction is the major force used to constrain the workpiece and hold it in force equilibrium.

7.6 Elasticity

Within VFDAS, elasticity represents the property of a physics surface that makes the object more or less bounce against another object. It also defines the bouncing property of the physics material associated the object. Thus, the property is also known as bounciness. The elasticity in VFDAS is a different concept from the deformation in Finite Element Analysis. The deformation simulation can not be applied to the present version of VFDAS but possibly the future development.

In order to illustrate the simulation result of elasticity in VFDAS system, a typical example with regard to a physical bouncing ball against the ground may be presented **(Fig.7-16)**. A physical ball falls down from a height under the gravity effect in the physics virtual environment and it bounces back after it collides with the ground. In this example, H represents the initial height of the ball. h_n represents the height of the ball after n time collisions with the ground. n represents how many times the ball collides with the ground. α represents the elasticity factor of the ball. The elasticity factor in VFDAS defines how many percentages of energy the physics object can remain after one bounciness takes place such as: the value of 0.8 stands for the object may keep 80% energy left after a bounce against another object. The value of 1.0 stands for the object may keep 100% energy left after a bounce, which means the object never stops bouncing.

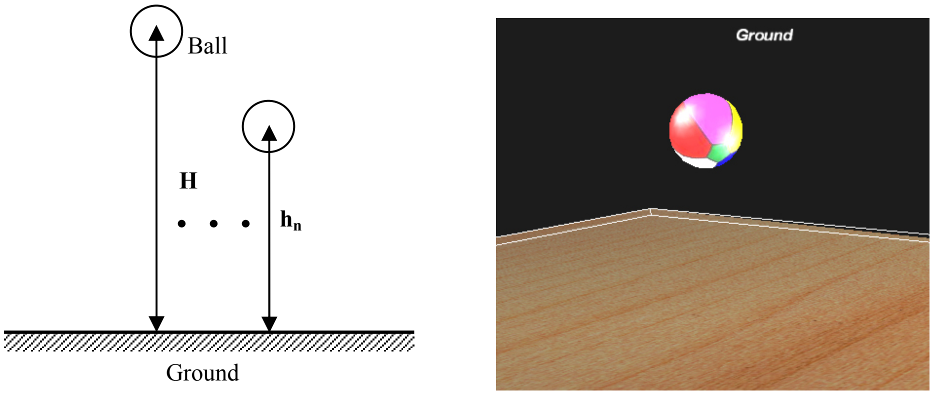


Fig.7-16. The Example of Elasticity Simulation Result in VFDAS

Assuming there is no air resistance in this simulation, the dampening force in the physics environment of VFDAS was specified as the value of 0. Thus, the motion mechanism of the ball within VFDAS will follow the physics equation shown below.

$$h_n = H \cdot \alpha^n \qquad (3)$$

In theory, according to this equation while H = 600mm, α = 0.8, the height of the ball after 6 time collisions with the ground should be h_1 = 480 mm, h_2 = 384 mm, h_3 = 307.2mm, h_4 = 245.76 mm, h_5 = 196.608 mm, h_6 = 157.2864 mm.

In real application of the VFDAS system, while H = 600mm, α = 0.8, the height of bouncing ball, h_1 = 478.94 mm, h_2 = 383.05 mm, h_3 = 306.32 mm, h_4 = 244.97 mm, h_5 = 195.08 mm, h_6 = 155.91 mm. These resulting values of the height derived from the experiment are approximately equal to the theoretical values presented above. As a result, it can be concluded that the motion mechanism of the bouncing ball within VFDAS could comply with the physics equation (3) reasonably well.

These experimental results can be obtained by exporting the animation of the bouncing ball from VFDAS into 3D Studio Max and then by calculating the height of the ball at the culminating point after each bounce using the ‘measurement’ tool of 3D Studio Max.

7.7 Toppling

The VFDAS system allows the toppling simulation of objects as if in the real physics world. In order to validate the simulation result of toppling in VFDAS system, a typical example with regard to a cube and a cuboid can be used and demonstrated in this section **(Fig.7-17)**. The cube is a fixed physical object on the ground and the cuboid is placed on the top surface of the cube.

The physical cuboid is gradually moved towards the right edge of the cube from the left using the interactive manipulation function provided by VFDAS. When the center of mass of the cuboid goes outside the boundary of the right edge that is indicated as a dashed line in the graph **(Fig.7-17)**, the cuboid starts toppling against the edge of the cube under the gravity effect in the physics virtual environment. The sequential screenshots of the toppling simulation of the cuboid are shown below **(Fig.7-18)**.

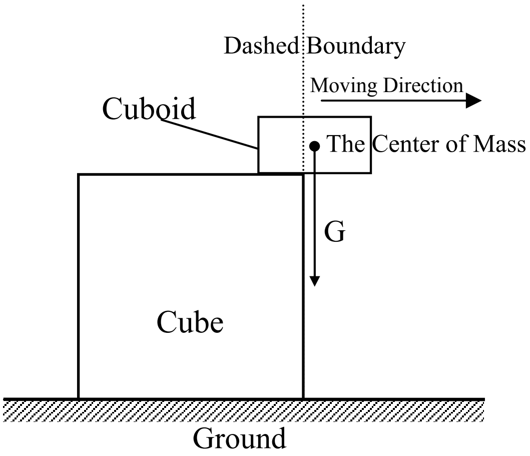
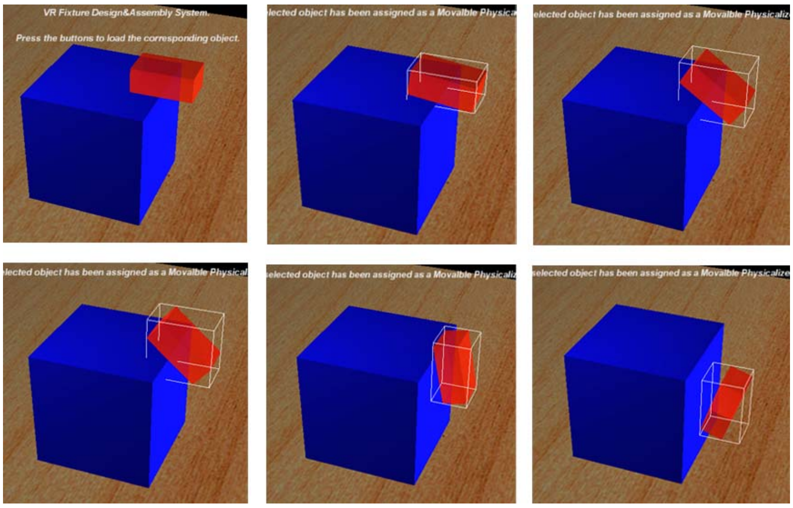


Fig.7-17. The Example of Toppling Simulation Result in VFDAS


VR Fixture Design&Assembly System.

Press the buttons to load the corresponding object.

lected object has been assigned as a Movalble Physica

elected object has been assigned as a Movalble Physicaliz

lected object has been assigned as a Movalble Physical

elected object has been assigned as a Movalble Physicaliz

elected object has been assigned as a Movalble Physicalize

Fig.7-18. The Sequential Screenshots of Toppling Simulation for the Cuboid

The center of mass of an object in the VFDAS system normally defaults to the geometric center of the object i.e. the centroid. However, the center of mass can be modified and shifted by the VE developer if the object in VFDAS does not represent uniform density.

7.8 Other System Limitations of VFDAS

Some limitations were reported throughout the real application process of VFDAS system by potential users and the VE developer, which need to be addressed as follows:

1. While two unphysicalized objects penetrate or intersect each other, the users can not assign the physical properties to both of objects at the same time because it would result in the computational errors or system crash.

2. Computationally the numerical registry can only contain 9 digit bits in the VFDAS system. Obviously, performing mathematical operations at this limit might lead to truncation of values. To some extent, it might also lead to inaccuracy of system simulations. In fact, the numbers used in simulation examples of the current VFDAS normally range from 999 to 0.001 which have not yet resulted in any truncation of values.

3. The users should not assign the physical properties to more than 10 objects simultaneously because this might cause that the rendering speed becomes very slow especially for complex concave objects. The users can unphysicalize or fix the objects in position after completing the fixture part assembly one by one so as to minimize CPU load.

7.9 Minimum Requirements for Computer Platform of VFDAS

Hardware

Pentium III or equivalent

256 MB of RAM

Monitor capable of displaying 1024 by 768 in 16 bit colour
Input device (computer keyboard)
Pointing device (mouse, trackball...)
Direct3D or OpenGL compatible 3D graphic card with 64 MB of RAM

Software
Microsoft Windows (2000, XP, Vista)
Microsoft DirectX 9.0C for DirectX compatible 3D graphic accelerator cards
For OpenGL, an OpenGL 2.0 compatible graphics card and driver
Microsoft Internet Explorer 6.0 (for the Online Reference)

7.10 Conclusions

In this chapter, the VFDAS system was validated in terms of collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results were presented and quantified by using a series of simple examples.

As a result, the accuracy of collision detection generally depends on how to build the optimal physics collision surface of a 3D object to represent its geometry. The physics collision surface may be defined by one or more designated meshes. The accuracy of collision detection for a concave object depends on how this concave surface is subdivided into the convex sub-part meshes. There is a tradeoff between the accuracy of collision detection and rendering speed needed to be identified. Even if the dimension and scale of some objects are changed in the VFDAS system, the accuracy of the collision detection results would remain the same as long as their geometric representations keep the same. Furthermore, the accuracy of the angle value related to friction simulation in VFDAS was 0.005°. The accuracy of the force value applied to the clamping example in VFDAS was 0.026N, which was related to the simulation aspects of mass, gravity, friction and applied force. The motion mechanism of the physical bouncing ball in VFDAS system would follow the physics equation (3) presented above. The toppling simulation of objects can be provided by the VFDAS system as if in the real physics world.

In general, the VFDAS system was validated as a useful computer aided fixture design system which can provide an acceptable quality of simulation results of physics properties and real-time 3D collision detection in accuracy.

Chapter 8. Conclusions and Future Recommendations

8.1 Introduction

The previous chapter has technically validated the VFDAS system in terms of collision detection, rendering speed, friction, mass, gravity, applied force, elasticity and toppling. These simulation results has been discussed and quantified by using a series of simple examples to show what the system can achieve and what the limitations are. This chapter will give rise to a summary of the conclusions with regard to the research in this thesis as well as declare the main contributions to the aims and objectives of the research. Finally, the recommendations for future research and the potential issues arising from developing the future version of VFDAS are also presented.

8.2 Conclusions

Along with developing the useful Virtual Reality Fixture Design & Assembly System (VFDAS), the research has demonstrated that the advantages of VR can be used to improve the existing computer aided fixture design (CAFD) as well as support the conventional fixture design and assembly process. As a result of the work done in this thesis, the following conclusions can be drawn.

1. The VFDAS system is able to provide an acceptable quality of real-time 3D collision detection in accuracy which comprises the collision detection of convex objects and concave objects. The accuracy of collision detection in VFDAS is good enough to support the design interference check between fixture elements, workpiece and machine tools during a fixture design process. More details with regard to the numerical accuracy of collision detection simulation can be seen in section 7.2 in the validation chapter.

2. The accuracy of collision detection in the VFDAS system may be improved by subdividing the complex concave object into a group of convex sub-parts.

However, there is a tradeoff between the accuracy of collision detection and rendering speed needed to be identified so that the VFDAS system not only provides a sufficient accuracy of collision detection result but also operates with good system performance. More explanation with regard to the tradeoff can be seen in section 7.3.

3. The VFDAS system provides a good quality of friction simulation in the physics virtual environment. According to the typical example, the accuracy of the angle value related to friction simulation in VFDAS was identified as 0.005 degrees. More details in relation to this example of friction simulation can be found in section 7.4 in Chapter 7.

4. The VFDAS system can also provide a good quality of simulation results for mass, gravity and applied force. According to the typical example that relates to the aspects of mass, gravity, friction and applied force, the accuracy of the force value applied to this clamping example in VFDAS system was determined as 0.026N. More details with regard to this clamping example can be found in section 7.5.

5. The VFDAS system can also provide an acceptable quality of elasticity simulation result in the physics virtual environment. According to the typical example of a virtual bouncing ball against the ground, it concluded that the motion mechanism of the bouncing ball in VFDAS can obey the physics equation $h_n = H \cdot \alpha^n$ reasonably well. More details with regard to this example of elasticity simulation can be found in section 7.6.

6. The VFDAS system allows the toppling simulation of objects as if in the real physics world. According to the typical example with regard to a cube and a cuboid, the simulation result of toppling in VFDAS was demonstrated using a series of sequential screenshots. More details with regard to this example of toppling simulation can be seen in section 7.7.

7. Other three limitations of the current VFDAS system were identified by users and the VE developer during the real application process. More details in relation to these clarified limitations can be found in section 7.8.

8.3 Contribution to Aims and Objectives

The main aims and objectives of this research, as stated in section 1.4, were met with the development of the VFDAS system for supporting the conventional fixture design and assembly process. Very few CAFD research adopts the interactive VR technology to improve the fixture design and assembly process and there is very little evidence of research exploring the use of VR for fixture and assembly. Due to the novel VFDAS system being developed, the main contribution in this research is considered as the research gap between VR and Fixture Design can be narrowed down by the author.

Besides the main contribution, more specific contributions to the main aims and objectives presented in this thesis were as follows:

1. The current VFDAS system can provide fixture designers with a platform to virtually implement interactive VR fixture design and assembly to generate the final fixture assembly as if in the real physics world. More details can be seen in section 5.3.4 in Chapter 5.

2. The VFDAS system achieved rapid evaluation and iteration of multiple fixture design ideas to support the problem definition in a virtual environment. More details can be seen in section 5.3.5.

3. Comparing the existing CAFD systems, the VFDAS can provide fixture designers with the relative intuitive manipulation and natural manner to conduct the fixture design and assembly process. The keyboard and mouse were the main control devices for users in current VFDAS. More details can be seen in section 5.3.4.

4. The VFDAS system comprised a user-friendly library which contains a number of modular fixtures to provide fixture designers with more alternatives for the element selection and facilitate the fixture design process. Although the current VR library only consists of the 15 standard fixturing elements, more other elements models will be gathered and included into it in the future. More details can be seen in section 5.3.2.

5. The VFDAS system was able to generate the appropriate visualization and 3D graphical representation for fixture elements during the fixture design and assembly process. More details can be seen in section 5.3.8.

6. The VFDAS system achieved the assembly process planning for fixtures using interactive VR simulation and provided the function of recording the animation for the virtual fixture assembly. Thus, the fixture assembly sequence, procedure and trajectory for each fixture element could be determined. More details can be seen in section 5.3.12.

7. The VFDAS system was able to generate a bill of materials (BOM) for fixture assembly and perform the cost analysis. More details can be seen in section 5.3.15.

8. The VFDAS system provided the capability of simulating real-time 3D collision detection in accuracy and physical properties e.g. friction, mass, gravity, applied force, elasticity and toppling. More details can be seen in Chapter 7 and the explanation about how to control and manage the physical properties can be found in section 5.3.3.

9. The VFDAS system can analyze the machining strategy and produce the cutting path envelope before implement the fixture layout design to avoid design interference. More details can be seen in section 5.3.14.

10. The VFDAS system acted as an effective role of communication between fixture designers and people from different areas to identify the fixture assembly errors. More details can be found in section 5.3.12.

11. The VFDAS system was able to improve fixture productivity and economy. The physical fixtures are not necessary involved in the design and verification. The fixture inventory and labour cost were reduced by using the VFDAS.

12. The current VFDAS system was able to perform the kinematics analysis for a simple example. More details can be found in section 7.4.

13. The VFDAS can reversely transfer the final fixture assembly model back to the CAD system so that it is ready for the manufacturing process. More details can be found in section 5.3.15.

14. The current VFDAS was developed by incrementally combining many micro focused functions of fixture design into the system so as to establish a comprehensive fixture design system. The VFDAS system comprises the functionality of fixture element selection, fixture layout design, assembly planning and relevant analysis.

8.4 Recommendations for Future Research

The development of the VFDAS system was an incremental developing process which was divided into three different phases. A series of research directions in relation to the future development of VFDAS may be continuously explored in future work. In terms of these research directions, more system functionality can be developed and incorporated in to the existing framework of VFDAS established at the development Phase Three. More details with regard to the development Phase Three can be found in previous section 6.5. As a result, the recommendations for the future research with regard to the further VFDAS development are summarized as follows.

1. The development of VFDAS system in this research focused on fixture planning and fixture configuration design rather than multi-setups planning. Therefore, the setup planning aspect of a fixture design will be the potential research direction that is worthwhile being investigated in future research. This research

direction should be based on the existing framework of VFDAS system to incorporate the function of managing setup planning into the VFDAS.

2. Some existing research has addressed the importance of developing computer-automated fixture design (CAFD) approaches. The research direction of computer semi-automated or automated fixture design will be the interesting study area to be explored in future research so that these functions can be included into the future version of VFDAS system.

3. The functionality of kinematics analysis in the current VFDAS system needs to be further developed in future research so that the VFDAS can be used to deal with more complex fixture design cases in the future.

4. The research direction related to the workpiece handling, ergonomics, and safety factors should be taken into account in the future research of the VFDAS development so as to incorporate the function of ergonomics evaluation into the existing framework of VFDAS system.

5. Tolerance analysis related to a fixture design process will also be a potential research direction that can be further investigated in future research in order to incorporate the function of tolerance analysis into the current VFDAS system.

6. Finite element analysis (FEM) is related to the deformation of the workpiece involved in a fixture design process. This function may be further developed and included into the future version of VFDAS system. Thus, Finite element analysis can be considered as another research direction which should be undertaken by the researcher in future work.

7. In order to improve the function of intuitive manipulation and interaction in the future version of VFDAS system, the use of haptic data gloves or instrumented cyber gloves associated into the VFDAS system will be another potential research direction. This research direction is worthwhile being further investigated in future research based on the existing framework of VFDAS system.

8. In order to achieve the functional goal of high-level immersion and presence in the future version of VFDAS system, the use of external devices e.g. Head-mounted display or Cyber glasses associated into the VFDAS system will be another potential research direction.

9. The fixture data management and rule-based knowledge database will be another potential research direction. This research direction is worthwhile being continuously explored and investigated in future research so as to improve the capability of the pre-defined VR library in the future version of VFDAS.

References

[1] Ping Ji, Albert C. K. Choi, Lizhong Tu (2002). "*VDAS: a virtual design and assembly system in a virtual reality environment.*" Assembly Automation 22(4): 337-342.

[2] Yu, K.M., Lam, T.W., Lee, A.H.C. (2003). "*Immobilization check for fixture design*" Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 217(4): 499-512.

[3] Tesic, R., Banerjee, P. (2001). "*Exact collision detection for a virtual manufacturing simulator*" IIE Transactions (Institute of Industrial Engineers) 33(1): 43-54.

[4] Cecil, J. (2001). "*Computer-aided fixture design - A review and future trends.*" International Journal of Advanced Manufacturing Technology 18(4): 790-793.

[5] Hargrove, S.K., Kusiak, A. (1994). "*Computer-aided fixture design: a review.*" International Journal of Production Research 32(4): 733-753.

[6] Hou, J.L., Trappey, A.J.C. (2001). "*Computer-aided fixture design system for comprehensive modular fixtures.*" International Journal of Production Research 39(16): 3703-3725.

[7] Lim, B.S., Imao, T., Yoshida, H., et al. (1992). "*Integrated modular fixture design, pricing and inventory control expert system*" International Journal of Production Research 30(9): 2019-2044.

[8] Surendra Babu, B., Madar Valli, P., et al. (2005). "*Automatic modular fixture generation in computer-aided process planning systems*" Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 219(10): 1147-1152.

[9] Wang, Y. (2004) "*A methodology of fixture evaluation, analysis and optimization*". PHD Thesis, University of Nottingham.

[10] Rong, Y.M., Bai, Y. (1997). "*Automated generation of fixture configuration design*" Journal of Manufacturing Science and Engineering, Transactions of the ASME 119(2): 208-219.

[11] Dai, J.R., Nee, A.Y.C., et al. (1997). "*An Approach to Automating Modular Fixture Design and Assembly*" Proceedings of the I MECH E Part B Journal of Engineering Manufacture 211(7): 509-521.

[12] Nee, A.Y.C., Whybrew, K., et al. (1995). "*Advanced Fixture Design for FMS*" Springer-Verlag, UK.

[13] Kow, T.S., Kumar, A.S., Fuh, J.Y.H. (2000). "*An integrated approach to collision-free computer-aided modular fixture design*" International Journal of Advanced Manufacturing Technology 16(4): 233-242.

[14] Tung, K.H., Poo, A.N., Nee, A.Y.C. (1990). "*A modular fixture configuration design system using a rule/object based expert system approach*". Proceedings of the IA 90 Conference, Singapore, pp. 568-599.

[15] Koh, S.L. (1990). "*Intelligent modular fixture design system*". Nanyang Technological Institute, School of Mechanical and Production Engineering, unpublished B.Sc. undergraduate thesis.

[16] Noe, D., Peklenik, J. (1991). "*A knowledge based planning of clamping in turning operations*". Proceedings of the International Conference on CIM, edited by B.S. Lim (Singapore: World Scientific), pp. 530-533.

[17] Lim, B.S., Knight, J.A.G. (1986). "*HOLDEX-holding device expert system*". Proceedings of the 1st International Conference on Applications of AI in Engineering Problems, Southampton (Springer-Verlag), pp. 483-502.

[18] Ma, W., Lei, Z., Rong, Y. (1998). "*FIX-DES: A computer-aided modular fixture configuration design system*" International Journal of Advanced Manufacturing Technology 14(1): 21-32.

[19] Dai, Fan, ed. (1998). "*Virtual Reality for Industrial Applications.*" Berlin, Springer.

[20] Vince, John. (1995). "*Virtual Reality Systems.*" Workingham, Addison-Wesley.

[21] Pausch, R., Proffitt, D., Williams, G. (1997). "*Quantifying immersion in virtual reality*". Proceedings SIGGRAPH, ACM. 1997.

[22] Bowman, D.A., McMahan, R. (2007). "*Virtual Reality: How much immersion is enough?* ". IEEE Computer Mag., July 2007.

[23] Chryssolouris, G., V. Karabatsou, D. Mavrikios, D. Fragos, K. Pistiolis, Petrakou, H. (2000). "*A virtual environment for assembly design and training*". Proceedings of the 33rd International CIRP Seminar on Manufacturing Systems, Stockholm, Sweden, (June 2000), pp. 326-330.

[24] Chryssolouris, G., Mavrikios, D., Fragos, D., Karabatsou, V., Pistiolis, K. (2002). "*A novel virtual experimentation approach to planning and training for manufacturing processes-the virtual machine shop*". International Journal of Computer Integrated Manufacturing, Vol.15, No.3, pp. 214-221.

[25] Mavrikios, D., V. Karabatsou, D. Fragos, Chryssolouris, G. (2006). "*A prototype virtual reality based demonstrator for immersive and interactive simulation of welding processes*". International Journal of Computer Integrated Manufacturing, Vol.19, No.3, pp. 294-300.

[26] Li, Q., Chen, X., Cobb, S., Eastgate, R. (2005). "*Virtual reality applications in fixture assembly and interactive simulation*". Proceedings of the 11th Annual Conference of the Chinese Automation and Computing Society in the UK (CACSUK05), Sheffield, UK, September 10, pp.183-188.

[27] Rong, Y., Huang, S.H., Hou, Z. (2005). "*Advanced Computer-Aided Fixture Design*". Boston, MA: Elsevier, pp. 94-132.

[28] Jezernik, A., Hren, G. (2003). "*A solution to integrate computer-aided design (CAD) and virtual reality (VR) databases in design and manufacturing processes*." Int J Adv Manuf Technol 22(11-12):768–774.

[29] Sheridan, T. B. (1992). "*Defining Our Terms*". Presence: Teleoperators and Virtual Environments, 1(2), 272-274.

[30] Steuer, J. (1992). "*Defining virtual reality: Dimensions determining telepresence*". Journal of communication, 42(4), 73-93.

[31] Griffiths, G.D. (2001). "*Virtual environment usability & user competence: the Nottingham Assessment of Interaction within Virtual Environments (NAÏVE) tool*". PhD Thesis, University of Nottingham, UK.

[32] Winn, William. (1997). "*The Impact of Three-Dimensional Immersive Virtual Environments on Modern Pedagogy*". HITL Technical Report R-97-15, Human Interface Technology Laboratory, University of Washington. [http://www.hitl.washington.edu/publications/r-97-15/r-97-15.rtf] Visited Aug 15th 2005.

[33] Zelter, D. (1992). "*Autonomy, Interaction, and Presence.*" Presence. 1(1): 127-132.

[34] Ong, S. K., Mannan, M. A. (2005). "*Virtual reality simulations and animations in a web-based interactive manufacturing engineering module*". Computers and Education 43(4): 361-382.

[35] Bell, John T., Fogler, H. Scott. (1997). "*Ten steps to developing virtual reality applications for engineering education*". ASEE Annual Conference Proceedings, Milwaukee, USA.

[36] Crosier, J. K. (2000). "*Virtual Environments for Science Education: A Schools-Based Development.*" PhD Thesis, University of Nottingham, Nottingham.

[37] Messner, J. I., Yerrapathruni, S., Baratta, A. J. and Whisker, V.E., (2003), "*Using Virtual Reality to Improve Construction Engineering Education,*" Proceedings of the 2003 ASEE Annual Conference, Nashville, TN, 8 pages.

[38] Kalisperis, L. N., Otto, G., Muramoto, K., Gundrum, J. S., Masters, R., Orland, B. "*Virtual Reality Space Visualization in Design Education*". Proceedings of the 2002 eCAADe 20th Conference, Warsaw University of Technology, Warsaw.

[39] Lemay, L. (1997). *Teach yourself web publishing with HTML 4 in a week* (4th ed.). Sams.net Publishing.

[40] Sackett, P. and McCluney, D. (1992). "*Inter enterprise CIM - a mechanism for graduate-education".* Robotics and Computer-Integrated Manufacturing 9(1): 9-13.

[41] Dessouky, M. M., Verma, S., Bailey, D. E., Rickel, J. (2001) "*A methodology for developing a web-based factory simulator for manufacturing education.*" IIE Transactions 33(3): 167-180.

[42] Steffan, R., Schull, U., Kuhlen, T. (1998). "*Integration of virtual reality based assembly simulation into CAD/CAM environments.*" Proceedings: 24th Annual Conference IEEE Industrial Electronics Society, Aachen, 4, pp. 2535-2537.

[43] Dewar, R. G., et al. (1997). "*Assembly Planning in a Virtual Environment.*" Proceedings: Portland Int'L Conf.on Management of Engineering and Technology (PICMET 97), IEEE Press, Piscataway, N.J., pp. 664-667.

[44] Jayaram, S., Jayaram, U., Wang, Y., Tirumali, H., Lyons, K., Hart, P. (1999). "*VADE: A Virtual Assembly Design Environment.*" IEEE Computer Graphics and Applications 19(6): 44-50.

[45] Kaufman, S., et al. (1996). "*The Archimedes 2 Mechanical Assembly Planning System.*" Proceedings: IEEE Int'l Conf. on Robotics and Automation, IEEE Press, Piscataway, N.J., pp. 3361-3368.

[46] Gottschalk, S., Lin, M. C., Manocha, D. (1996). "*OBB-Tree: A Hierarchical Structure for Rapid Interface Detection.*" Tech. Report TR96-013, Department of Computer Science, University of North Carolina, Chapel Hill.

[47] Jayaram, S., Connacher, H. I., Lyons, K. W. (1997). "*Virtual assembly using virtual reality techniques.*" Computer-Aided Design 29(8): 575-584.

[48] Liu, J. S., Yao, Y. X. (2004). "*Preliminary investigation of virtual assembly constructing.*" Assembly Automation 24(4): 379-385.

[49] Biermann, P., Jung, B., Latoschik, M., Wachsmuth, I. (2002). "*Virtuelle Werkstatt: A Platform for Multimodal Assembly in VR.*" Proceedings: Fourth Virtual Reality International Conference (VRIC 2002), Laval, France, pp.53-62.

[50] Williams, D. L., Finke, D. A., Medeiros, D. J., Traband, M. T. "*Discrete Simulation Development for a Proposed Shipyard Steel Processing Facility.*" Proceedings: the 2001 Winter Simulation Conference, Piscataway, New Jersey, pp. 882-887.

[51] Albastro, M. S., Beckman, G., Gifford, G., Massey, A. P., Wallace, W. A. (1995). "*The Use of Visual Modeling in Designing a Manufacturing Process for Advanced Composite Structures.*" IEEE Transactions on Engineering Management 42(3): 233-242.

[52] Kibira, D., McLean, C. (2002). "*Virtual Reality Simulation of A Mechanical Assembly Production Line.*" Proceedings: The 2002 Winter Simulation Conference, San Diego, CA, United States, pp. 1130-1137.

[53] Nomura, J., Sawada, K. (2001). "*Virtual Reality Technology and Its Industrial applications.*" Annual Reviews in Control 25: 99-109.

[54] Shinomiya, Y. et al. (1997). "*Horse Riding Therapy Simulator with VR Technology.*" Symposium on Virtual Reality Software and Technology 1997- VRST'97, ACM press, pp. 9-14.

[55] Yamamura, A. et al. (1996). "*Kitchen Layout Design in Virtual Environments.*" Proceedings: The 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference.

[56] Simmons, J., Ritchie, J. (2002). "*Human in the Loop: The Use of Immersive Virtual Reality to Aid Cable Harness Design.*" Proceedings: the 1st CIRP(UK) Seminar on Digital Enterprise Technology, School of Engineering, University of Durham, UK, pp. 109-112.

[57] Murray, N., Fernando, T. (2004). "*An immersive assembly and maintenance simulation environment.*" Proceedings: Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications, Budapest, Hungary, pp. 159-166.

[58] Li, J., Ma, W., Rong, Y. (1999). "*Fixturing surface accessibility analysis for automated fixture design*" International Journal of Production Research 37(13): 2997-3016

[59] Trappey, A.J.C., Liu, C.R. (1990). "*A literature survey of fixture-design automation*" International Journal of Advanced Manufacturing Technology 5(3): 240-255

[60] Wu, Y., Rong, Y., Ma, W., LeClair, S.R. (1998). "*Automated modular fixture planning: accuracy, clamping, and accessibility analyses*" Robotics and Computer-Integrated Manufacturing 14(1): 17-26

[61] Callele, D., Carthy, C.Mc. (2004). "*Virtools Dev User Guide*", Virtools SA.

[62] Superscape (1998). "*VRT for windows User Guide*", Superscape, England.

[63] Kerr, S. J. (2005). "*Developing scaffolded virtual learning environments for people with autism*". PHD Thesis, University of Nottingham.

[64] Reynard, G., Eastgate, R. (2001). "*Suppporting Working adults with Asperger's Syndrome − Technology, Platforms and Internet Resources*". AS Interactive internal project report, University of Nottingham.

[65] Snowden, D. N., West, A. J., Howard, T. L. J. (1993). "*Towards the next generation of Human-Computer Interface*". In proceedings of Informatique' 93: Interface to Real & Virtual Worlds, Montpellier, France, 399-408.

[66] Greenhalgh, C. (1997). "*Large Scale Collaborative Virtual Environments*". Unpublished PhD, University of Nottingham, Nottingham.

[67] Kessler, G. D., Bowman, D. A., Hodges, L. F. (2000). "*The Simple Virtual Environment Library: An Extensible Framework for Building VE Applications*". Presence 9(2): 187-208.

[68] Boyd, D., Sastry, L. (1999). "*Development of the INQUISITIVE Interaction Toolkit - Concept and Realisation*". In proceedings of User Centred Design and the Implementation of Virtual Environments, York, 1-6.

[69] Neale, H.R., Cobb, S.V., Kerr, S.J., Leonard, A. (2002). "*Exploring the role of Virtual Environments in the Special Needs Classroom*". In, Sharkey (Ed) Proceedings of the 4th International Conference on Disability, Virtual Reality and Associated Technologies (ICDVRAT), Veszprem, Hungary, September 2002, 259-266.

[70] Crosier, J. (2000). "*Virtual environments in science education: a schools-based development*". Unpublished PhD thesis, University of Nottingham, Nottingham.

[71] Neale, H., Cobb, S., Wilson, J. (2000). "*Designing virtual learning environments for people with learning disabilities: usability issues*". In proceedings of ICDVRAT 2000, Alghero, Sardinia, University of Reading, 265-272.

[72] Kaur Deol, K., Steed, A., Hand, C., Istance, H., Tromp, J. (2000). "*Usability evaluation for virtual environments: Workshop at De Montfort University*", Leicester, UK. Interfaces(44), 4-7.

[73] Newman, W., Lamming, M. (1995). "*Interactive Systems Design*". Reading, Mass.: Addison-Wesley.

[74] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Hooland, S., Carey, T. (1994). "*Human- Computer Interaction*".: Addison-Wesley Longman Ltd.

[75] Eastgate, R. (2001). "*The Structured Development of Virtual Environments: Enhancing Functionality and Interactivity* ". PHD Thesis, University of Nottingham.

[76] Stanney, K. M., Mollaghasemi, M., Reeves, L., Breaux, R., Graeber, D. A. (2003). "*Usability engineering of virtual environments (VEs): Identifying multiple criteria that drive effective VE system design*". International Journal of Human Computer Studies 58(4): 447-481

[77] Bowman, D. (1999). "*Interaction techniques for common tasks in immersive virtual environments: Design, evaluation, and application*". Unpublished doctoral dissertation, Georgia University of Technology, Georgia.

[78] Gabbard, J., Hix, D. (1997). "*A taxonomy of usability characteristics in virtual environments*". (Technical Report http://csgrad.cs.vt.edu/~jgabbard/ve/taxonomy): Virginia Polytechnic Institute and State University.

[79] Schneiderman, B. (1998). "*Designing the User Interface – Third Edition*", Addison-Wesly, New York

[80] Nielsen, J. (1994). "*Heuristic Evaluation*". in Nielsen, J. and Mack, R.L. (eds) Usability Inspection Methods, New York: John Wiley & Sons, pp. 25–62.

[81] Kaur, K. (1998). "*Designing virtual environments for usability*". Unpublished Doctor of Philosophy, City University, London.

[82] Wilson, J. R., Eastgate, R. M., D'Cruz, M. (2001). "*Structured Development of Virtual Environments*". In K. Stanney (Ed.), The Handbook of Virtual Environment Technology.

[83] Gabbard, J., Hix, D., Swan, J. (1999). "*User-centred design and evaluation of virtual environments*". IEEE Computer Graphics and Applications, 19, 51-59.

[84] Fencott, C. (In preparation). "*A Practical Content Model for Virtual Environment Design*". International Journal of Human-Computer Studies.

[85] Li, Q., Chen, X., Cobb, S., Eastgate, R. (2006). "*Physical behaviour simulation and exact collision detection within VFDAS*". Proceedings of the 12th Annual Conference of the Chinese Automation and Computing Society in the UK (CACSUK06), Loughborough, UK, September 16, pp.33-37.

[86] Cohen, J., Lin, M., Manocha, D., Ponamgi, K. (1995). "*I-COLLIDE: an Interactive and exact collision detection system for large scale environments*". In Proceedings of ACM International 3D Graphics Conference, pp. 189-196.

[87] Jayaram, S., Connacher, H.I., Lyons, K.W. (1997). "*Virtual assembly using virtual reality techniques*". Computer-Aided Design 29(8): 575–584.

[88] Lewis, G. (1983). "*Modular fixturing systems*". Proceedings of the 2nd International Conference on Flexible Manufacturing Systems, London, IFS (Publ) Ltd, pp. 451-461.

[89] Hammer, H., Gallien, D. (1983). "*Efficient and cost effective use of modular fixture kits at machine site*". TZ fur Metallbeatbeitung, 5.

[90] Markus, A. (1984). "*Fixture design using Prolog: an expert system*". Robotic and Computer Integrated Manufacturing 1(2): 167-172.

[91] Mervyn, F., Kumar, A.S., Nee, A.Y.C. (2005). "*Automated synthesis of modular fixture designs using an evolutionary search algorithm*". International Journal of Production Research 43(23): 5047-5070.

[92] Kuehne, R.P., Oliver, J.H. (1995). "*Virtual environment for interactive assembly planning and evaluation*". Proceedings of ASME Design Engineering Technical Conference, New York, USA, pp. 863-867.

[93] Garey, M., Johnson, D. (1979). "*Computers and Intractability: A Guide to the Theory of NP-Completeness*". Freeman Publishing, San Francisco, CA.

[94] O'Rourke, J., Supowit, K.J. (1983). "*Some NP-hard polygon decomposition problems*". IEEE Transactions on Information Theory, IT-29, 181-190.

[95] Richardson, D. (2005). "*Convex Hull*". (Technical Report http://www.cis.umassd.edu/~lshen/courses/2005_spring_cis467/notes/042605_04.pdf): Visited Jul 20th 2008, Computer and Information Science Department, University of Massachusetts Dartmouth.

[96] Shokri, M., Arezoo, B. (2008) "*Computer-aided CMM modular fixture configuration design.*" International Journal of Manufacturing Technology and Management 14(1-2): 174-188.

[97] Kang, Y.G., Wang, Z., Li, R., Jiang, C. (2007) "*A fixture design system for networked manufacturing.*" International Journal of Computer Integrated Manufacturing 20(2-3): 143-159.

[98] Peng, G.L., Liu, W.J. (2006). "*A novel modular fixture design and assembly system based on VR.*" Proceedings of 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, pp. 2650-2655.

[99] Ji, P., Lau, Francis K.H., Jiang, L.L., Li, M., Li, Z.B. (2007). "*Computer-aided generation of fixture configuration design using polychromatic sets.*" International Journal of Computer Applications in Technology 28(4): 289-294.

[100] Boyle, I.M., Rong, K., Brown, D.C. (2006). "*CAFixD: A case-based reasoning fixture design method. framework and indexing mechanisms.*" Journal of Computing and Information Science in Engineering 6(1): 40-48.

[101] Guo, H., Yan, X.G., Li, J.W. (2006). "*The study of a virtual modular fixture management and aid-design system.*" Proceedings of International Technology and Innovation Conference 2006 (ITIC 2006), Hangzhou, China, pp. 449-452.

[102] Mervyn, F., Kumar, A.S., Nee, A.Y.C. (2006). "*Fixture design information support for integrated design and manufacturing.*" International Journal of Production Research 44(11): 23-37.

[103] Peng, G.L., He, X., Yu, H.Q., Hou, X., Khalil, A. (2008) "*Precise manipulation approach to facilitate interactive modular fixture assembly design in a virtual environment.*" International Journal of Assembly Automation 28(3): 216-224.

APPENDIX A

1. Other Interaction Functions in Turbine Blade VR Simulation

In attempt to accomplish the interaction function of manually playing and controlling the turbine blade fixturing animation sequence, there have been three 2D Frames created and positioned on the top screen so as to become the three fundamental play buttons on this simulation interface. Pressing on the 'BACK' button can manually set the previous step of the animation sequence. Pressing on the 'NEXT' button can set the next step of the animation sequence. These two buttons can also be used to play the fixturing animation one frame by one frame and observe the detailed procedures how the workpiece can be gradually mounted upon the particular designed fixture mechanism. Alternatively, the potential users are allowed to play a full animation at once with clicking the control button called 'PLAY' **(Fig.A-1)**.

Fig.A-1. The Three Play Buttons on the Interface

As a result, the specific behaviour programming context that has been adopted to fulfil the interaction function of setting the previous or the next step of the animation sequence by respectively pressing on the 'BACK' button or the 'NEXT' button would be graphically manifested in the picture below **(Fig.A-2)**. There have been a series of different Behaviour Building Blocks (BBs) selected and involved to this behaviour programming process, which are all necessarily assigned to the 'Level' in the simulation. In addition, there is only one Level in a composition, called the default Level. As a Level acts as a global container, holding everything that is in a composition, any Place or Object presented in a Scene is also contained in the default Level. In general, the essential BBs and Parameter Operation undertaken to this behaviour programming process are respectively the BB named 'Set Global Animation Step', the BB named 'Identity', the BB named 'Wait Message', the BB named 'Op', the BB named 'Threshold' and the Parameter Operation called 'Multiplication'.

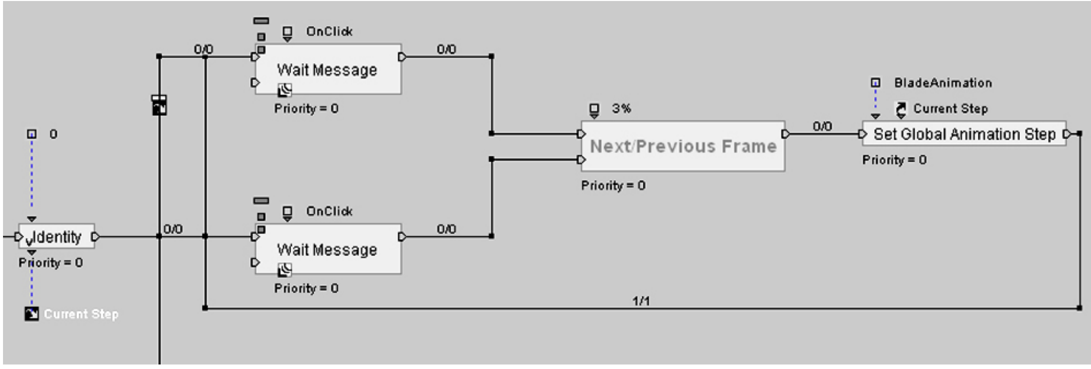
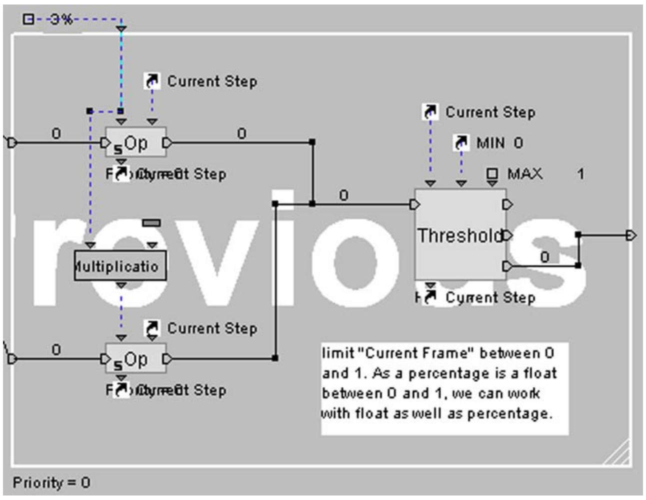


Fig.A-2. The Script for the Function of Setting the Previous/Next Step of the Animation

Furthermore, there is also an essential Behaviour Graph named ‘Next/Previous Frame’ included within the Behaviour Script. Hence, the BB named ‘Op’, the BB named ‘Threshold’ and the Parameter Operation called ‘Multiplication’ mentioned above have been encapsulated into this Behaviour Graph already. This expanded Behaviour Graph will be shown in the following graph **(Fig.A-3)**.



limit "Current Frame" between 0 and 1. As a percentage is a float between 0 and 1, we can work with float as well as percentage.

Fig.A-3. The Expanded Behaviour Graph Named ‘Next/Previous Frame’

In contrast, the specific behaviour programming syntax that has been employed to fulfil the interaction function of triggering and playing the whole animation of turbine blade fixturing at once by clicking the ‘PLAY’ control button would be evidently articulated in the following picture **(Fig.A-4)**. There have been a series of distinct BBs involved to this behaviour programming process, which are all essentially applied to the ‘Level’ in the simulation as well. In essence, the key BBs and Parameter Operation embarked to this behaviour programming process are respectively the BB named ‘Wait Message’, the BB named 'Set Global Animation Step', the BB named 'Restore IC', the BB named 'Identity', the BB named 'Delayer', the BB named 'While', the BB named 'Op' and the Parameter Operation called 'Superior'.

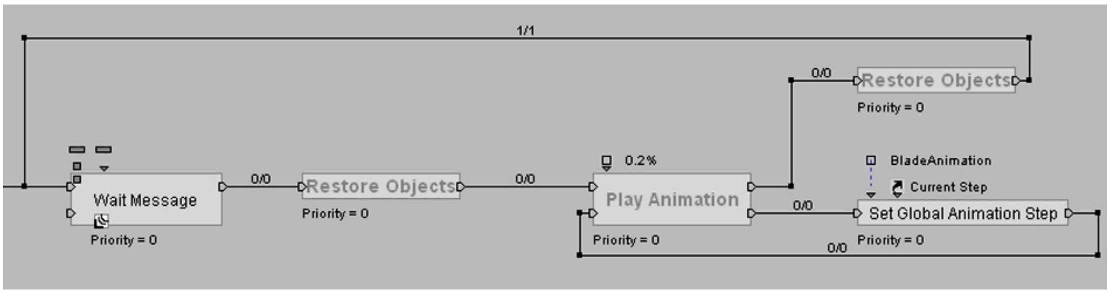


Fig.A-4. The Script for the Interaction Function of Playing the Animation

Moreover, there are also two type of fundamental Behaviour Graphs named 'Restore Objects' and 'Play Animation' comprised within the Script. Consequently, the BB named 'Delayer', the BB named 'While', the BB named 'Op' and the Parameter Operation called 'Superior' mentioned before have been included into the Behaviour Graph named 'Play Animation'. In addition, the BB named 'Restore IC' and the BB named 'Identity' declared above have been also encapsulated into the target Behaviour Graph named 'Restore Objects'. These two types of expanded Behaviour Graphs would be explicitly displayed in the following graphs **(Fig.A-5) (Fig.A-6)**.

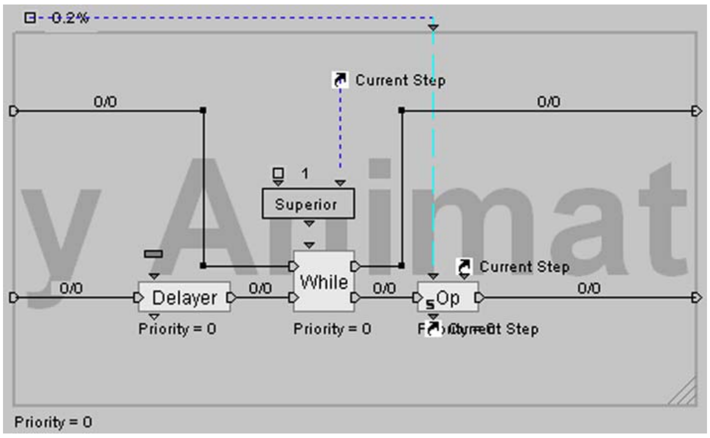


Fig.A-5. The Expanded Behaviour Graph Named 'Play Animation'

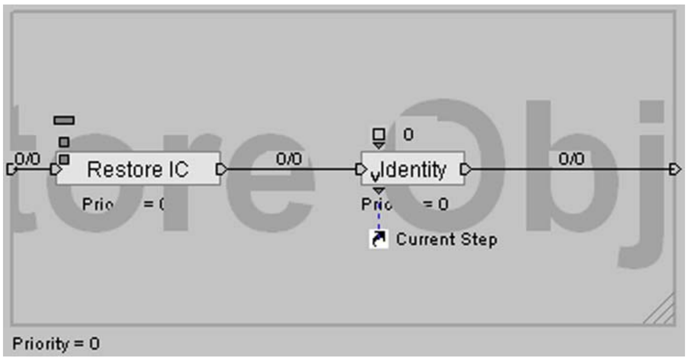


Fig.A-6. The Expanded Behaviour Graph Named 'Restore Objects'

Finally, the significant BBs in relation to these interaction functions distributed to the three control Buttons will require to be expatiated in detail regarding their individual functionality and utilities. The frequently used BB named 'Set Global Animation Step' can be regarded as the most crucial BB during this specific behaviour programming process, which often could be employed to set the advancement in percentage of the global animation. The BB named 'Wait Message' can be utilized to wait the receipt of a message. The BB named 'Identity' may be adopted to output the input parameters while it is activated. The BB of 'Op' can be applied to process any valid Parameter Operation (paramOp). Furthermore, the BB of 'Restore IC' can be exploited to restore an object's Initial Condition for the scene being played. The BB of 'Delayer' could be used to wait for a given time to elapse using internal looping. The BB of 'While' could be normally used to loop while 'Condition' is TRUE. At last, The BB of 'Threshold' could be often adopted to bind a variable between two limits (Min and Max), and exit by the corresponding output according to the comparison result. The introductory graphs with reference to these diverse BBs would be evidently illuminated in a number of following pictures so as to set out more their specifications and attributes **(Fig.A-7) (Fig.A-8) (Fig.A-9) (Fig.A-10) (Fig.A-11) (Fig.A-12) (Fig.A-13) (Fig.A-14)**.

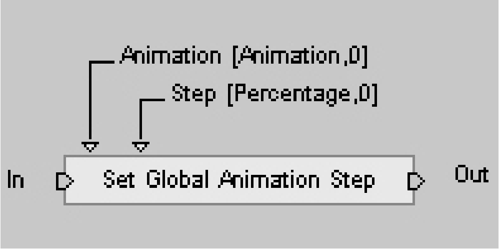


Fig.A-7. The Introductory Graph of the 'Set Global Animation Step' BB

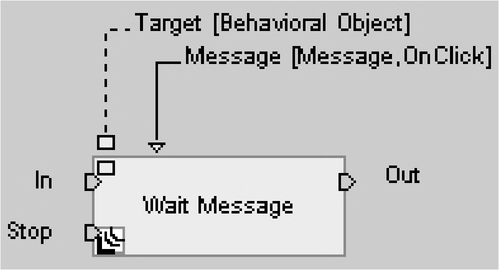


Fig.A-8. The Introductory Graph of the 'Wait Message' BB

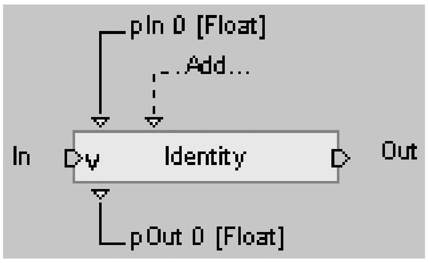


Fig.A-9. The Introductory Graph of the 'Identity' BB

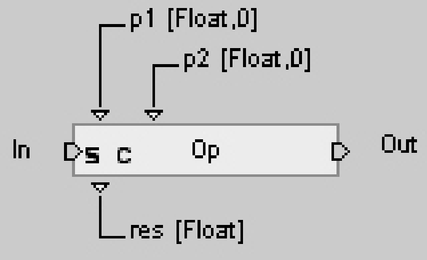


Fig.A-10. The Introductory Graph of the 'Op' BB

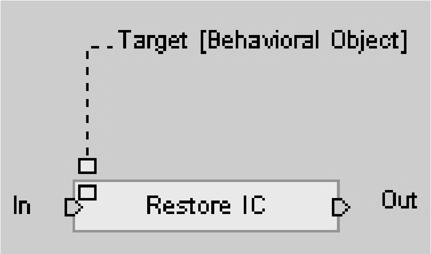


Fig.A-11. The Introductory Graph of the 'Restore IC' BB

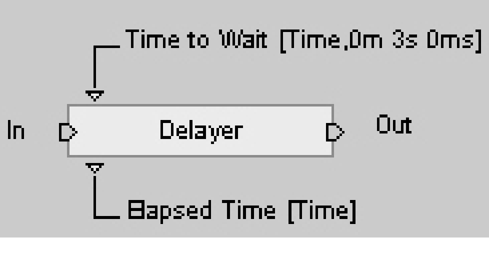


Fig.A-12. The Introductory Graph of the 'Delayer' BB

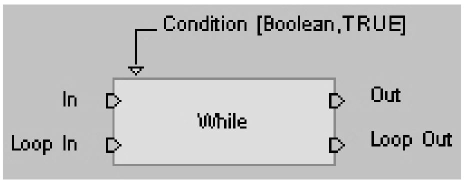


Fig.A-13. The Introductory Graph of the 'While' BB

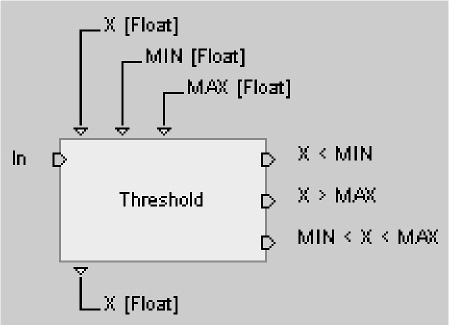


Fig.A-14. The Introductory Graph of the 'Threshold' BB

On the other hand, in purpose to achieve the simulation capability of interacting and observing the turbine blade fixturing scenario from different angles, perspective viewpoints and distance, there have been two key BBs involved into this specific behaviour programming. One key BB called 'Camera Obit' could be used and assigned to a proposed camera in order to enable it to orbit around a target 3D entity. Another key BB called 'Set As Active Camera' can be utilized to change the active camera in the scene in order to reach the goal of dynamically switching between different cameras. More details with regard to those two BBs can be gained in experimentation section 3.4.2.4. As a result, the specific behaviour programming syntax that has been adopted to fulfil this interaction function with regard to the scene camera would be visually represented in the picture as follow **(Fig.A-15)**.

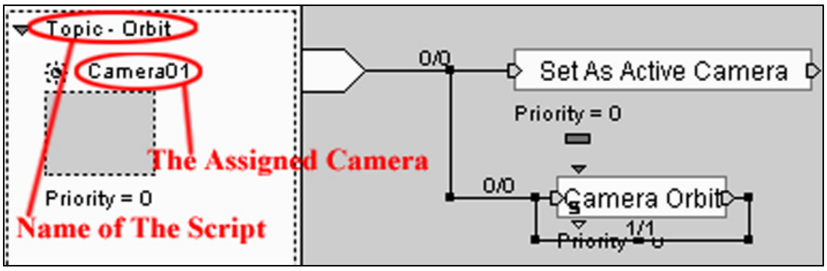


Fig.A-15. The Behaviour Programming Script Applied to the Scene Camera

In addition, in order to complete another simulation feature of autonomously rotating the 3D 'start' object **(Fig.A-16)** in the scene, there have been two primary BBs undertaken to this specific behaviour programming syntax. One important BB named 'Per Second' may be normally adopted to calculate a progression (Y) according to a given velocity (X): Y=X*Elapsed Time during one Frame. X could be any type of parameter as long as this parameter is only made up of float values or derived type. Another important BB named 'Rotate' can be generally employed to rotate the target 3D entity in the simulation scene. The introductory illustrations in relation to these two primary BBs could be explicitly clarified in the two following graphs in purpose to articulate more specification and attribute about them **(Fig.A-17) (Fig.A-18)**. As a result, the specific behaviour programming details that have been exploited to accomplish this simulation feature concerning the constant rotation of the 3D 'start' object would be clearly presented and explained in the picture shown below **(Fig.A-19)**.

Fig.A-16. The 3D 'Start' Object in the Simulation

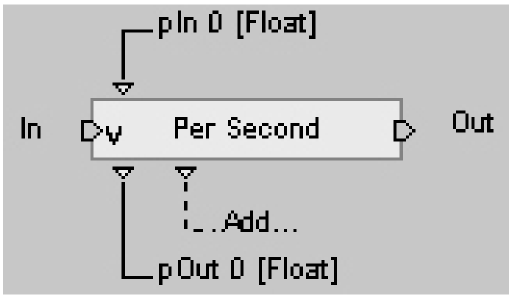


Fig.A-17. The Introductory Graph of the 'Per Second' BB

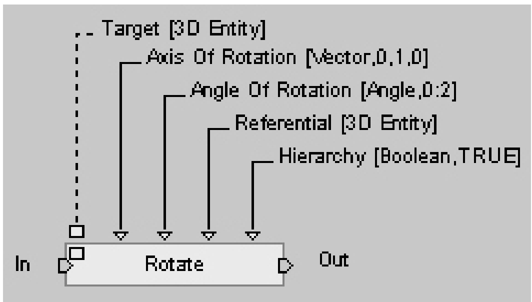


Fig.A-18. The Introductory Graph of the ‘Rotate’ BB

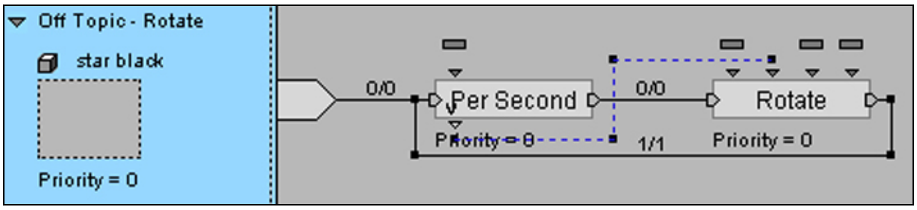
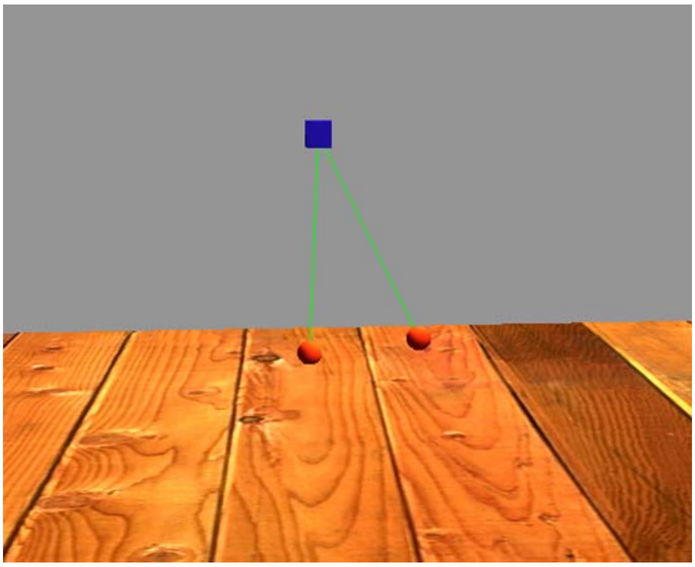


Fig.A-19. The Behaviour Programming Script Used for the 3D ‘Start’ Object

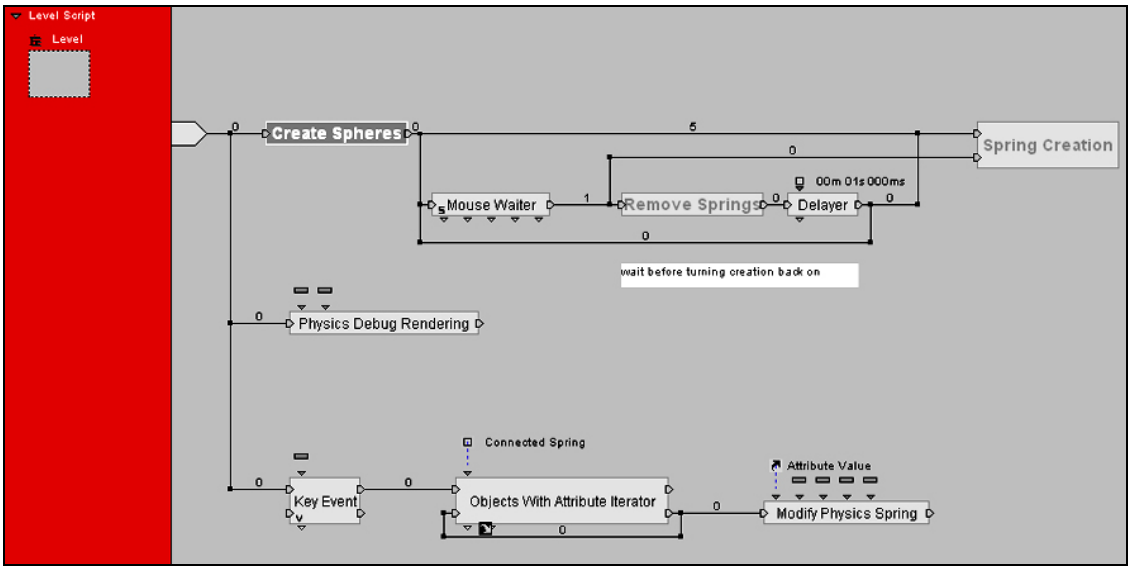
APPENDIX B

1. The Experimentation of Pendulum Simulation

The screenshot of interactive pendulum simulation with physics interactions

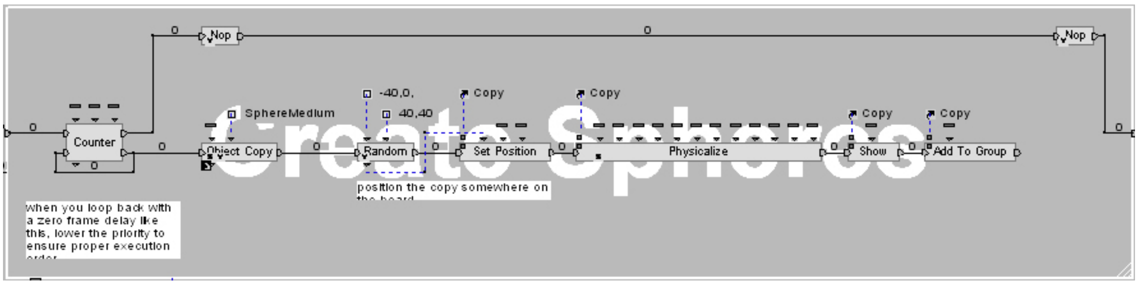


The behaviour programming script (Part 1)



wait before turning creation back on

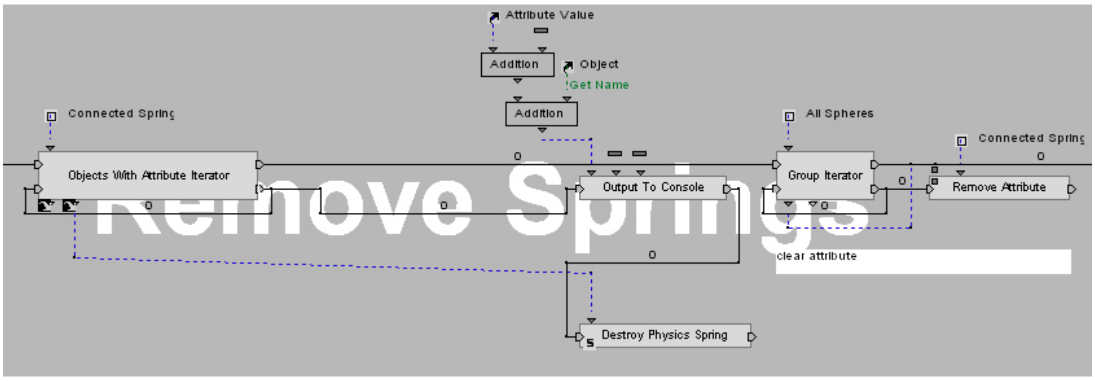
The expanded behaviour graph named ‘Create Spheres’ (Part 1)



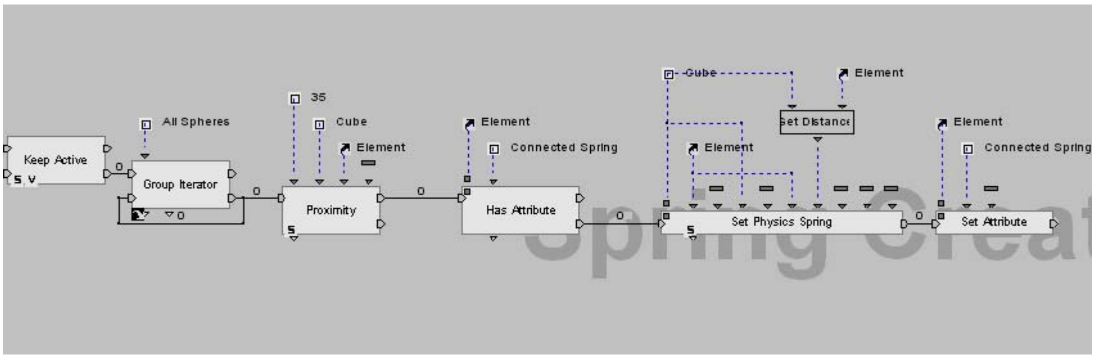
position the copy somewhere on the board

when you loop back with a zero frame delay like this, lower the priority to ensure proper execution order

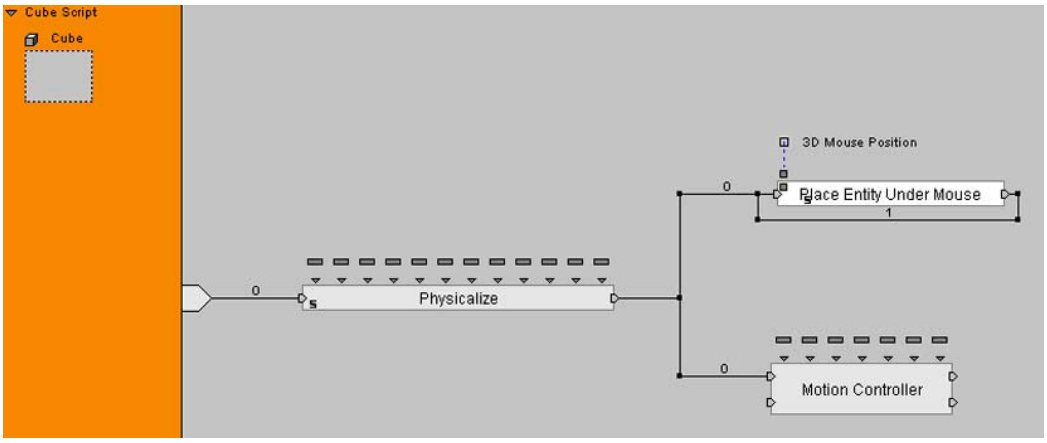
The expanded behaviour graph named ‘Remove Springs’ (Part 1)



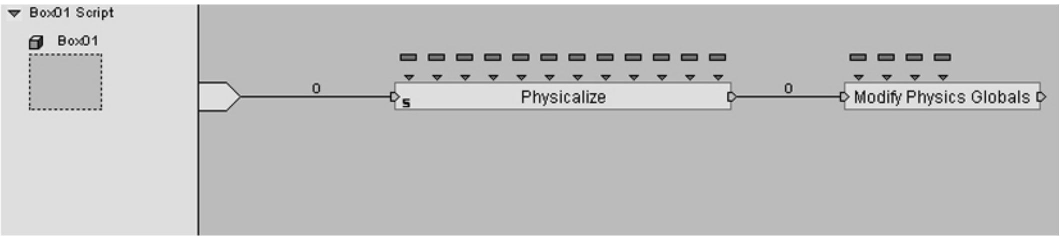
The expanded behaviour graph named ‘Spring Creation’ (Part 1)



The behaviour programming script (Part 2)

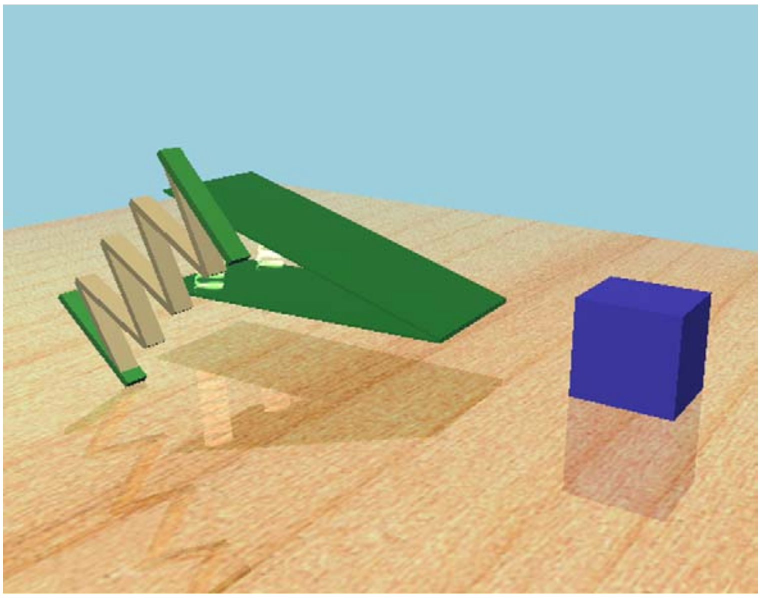


The behaviour programming script (Part 3)

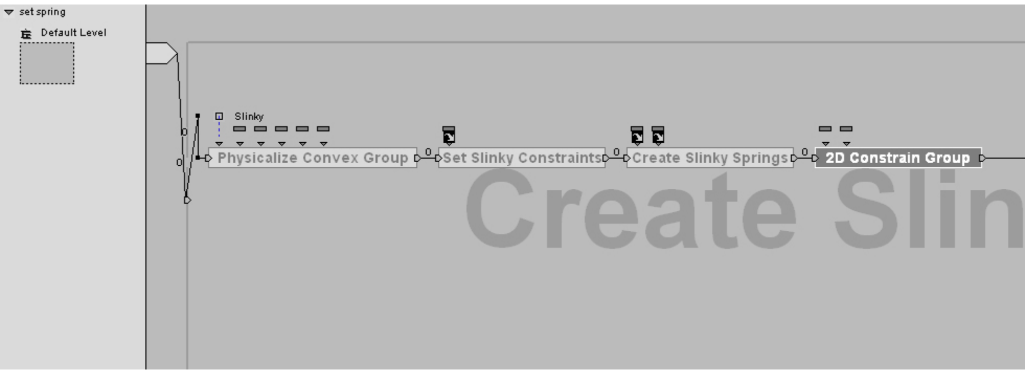


2. The Experimentation of Spring Physics Mechanism

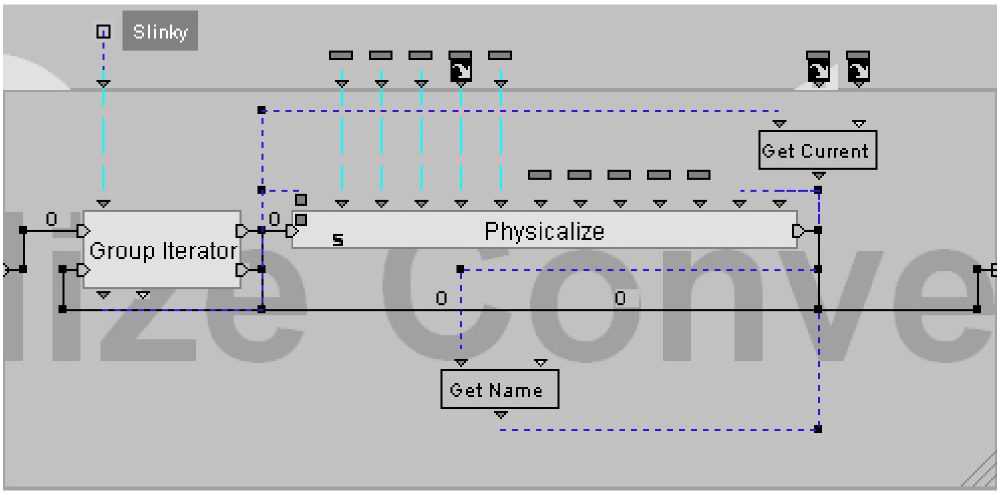
The screenshot of interactive spring physics mechanism simulation



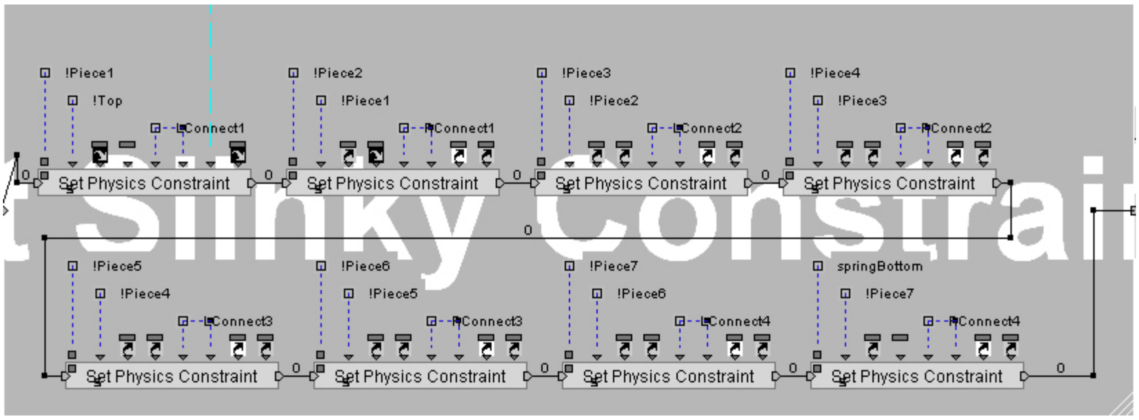
The behaviour programming script (Part 1)



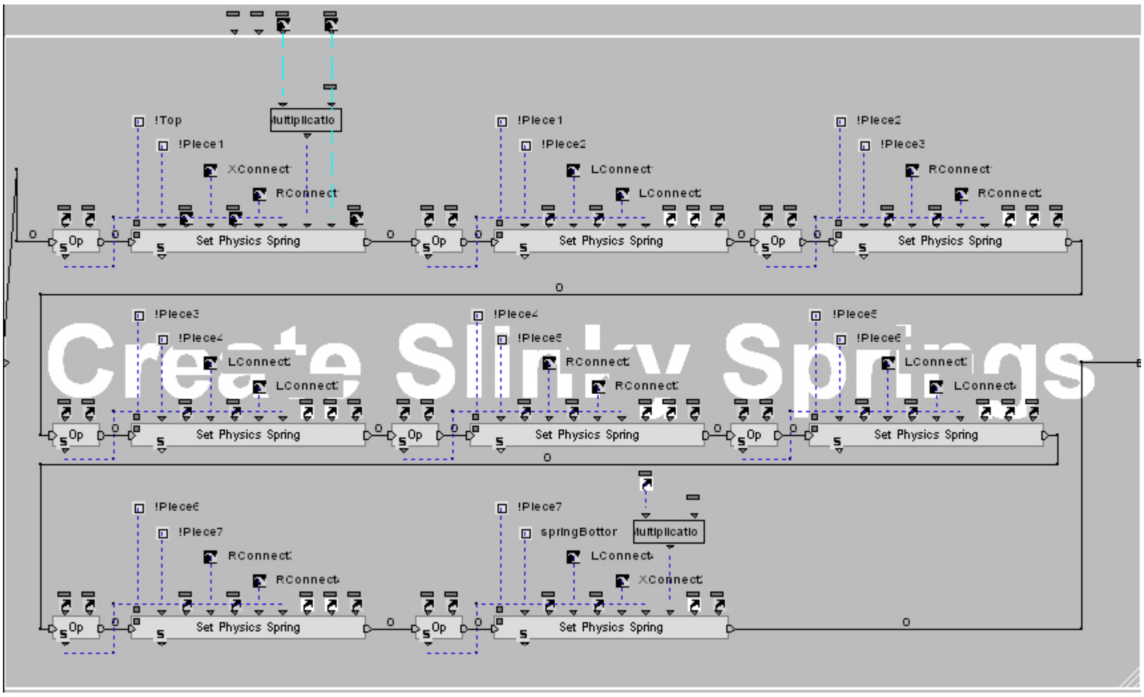
The expanded behaviour graph named ‘Physicalize Convex Group’ (Part 1)



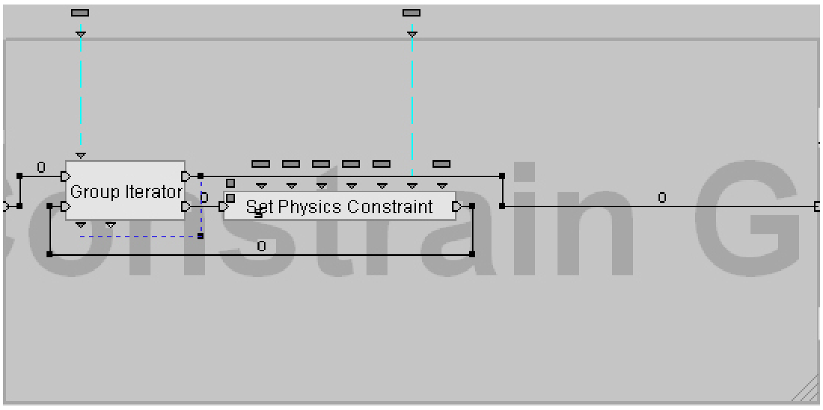
The expanded behaviour graph named ‘Set Slinky Constraints’ (Part 1)



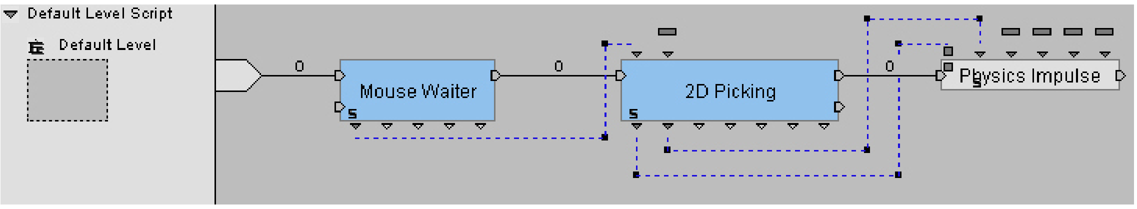
The expanded behaviour graph named ‘Create Slinky Springs’ (Part 1)



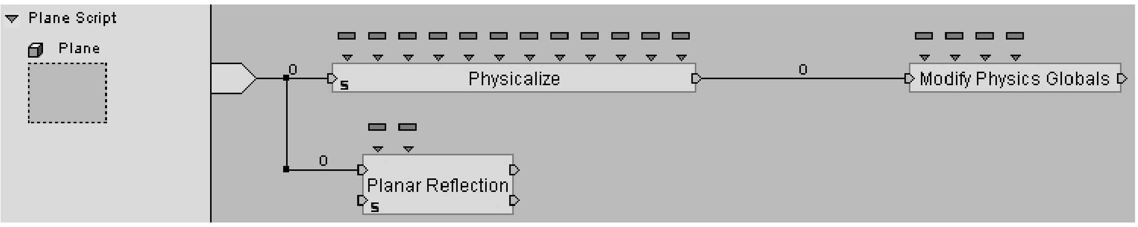
The expanded behaviour graph named ‘2D Constrain Group’ (Part 1)



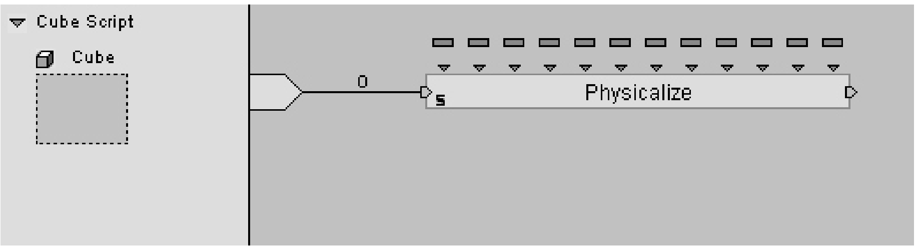
The behaviour programming script (Part 2)



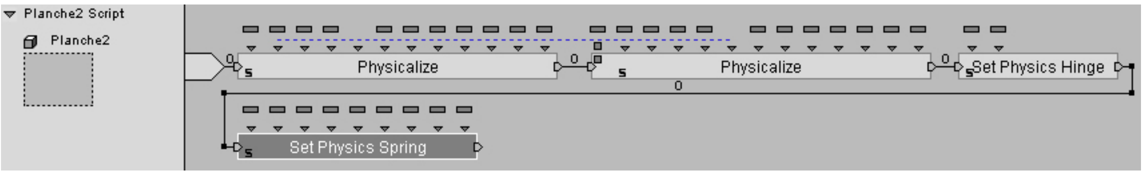
The behaviour programming script (Part 3)



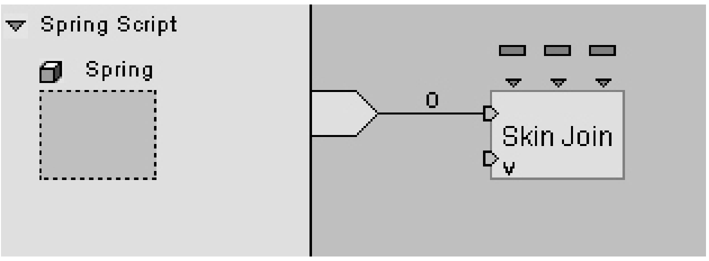
The behaviour programming script (Part 4)



The behaviour programming script (Part 5)

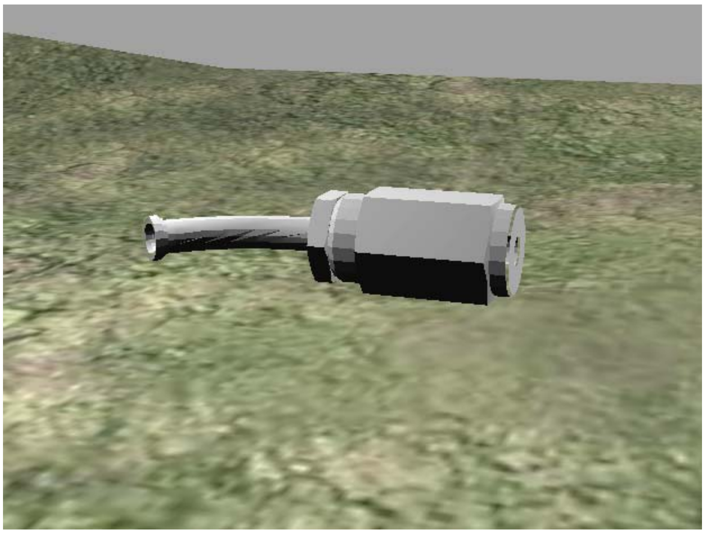


The behaviour programming script (Part 6)

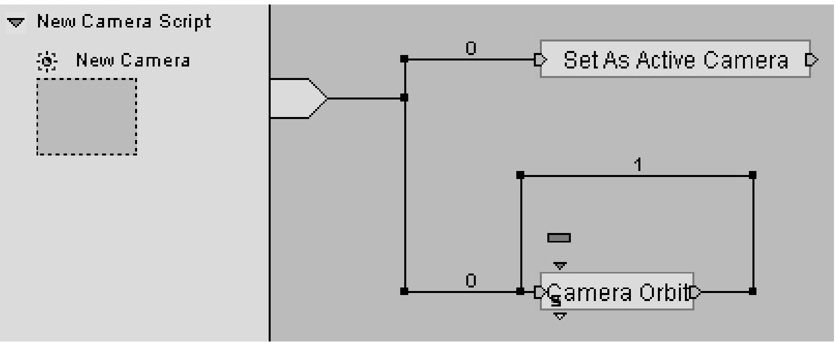


3. The Experimentation of Mechanical Component Showcase

The screenshot of interactive mechanical component showcase simulation



The behaviour programming script



APPENDIX C

1. The Specification of ‘Push Button’ BB

In accordance with the introductory illustration indicated in **Fig.C-1**, the Behaviour Input (bIn) called ‘On’ in this BB symbolizes that the BB process would be triggered as soon as an activation flow can be received by the ‘On’ bIn. In contrast, the Behaviour Input called ‘Off’ implies that the BB process would be deactivated once an activation flow can be gained by ‘Off’. In addition, the Behaviour Input called ‘Disable’ in this BB indicates that the target button is able to be disabled if necessary while an activation flow can be detected by the ‘Disable’ bIn. The Behaviour Output (bOut) called ‘Pressed’ in this BB denotes that it would be activated while the target button can be pressed by the users. The Behaviour Output called ‘Released’ in this BB means that it would be triggered while the target button can be released by the users. The Behaviour Output called ‘Active’ indicates that it would be activated once the target button can be activated. Besides, The Behaviour Output called ‘Enter Button’ signifies that it would be activated as soon as the cursor enters the button. The Behaviour Output called ‘Exit Button’ signifies that it would be activated as soon as the cursor exists from the button. The Behaviour Output called ‘In Button’ indicates that it would be activated as soon as the cursor is located within the button area. On the other hand, the Local Parameter that is linked with the Parameter Input called ‘Released Material’ indicates the specified material to use while the button is released. The Local Parameter called ‘Pressed Material’ indicates the specified material to use while the button is pressed. The Local Parameter called ‘Roll Over Material’ indicates the specified material to use while the mouse is located on the button. The Local Parameter called ‘Font’ indicates the specific font to use if the button needs to have a text on it. The Local Parameter called ‘Text’ implies the specific text to display on the target button. The Local Parameter called ‘Pressed Offset’ is able to imply the offset applied to the text while the button is pressed. The Local Parameter called ‘Disabled Material’ indicates the specific material to use if the button is disabled. Lastly, the Parameter Output called ‘Current Material’ denotes the material which is presently applied on the target button.

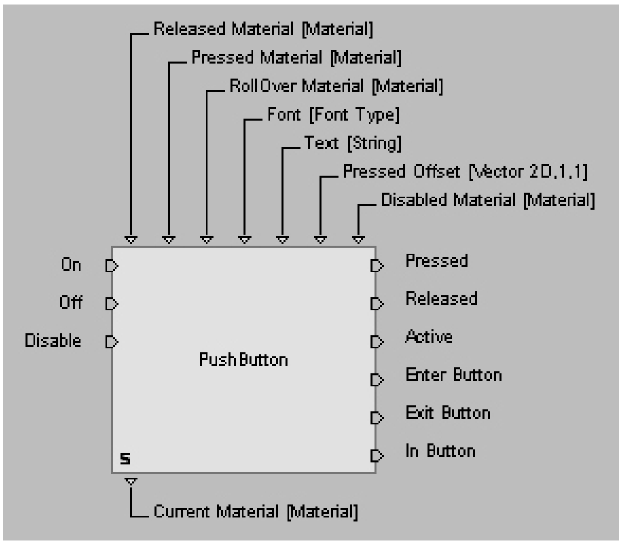

Released Material [Material]
Pressed Material [Material]
RollOver Material [Material]
Font [Font Type]
Text [String]
Pressed Offset [Vector 2D,1,1]
Disabled Material [Material]
On
Off
Disable
PushButton
Pressed
Released
Active
Enter Button
Exit Button
In Button
Current Material [Material]

Fig.C-1. The Introductory Illustration of The BB named 'Push Button'

2. The Specification of 'Send Message' BB

According to the introductory illustration indicated in **Fig.C-2**, the Behaviour Input (bIn) in this BB represents that the BB process would be triggered once an activation message can be received by 'In'. The Behaviour Output (bOut) represents that it would be activated as soon as the BB process can be accomplished. In addition, the Local Parameter that is associated with a Parameter Input named 'Message' denotes the specified message to propagate, which can be easily created and given a name by the VE programmers. The Local Parameter named 'Dest' in this BB denotes the target 3D Entity to which the message aims to be sent. In addition, through adding the pertinent Input Parameters upon this BB, the related data can be appropriately bound and associated with the message.

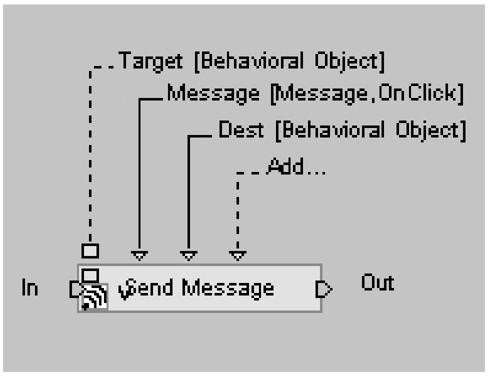


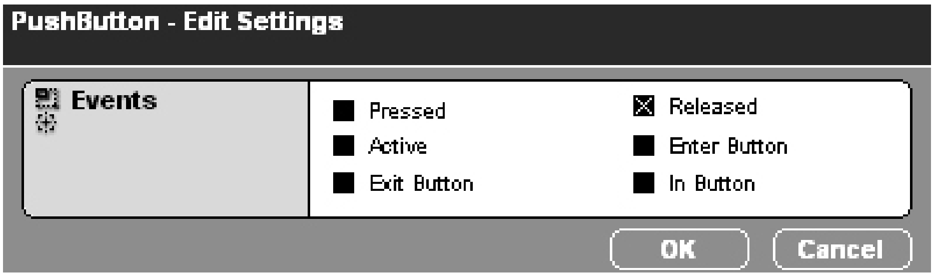
Fig.C-2. The Introductory Illustration of The BB named 'Send Message'

APPENDIX D

1. Detailed Behaviour Programming Process for Loading Buttons

Initially, these two BBs can be gained and dragged from the BBs standard library embedded in Virtools Platform according to the specific directory, then distributed into the Script named ‘Clamp Loading Script’, i.e. ‘Push Button’ BB (Interface/Controls) and ‘Send Message’ BB (Logics/Message).

Secondly, the Behaviour Outputs (bOut) of ‘Push Button’ BB can be attributed via Edit Settings dialogue. Then configure the Behaviour Outputs in the light of the graph shown in **Fig.D-1:** the bOut of ‘Released’ is needed to be checked. It symbolizes this bOut would be triggered while the target button has been released. Moreover, the Local Parameters of ‘Push Button’ BB can be modified through Edit Parameters dialogue **(Fig.D-2)**. Then configure the Local Parameter called ‘Released Material’ to be the specified material of ‘Released Button Material’; the Local Parameter called ‘Press Material’ to be the specified material of ‘Pressed Button Material’; the Local Parameter called ‘Roll Over Material’ to be the specified material of ‘RollOver Button Material’; the Local Parameter called ‘Text’ to be the string of ‘Clamp’; the Local Parameter called ‘Pressed Offset’ to be X: 1 and Y: 1 and the Local Parameter called ‘Disabled Material’ to be the specified material of ‘Disabled Button Material’. On the other hand, the Local Parameters of ‘Send Message’ BB can be specified via Edit Parameters dialogue **(Fig.D-3)**. Then configure the Local Parameter called ‘Message’ to be the message named ‘Clamp’; the Local Parameter called ‘Dest’ to be the recipient of ‘Level' where the ‘Clamp’ message would be propagated to; the Local Parameter called ‘Pin 2’ to be the specific directory *Fixture\Clamps\clamp.nmo* according to the systematic fixture database of the Pre-defined VR library and the Local Parameter called ‘Pin 3’ to be X: 8, Y: 0, Z: 0. In particular, the both Local Parameters of ‘Pin 2’ and ‘Pin3’ that were created through Add Parameter Inputs of the BB can represent the relevant input data in relation to the location and database directory of the proposed fixture element model named ‘Clamp’. These data have been bound and associated with the message.

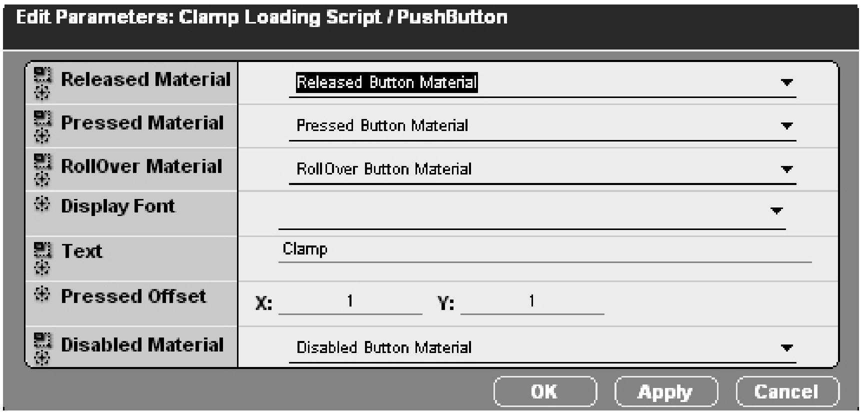


PushButton - Edit Settings

Events

■ Pressed ☒ Released
■ Active ■ Enter Button
■ Exit Button ■ In Button

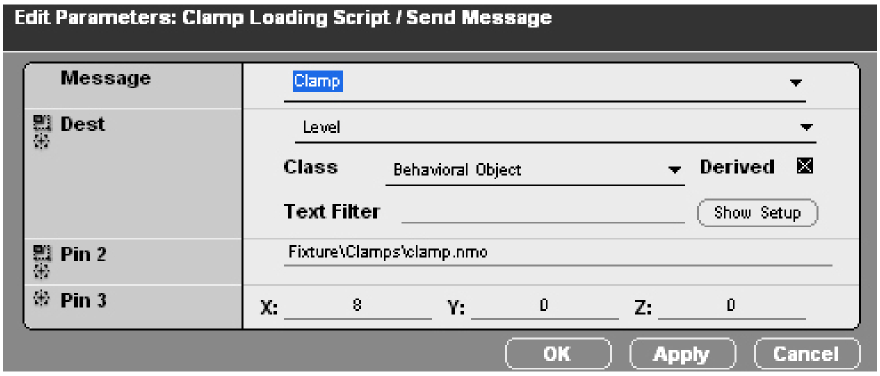
Fig.D-1. The Edit Settings of ‘Push Button’ BB



Edit Parameters: Clamp Loading Script / PushButton

Released Material	Released Button Material
Pressed Material	Pressed Button Material
RollOver Material	RollOver Button Material
Display Font	
Text	Clamp
Pressed Offset	X: 1 Y: 1
Disabled Material	Disabled Button Material

Fig.D-2. The Edit Parameters of ‘Push Button’ BB



Edit Parameters: Clamp Loading Script / Send Message

Message	Clamp
Dest	Level Class Behavioral Object Derived ☒ Text Filter
Pin 2	Fixture\Clamps\clamp.nmo
Pin 3	X: 8 Y: 0 Z: 0

Fig.D-3. The Edit Parameters for ‘Send Message’ BB

Thirdly, all that remains is to add Behaviour Links between these BBs. Accordingly, working from the left, link the top bIn of ‘Push Button’ BB to the Script named ‘Clamp Loading Script’. Next link the bOut of ‘Push Button’ BB to the bIn of ‘Send Message’ BB **(Fig.D-4)**. Thereafter, the entire process of behaviour programming for the ‘Clamp’ loading button can be eventually completed.

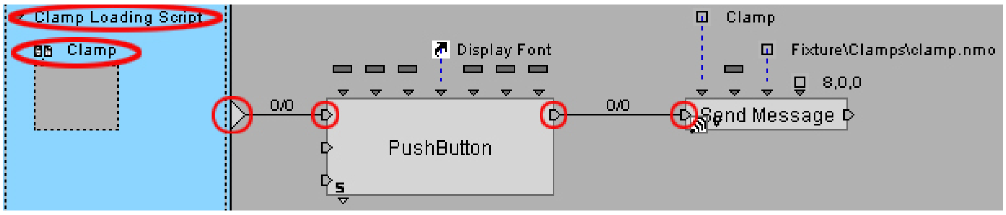


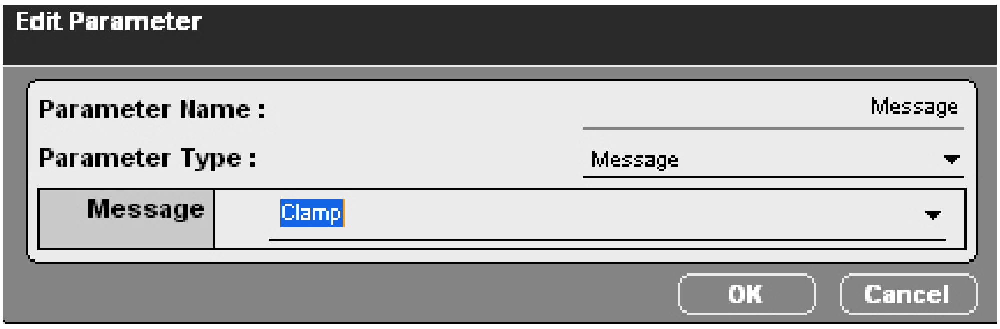
Fig.D-4. The Behaviour Links for These Two BBs

APPENDIX E

1. Detailed Behaviour Programming Process for the Input Interaction of a Fixture Element

Firstly, these five BB could be acquired and dragged from the BBs standard library according to the specific directory, then applied to the Script named ‘Object Load’, such as: ‘Wait Message’ BB (Logics/Message), ‘Get Message Data’ BB (Logics/Message), ‘Object Load’ BB (Narratives/Object Management), ‘Set Position’ BB (3D Transformations/Basic), ‘Identity’ BB (Logics/Calculator).

Secondly, the Local Parameter of ‘Wait Message’ BB could be specified via Edit Parameter dialogue. Then configure the Local Parameter called ‘Message’ to be the target message of ‘Clamp’ **(Fig.E-1)**. Moreover, the two extra Parameter Outputs of both ‘Object to Load’ and ‘Loaded Object Pos’ based on ‘Get Message Data’ BB were generated through Add Parameter Output option **(Fig.E-2)**. Using this BB, the pertinent input data bound with the ‘Clamp’ message produced by the ‘Clamp’ loading button as above-mentioned can be properly interpreted and retrieved such as: the input location and retrieval directory. Then configure the Local Parameter called ‘Message’ associated with the ‘Get Message Data’ BB to be the target message of ‘Clamp’ as well. Furthermore, the Local Parameter named ‘File’ based on the ‘Object Load’ BB can be attributed as a Parameter Shortcut of the Parameter Output named ‘Object to Load’ associated with the ‘Get Message Data’ BB **(Fig.E-3)**. The Local Parameter named ‘Position’ based on the ‘Set Position’ BB can be assigned as a Parameter Shortcut of the Parameter Output named ‘Loaded Object Pos’ associated with the ‘Get Message Data’ BB **(Fig.E-4)**. The Local Parameter of ‘Identity’ BB can be defined via Edit Parameter dialogue. Then configure the Local Parameter called ‘Warning’ to be the String of ‘The CAD models required is not in stock’ **(Fig.E-5)**.



Edit Parameter

Parameter Name : Message

Parameter Type : Message

Message: Clamp

Fig.E-1. The Edit Parameter Dialogue for ‘Wait Message’ BB

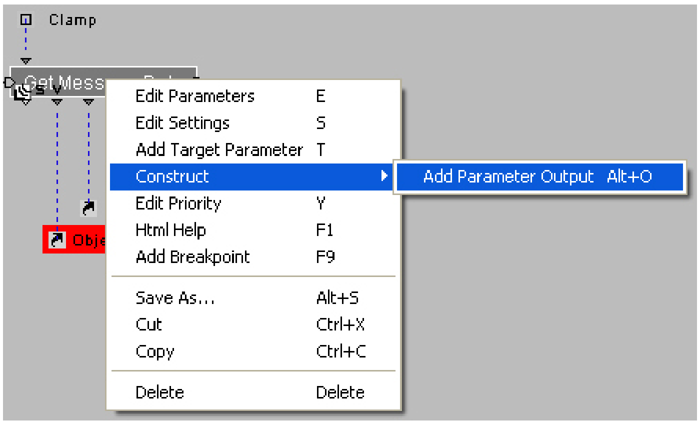


Fig.E-2. The Add Parameter Input Option of 'Get Message Data' BB

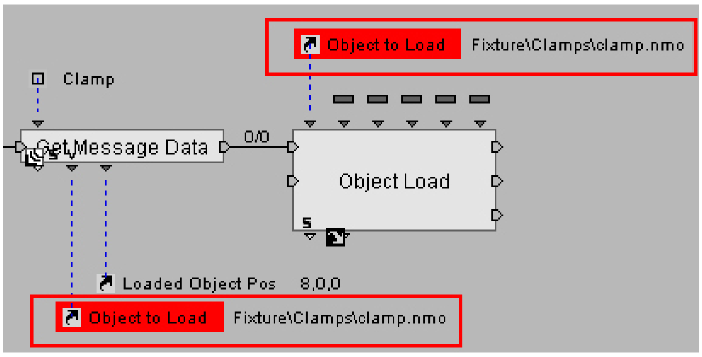

Object to Load Fixture\Clamps\clamp.nmo
Clamp
Get Message Data
0/0
Object Load
Loaded Object Pos 8,0,0
Object to Load Fixture\Clamps\clamp.nmo

Fig.E-3. The Parameter Shortcut Applied to the 'Object Load' BB

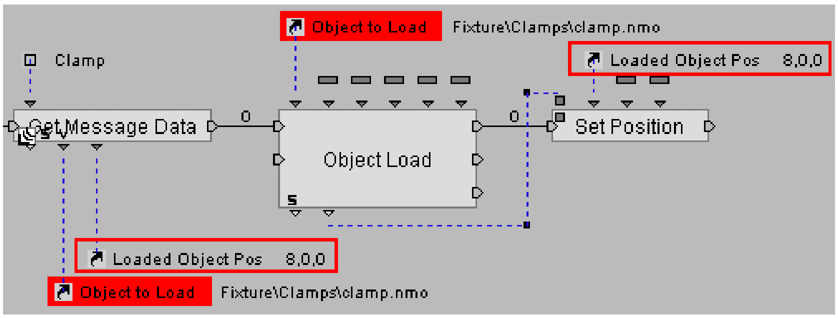
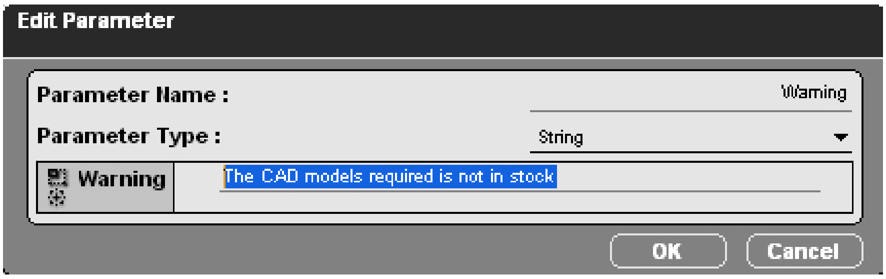


Fig.E-4. The Parameter Shortcut Applied to the 'Set Position' BB


Edit Parameter

Parameter Name : Warning

Parameter Type : String

Warning The CAD models required is not in stock

Fig.E-5. The Edit Parameter Dialogue for the 'Identity' BB

Thirdly, all that remains is to add Behaviour Links amongst these BBs and these Parameter Links from a pOut on one BB to a pIn on another BB. Consequently, working from the left, link the top bIn of ‘Wait Message’ BB to the Script named ‘Object Load’. Next link the top bIn to the bOut of ‘Wait Message’ BB. Thereafter, link respectively the bOut of ‘Wait Message’ BB to the bIn of ‘Get Message Data’ BB and then link the bOut of ‘Get Message Data’ BB to the top bIn of ‘Object Load’ BB. Then link the top bOut of ‘Object Load’ BB to the bIn of ‘Set Position’ BB and link the third top bOut of ‘Object Load’ BB to the bIn of ‘Identity’ BB. Next link the bOut of ‘Set Position’ BB to the bIn of the Behaviour Graph named ‘Parts Collision’. All Behaviour Links (bLinks) have been finalized up to now. Afterwards, link the second left pOut named ‘Master Object’ of ‘Object Load’ BB to the first left pIn named ‘Target’ of ‘Set Position’ BB **(Fig.E-6)**. At last, the entire process of behaviour programming with regard to this functional feature of input interaction can be finally completed.

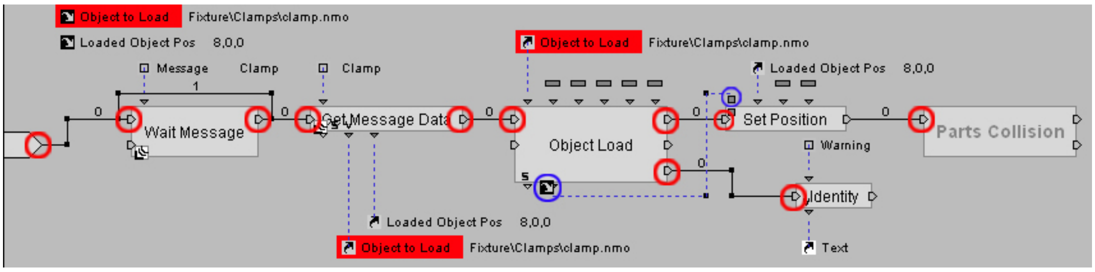


Fig.E-6. The Behaviour Links and the Parameter Link for These Five BBs