School of Electronic Engineering and Computer Science

Media and Arts Technology Programme
Project Report 2010

UNTETHERED HUMAN COMPUTER INTERACTION

Nicola Plant
Student number
090364835

August 2010

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my host supervisor Jeff Ferguson, my colleagues Steven Gaukrodger, Alexander Adderley and all at Inition for their guidance, encouragement and for providing a stimulating place to conduct research. I would like to thank my supervisor Dr Yiannis Patras and Dr Pat Healey for their academic advice and support. My thanks also go to all at Studio Seven for their co-operation and for being patient test subjects. I'd like to thank Kristine Middlemiss at Autodesk for her technical support. Finally, I would like to thank my fellow peers for their encouragement, as well as family and friends who helped along the way.

ABSTRACT

Inition are in the process of developing their LILA system (Live Interactive Lifelike Avatar). The system is based on the Organic Motion Stage (a marker-less motion capture system), and MotionBuilder Software with customised plug-ins using the ORSDK (Open Reality Software Development Kit). The LILA system offers users real-time 3D interactions with virtual worlds and interfaces by simply walking into the area in regular clothes and moving naturally. The system is being used in this research to develop a library of universally intuitive gesture-based interactions to; drive a graphical user interface, control actions in live 3D gaming, interact with avatars and virtual objects that have real-time physics.

This report describes research based on exploring what physical actions have meaningful connotations when the user is confronted with these virtual interfaces, characters, and objects, as defined as interaction techniques. The aim of the project is to evaluate the success of each interaction by assessing how well the user can control the system by using gesture-based interaction techniques, determining how fluid and easy this becomes with each instance.

This report also explores a user study which compares the effectiveness of three different types of gesture interaction techniques that can be used to select items on a display menu. The expected result of these studies is that the gestures which are representative of known actions would serve as the most intuitive when applied to a 3D gesture-based interface.

CONTENTS

1.0 INTRODUCTION

1.1 Background: Towards 3D Human Computer Interaction

As advancements in computer hardware and software technologies progress, the way users interact with computers significantly changes. There have already been considerable developments from early command based interfaces to the Graphical User Interfaces (GUI) currently widespread. The heavily dominating Windows, Icons, Menus, and Pointer (WIMP) (see the glossary in Appendix A for a definition) interface paradigm is used in most computing systems at present. However, the WIMP based interaction technique was originally produced for computer systems that are now nearly 10 years out of date. (BenHajji & Dybner, 1999)

The main problem with current user interfaces (UI) is that interaction is always mediated via devices non-intuitive to humans, such as the mouse and keyboard. 'Communication is on the terms of the computer rather than natural human terms such as speech or body language.' (Moeslund, 2000, p.1) 'We believe that many of our natural abilities are blocked by this standard HCI, forcing complexity on what could otherwise become a simple, even natural HCI task.' (Sutphen et al. 2000 p. 4)

It is now a popular notion that Human-Computer Interaction (HCI) is on the brink of evolving past the Keyboard-Mouse-Monitor interface and WIMP interaction metaphor and will progress to exploit emergent technologies[1] that transform the way users interact with computers. This coupled with that computer usage is increasingly being incorporated into many aspects of our daily lives, the need for a more efficient and intuitive means of interacting with computers is becoming more valuable. A system that more effectively matches human capabilities and perception would enable users' quicker, intuitive and more engaging interaction with the computer system.

As it stands, research into HCI is mainly concerned with information spaces displayed on desktop computers. However Weiser, (1995) quite rightly proposes

[1] Such as alternative input devices, including motion tracking, as well as alternative output technologies like large or 3D displays or head mounted displays.

that human-information interaction and human-human interaction and cooperation should be the main focus of designing such interfaces.
Weiser states that:

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. (p. 95)

Taking a step back, it cannot be denied that current user interfaces are widely used as the standard framework for a good reason. Sato (2001) argues that a reason for the wide acceptance of this framework is that it provides direct manipulation of virtual objects on a computer monitor by means of simple, cheap input device such as the mouse. This attribute provides the user a clear, generic model of the 'commands and actions possible and what their affects will be'. However, Sato also points out that this framework is not suitable for applications that require controls with higher degrees of freedom. Manipulating virtual 3D objects by using a mouse is not ideal because it operates in 2D space. This situation calls for direct manipulation with a user's hand instead offer an ideal alternative for such applications. In this way, users can control the position and orientation of a 3D object directly by simply moving their hands. (Sato et al., 2001)

Moreover, the development of computer-vision technologies and motion capture systems means the idea of conversing with computers via human gesture communication is becoming increasingly more realistic. Consequently with the three-dimensional nature of human motion, the addition of a third dimension adds more control but also more problems to the user when interacting with a computer system. For example, principles and interface models taken from existing knowledge in designing 2D UIs for desktop computers are not wholly compatible when addressing the requirements of systems where interaction takes place in 3D, with six degrees of freedom. Further research is needed to develop more appropriate interaction techniques and UI metaphors for 3D applications or interaction systems. (Bowman et al, 2004)

During the nineties, various researchers developed a number of 3D interaction techniques. Interaction was broken down into smaller tasks; these are categorized into selection, manipulation, navigation, system control, or into more complex tasks like modelling or sculpting. (Csisinko & Kaumann, 2007) The main

fundamental 3D interaction metaphors developed for 3D gesture-based interaction were the ray-casting pointing technique for 3D travel (Mine, 1995), the virtual hand for 3D selection (Pierce,1997), the Go-Go technique for 3D manipulation (Poupyrev et al, 1996), A comprehensive overview is given by Bowman (1999) . This report uses his taxonomy as a point of entry to further development and evaluation of these techniques.

1.2 Research Motivation

A basic requirement of this study is to enable users to behave naturally around a UI, free from major restrictive influences (such as markers or cables). Producing research into the development of an un-instrumented interactive system and GUI that uses human motion to control, is intuitive to use and that makes human computer interaction easy, fluid, quicker and more engaging for the user is the main drive behind this research. With a view to add, modify and tweak existing 3D interaction techniques for more successful interaction, address the implementation issues for 3D UIs and create a pipeline for applying 3D UIs to emerging technologies. As set out from a research model by Bowman et al. (2004)

In addition, developing such a system could potentially support a wide variety of applications; including avatar creation/use, data/media interaction, artistic installation, 3D games and so on also has its worth in the two following areas: the commercial sector and within the artistic domain.

1.2.1 Commercial Worth

Many instances of the sort of applications mentioned above have particular commercial popularity, especially in the area of interactive advertising or marketing. This popularity stems from the impression that the more engaging the interaction the consumer has with the content of the advertised product or brand, the better the relationship they have with it. It is of commercial worth that the technology for marketing a product is able to attract and maintain the attention of a potential customer. For example, it could be beneficial if a potential buyer was able to interact with an animated 3D model of a product. Engaging the attention of the buyer at the same time as providing information about the product only otherwise gained from a real experience of the product. In addition, as an emerging

technology, consumers are impressed with 3D interaction, so if this is built into the environment, it will not only attract interest but maintain it as well. For this reason, the application must be easy to use for inexperienced or unfamiliar users, such as the average consumer. (Burton et al. 1997)

Recent developments in low-cost digital signage have led to an increasing number of experimental and commercial platforms being invested in the commercial sector, including interactive information points and advertisements such as those seen in museums and the shop windows of technology retailers. Extra commercial value exists in using these applications to harvest consumer information, catalogue products, provide arcade style games or interactive information points for events (such as sports tournaments, festivals and conferences).

1.2.2 Artistic Worth

The use of free motion within this type of system has the potential to enhance 3D game interaction. The limitations of being physically connected to a UI system prevent the user to move naturally within the interaction space and often cause uncomfortable or erratic motions to accomplish the intended manipulation. Therefore, with the increased degree of freedom, real 3D movement brings, the easier the completion of explicit tasks in game-play.

As well as potentially providing a technology for a new interactive art medium, using such a system within other artistic disciplines could be of worth also. For example, it has been criticised that electronic sound or VJ performances lack expression, often referred to as ‘laptop performances’. As the software interfaces used by many leading performance packages are quite complex and often hidden from the audience, the performer lacks a clear and engaging dynamic to observe. Using gestural manipulation could solve this ‘laptop performer’ issue by providing clues of intention through body language.

1.3 Research Questions

The basis of this report focuses around these research questions; how do people perceive and learn to use different gesture-based interactive 3D interfaces? What factors help promote easy interaction between system and user? And lastly, but

most importantly; what interface features and interaction techniques allow users to work intuitively with the system without prior experience with it?

The design of 3D interfaces is still predominantly an art-form, usually designed relying on designer intuition and common sense. The main reason for this is that there are no existing standard 3D interface paradigms. There are some 3D interaction techniques listed in the literature (see chapter 2), yet it is not totally clear how they all relate and compare to each other, or how the design of interaction sequences to do complex tasks should be approached. Together with that there are currently next to no tools to support the design of 3D user interfaces, (if designers need to use certain interaction techniques they must either programme them from scratch or invent new techniques) designing effective interactive 3D user interfaces is a very time consuming task and as a result the produced interfaces are seldom formally user tested. (Bowman, 2004)

1.3.1 Design Principles

The following list of design principles are put forward as a guide to 'effective' interaction:

- The interactive interface should reveal meaning and functionality naturally.
- Prior training should not be required to use the system.
- The system should facilitate robust motion reaction, enhancing the performance of the user by deliberately using coarse grained motion that rely on large positional changes (Vogel & Balakrishnan, 2004)
- A natural, intuitive control scheme, with simple UI that mimics the way humans interact with the real non digital world.
- The system must meet the requirements in terms of real-time, accuracy, and robustness.
- Provide liberal visual feedback to guide the interaction.

1.4 Placement Host: Inition - Everything in 3D

Inition is a commercial centre of 3D expertise and technology. Its services are streamed into three main areas: 3D Productions: 3D film and interactive content production and technology**,** 3D printing and scanning, Resellers of a range of 3D

and VR products including HMDS, Motion capture systems and gloves, haptics, software and 3D graphical cards.

As mentioned previously, Inition are in the process of developing their LILA system (Live Interactive Lifelike Avatar). The system is based on the Organic Motion Stage (a marker-less motion capture system), and MotionBuilder Software with customised plug-ins using the ORSDK (Open Reality Software Development Kit). The LILA system offers users real-time 3D interactions with virtual worlds and interfaces by simply walking into the area in regular clothes and moving naturally.

1.5 Report Structure

Firstly, this report provides a comprehensive but relevant literature review in section two. The literature review covers the universal principles in interface interaction such as the use of interface metaphors, interface heuristics and the cognitive aspects of interface interaction. This is followed by a discussion of literature that highlights the benefits as well as the issues of interaction with a 3D user interfaces in a generic way, moving on to discuss existing interaction techniques and their implementation issues. Existing systems incorporating relevant interaction techniques will be briefly evaluated and summary of the literatures main points will be presented along with a short mention of the main research models that apply.

Next, in section three, a user study will be presented as a relevant and useful method of investigation. The techniques used to conduct the user study will be scrutinised and their merits when applied to the above mentioned research questions will be presented. In section four, this report examines and evaluates the results of the user study before, in section five, discussing future work and conclusions.

2.0 LITERATURE REVIEW

This section will introduce relevant literature covering the following areas; the universal principles in interface interaction; the benefits and issues of interaction with a 3D user interfaces; existing 3D interaction techniques; existing systems incorporating relevant interaction techniques followed by a summary of the literatures main points will be presented as well as research models that apply.

2.1 Generic Principles in Interface Interaction Design

2.1.1 Interface Metaphors

Preece et al. (2002) talk about interface metaphors as a good example of an interaction model. An interface metaphor is an interaction model that is developed with deliberate similarities to some aspects of a recognisable real-world entity (or entities) based on a distinct activity and/or object. The tool is purposely designed with the intention of encouraging the user to draw a comparison between the interaction technique and the actual real-world object or common everyday action, mapping the familiar to the unfamiliar, enabling users to quickly understand and learn about the new domain. Thus allowing them to call upon their existing knowledge and experience then apply it to the interface. This model is thought to be successful because if the interface is presented in terms already familiar to the user, it is already implicitly understood and instinctive to use.

However, Preece et al. also mention that when using this interaction model there is a danger that the design of the interface metaphor is made to *completely* mimic the real-world entity it is an analogy of. So, if the real-world entity conflicts with the design principles or was originally a bad idea to start with this could be detrimental when applied to the interface. Also, overly literal translations of real-world entities could result in the user not being able to understand the system functionality beyond the metaphor.

This principle is useful when presenting the user with a new system, such as 3D user interfaces. Borrowing natural physical gestures to perform interaction tasks is a way of incorporating this model into 3D UIs. For example, in the Smart Scenes

by Multigen Inc. (Mapes & Moshell, 1995) the user moves in the environment by pulling themselves around by grappling an invisible rope.

Another method of applying this model to 3D UIs is the inclusion of cultural influences. Movies (especially science-fiction) is a source of inspiration for much of 3D interaction design, again the same basic concept applies, as the user is already familiar with the interface and its functionality it is quicker and easier to learn to use.

2.1.2 Interaction Design Heuristics

Preece et al, (2002) defines a set of cohesive guidelines for designing interfaces with human factors in mind:

- A constant knowledge of system status should be communicated by the interface to the user through providing feedback of operation completion and clearly representing system states.
- Avoid using terms that are system orientated by translating technical commands into real-world words and phrases that are understandable to the user.
- Use consistent interface syntax and semantics; this avoids confusing the user by using different actions to mean the same things.
- Conduct proper error prevention and handling by providing ways for users to escape from unexpected places within the interface.
- Keep the design aesthetically appealing but minimalistic and simple. This can help to reduce short-term memory load.
- It could be useful to provide accelerators invisible to new users but allow familiar users to quickly navigate the interface.

2.1.3 Cognitive Aspects of User Interface Design

BenHajji & Dybner (1999) state that ‘an understanding of how humans perceive and interact with their environment can contribute to the design of more intuitive user interfaces.’ (p. 4) They elaborate by saying that when navigating a real-world environment a process of using environmental cues and artificial aids (such as maps) are used by people to control their movements, this enables them to reach their desired destination without losing their place. When presented with a 3D

virtual space a major problem for users is how to maintain knowledge of their location and orientation while moving through space. It is suggested that points of reference within the virtual space is essential to overcome this issue.

Particularly relevant is the concept BenHajji & Dybner present when discussing how interaction deals with the way humans exchange information with the environment; A psychology of causality is a consideration as we interact with the environment, meaning that the perceived result of an action appears to be caused by a particular action. The failure of the action to have the expected result may very well be related only through a coincidence, yet this kind of false causality may occur in user interfaces due to poor design. This reiterates the need for consistent interaction procedures throughout the interface, as previously mentioned by Preece et al. (2002).

Benhajji & Dybner (1999) refer to the gestalt law of perception as an important consideration when design user interfaces:

> The amount of data that flows through the human senses is so large that we are forced to filter much of it, in order to focus on elements that we consider being important. The problem is that filtering removes global context and may leave us disoriented. One solution to retain global context is to organise the data that we do retain in ways that promote understanding at the expense of detail. (p.8)

This can be applied to 3D GUIs by firstly making sure that objects related to each other are placed together. The attention of the user could easily be manipulated by the placement of virtual objects within the 3D space by correct placement of the object.

Smith (1993) also states that an understanding of depth perception is an important consideration when designing 3D GUIs. Two main considerations that should be noted are; firstly the proximity is judged by relative size, secondly the closer objects are the sharper the focus. This could aid in visualising virtual objects so the users can comprehend their shape as well as their relative positions in virtual 3D space.

2.2 3D User Interfaces

The addition of a third-dimension to UIs has the benefit of more accurately emulating how we physically interact with the real-world [Conner 1992]. This allows natural affordances when interacting with the interface; these can be used when controlling virtual objects or menus (usually defined as isomorphism; more on this later). Yet, the addition of a third-dimension makes it more difficult for the user to interact with the interface in some aspects. This is since the user must correctly align their hand position with all three dimensions of the desired coordinates if this technique is used.

Preece et al, suggests that to aid the ease of interaction, the use of geometrical constraints would be useful. Firstly, an indication of when a collision between the users input and the virtual object has occurred would be beneficial. Virtual objects adopting the behaviour of real-world objects would make their behaviour easier to predict and control, i.e. colliding with each other, possessing gravity etc. Also suggested is adding a 'snapping' constraint to the selection technique, this could make things easy to select and manipulate. For example, when the input nears a virtual object, selection snaps to it allowing for precise alignment. The use of inertia and speed constraints is also suggested when manipulating the position of virtual objects. As well as functions that reduce the degree of freedom.

The use of realism (or isomorphism) in 3D interaction is much like the interface metaphor discussed above, just in a more literal manner. This approach involves the interface being accurately mapped (i.e. one-to-one mapping) to the interaction space for direct manipulation of virtual objects that respond with real-world reactions. The advantages of using this model are similar to those of the interface metaphor, where the user is already familiar with the system, so will find it intuitive and natural to use. It is easy to implement, as the causality is already known and understood. Also, it is a must for simulation applications. The disadvantages to using this model are that limitations of technology do not allow for exact isomorphism so will never be truly realistic. Also, limitations of the physical world apply and these do not permit ultimate freedom to the user. As well as this, isomorphic mappings are often impractical because of input device constraints such as limited tracking range or the limits of human operators such as anatomical constraints. (Pouprev et al, 1996)

Beaudouin-Lafon (2004) states that users have an increased sense of engagement, when using direct manipulation, because they manipulate intermediate objects directly. This matches our experiences in the physical world:

> We rarely finger-paint, but often use pens and pencils to write. We cook with pots and pans, hang pictures with hammers and power drills, open doors with handles and turn off lights with switches. Our interaction with the physical world is governed by our use of tools. (Beaudouin-Lafon, 2004, p. 3)

Direct manipulation of physical objects of interest occurs when we bring them into our current context of operation, usually with two hands. Our hands are metaphorically representative of a tool, and this serves as an example of how a 3D interface mimics 'tool' use successfully. Two-handed operation allow for the transference of everyday skills interaction. Where we can perform two tasks in parallel (symmetric manipulation), or use both hands to complete a single task (asymmetric manipulation).

On the opposite scale to realism, the use of a non-isomorphic approach suggests that interaction does not behave strictly in the exact way the real-world does. Mappings are amplified or distorted to ease to completion of certain tasks. Some advantages of non-isomorphism are that it can be enjoyable, providing users with "magic" virtual tools. It also allows for a larger scope that the real-world where use of scaled linear and non-linear mapping functions which, in effect, give users more power to manipulate virtual objects in 3D worlds. The disadvantages are that the interaction can sometimes be misleading, or 'kinaesthetically' inconsistent. (Poupyrev et al. 1996)

When used in 3D user interfaces, the non-isomorphic approach has been particularly useful for dealing with translational manipulation in multiple DOF input as it is a good work around for overcoming anatomical or motion tracking limitations. Predominantly for 3D rotations, interaction techniques mostly use simple real-world mappings between virtual objects and the 3D input. The advantages gained from using a non-isomorphic approach to 3D spatial rotations is that it provides a work around for handling basic anatomical limitations. For example, with an isomorphic approach, rotating an object a full circle is extremely difficult in one motion since our arms have limited rotational movement both

around each other and about the elbow depending upon what technique is being used. A non-isomorphic mapping could solve this problem by amplifying the mapping of the users' spatially limited input in the interaction space to rotate a full 360 degrees.

On the other hand, in a real-world situation or when using an isomorphic approach, one would rotate an object full circle by grasping the object and rotating it a certain amount, release and then grasp again to rotate further to achieve a full 360 degrees by accumulation. The question we must consider is how these interaction approaches improve or hinder user performance.

In addition to this, rotations in 3D space do not obey the laws of Euclidean geometry that most 3D applications employ. For example, if an object is rotated in a certain direction it will always return to its original orientation. Pouprev et al, (1996) states that the reason for this is that the space of rotations is not a vector space, but a closed and curved surface that can be represented as a 4D sphere. Here, using quaternion's is the most accurate mathematical method to describe rotations within the context of this 4D sphere.

A quaternion is a four-dimensional vector (x, y, z, w), sometimes represented as a pair ($\boldsymbol{v}$, w) where $\boldsymbol{v}$ is a 3D vector defining the axis of orientation and w is a real number that defines the amount the objects is rotation around this axis. By using a quaternion mathematical framework the problem of gimbals locking, caused by non-accumulated vector rotations, is resolved. This problem occurs because absolute mappings using Euler rotations do not preserve directional compliance as the absolute angular displacement is only ever relative to the initial, zero orientation rather than its own shifting orientation. (Poupyrev, 1996)

Poupyrev et al. (1996) asks how effective non-isomorphic rotation techniques are in terms of speed and precision? They describe one user evaluation comparing a linear non-isomorphic rotation technique with relative mapping versus a conventional one-to-one mapping scheme. It was expected that a relative amplification of rotations from an input device with multiple degrees of freedom would permit users to perform a rotation task faster than one-to-one mapping assuming large range rotations are desired. But the downside would be that amplified non-isomorphic rotation techniques requiring smaller ranging of rotation would decrease precision. The results of the user study indicated that the subject

performed the rotation task 13% faster with the amplified non-isomorphic approach and there was in fact no accuracy degradation. (Poupyrev et al, 1996)

2.3 Gesture-Based Interaction

2.3.1 Benefits of Gesture-Based Interaction

By implementing a system that uses gesture-based interaction, one can open up a new set of techniques available to the user. O'Hagan et al (2002) points out that instead of being limited to the 'point and click' interaction metaphor generally used in WIMP interfaces, new metaphors can be explored that could potentially remove the necessity for menus or toolbars on the interface, simplifying the display presented to the user. He suggests that, rather than choosing tools from a menu onscreen, different gestures could be used to indicate the desired action. As long as careful consideration is given to choosing a gestures that are intuitive ensures the cognitive load on the user is not increased by the need to remember specific gestures, but reduced by simplifying interaction with the interface, interaction would be faster and more efficient.

Hand (1997) suggests that the design of gesture-based interaction techniques should be informed by how 3D tasks are conceptualised by the user. He suggests that in the real-world many manipulation and navigation tasks are performed without conscious attention and it is this level of natural interaction such technologies seek to attain. The benefit of this is that gesture-based interfaces would almost become invisible when virtual objects can be manipulated as if they existed in the real-world. This would allow the user to focus on the task, engaging with the virtual world and 'feeling as if they are interacting with the objects directly, with no intermediary'. (Hand, 1997, p. 3)

To add, a study on the usability of gesture-based interaction in user interfaces was made by Kwon & Gross (2007). Usability tests of their system found users often formed the opinion that a gesture-based interface made the application more appealing and encouraged the users to concentrate more on the tasks to be achieved.

2.3.2 Issues with Gesture-Based Interaction

As is the nature of gesture-based interaction, unrestrained movement encourages a liberal and intuitive interaction. (Albuquerquw et al., 2004, p. 3) However, Albuquerquw et al also suggest that the fact the user can move so freely implies that the interaction system must be very robust. He says that the gestures must therefore be chosen to be meaningful to the user in order to be both easily issued by the user, and robustly interpreted by the system. The variety of communicative gestures is a challenge by itself. It has also been said by Albuquerquw et al. that even though some gestures with a defined shape can be linked to a precise meaning, most of the communicative gestures are creative, i.e. they are created on the spot. Thus, gesture shape can vary a lot depending on their meaning and discourse context. Implementing such communicative gestures into a gesture-based interaction technique could be detrimental to the robustness of a system.

Bowman et al. (2004) points out that an issue with gesture-based interaction is that the user would need to remember what actions execute what commands. Due to the limited capacity of the average users working memory, this may not be so straight-forward It may lead to unfamiliar users to have problems with when navigating the interface, especially when the application is complex and requires a larger amount of functions. They suggest that, in order to make gesture-based interaction easier to use for new users, strong feedback, like visual cues after initiation of a command, would be needed. (Bowman et al., 2004)

Mine (1995) points out that reproducing specific hand gestures to execute commands in an interface under practical situations is very difficult due to the high degree of freedom of the human hand. This could make unrestrained interaction in a virtual world lack any sort of precision from the user. Mine describes this well with:

> Virtual environments suffer from a lack of haptic feedback (which helps us to control our interaction in the real world) and current alphanumeric input techniques for the virtual world (which we use for precise interaction in the computer world) are ineffective. We are unfamiliar with this new medium we work in; we do not fully understand how to immerse a user within an application. Before we can create virtual world solutions to real world problems we must learn how to interact with information and controls

> distributed about a user instead of concentrated in a window in front of him. We must identify natural forms of interaction and extend them in ways not possible in the real world. (Mine, 1995, p. 1)

To add Albuquerquw (2006) states that to control virtual objects the user 'touches' them with the interaction interfaces. 'Without the physical collision, however, it is not easy to perceive the distance between the interface and the target.' (p. 4) He presents a system that defines 'gizmos' as a solution to this problem. The system uses a number of 'gizmos', these are in the form of proxy objects which are set into the virtual environment as visual feedback clues. The 'gizmos' are not part of the 3D model, but serve as a visual representative for a particular operation, to indicate a specific mode, or to give information about the status of the interaction. The idea behind this system is that that the 'gizmos' improve the robustness of the interaction tasks by using them to execute certain tasks and therefore making them more controllable as they are not directly issued by gestures, but indirectly. The gizmo is still manipulated only by gestures, but the difference is that once called; the 'gizmo' can issue the command with the correct precision. (Albuquerquw, 2006)

2.3.3 Existing Gesture-Based Interaction Techniques

The main 3D interaction techniques to be discussed below address the following interaction tasks as classified by Bowman (1999) also listed are some interaction techniques that achieve the specific task**:**

- Selection: picking objects from a set – Virtual Hand, Ray-Casting, Sticky Finger (occlusion), Go-go (arm-extension)
- Manipulation: modifying object properties such as translational or object orientation – HOMER, Scaled-world grab, World In Miniature.

Selection

Virtual Hand Technique: This technique works by seeing that objects are selected by contact of pointer and object or its close neighbourhood (such as its bounding volume). Usually, the pointer is represented by a virtual hand and is controlled by the users hand. Poupyrev et al. claims that in this case it is not necessary to confirm

selection because visible collision between the virtual hand and object can be regarded as unambiguous confirmation of selection.

Yet, quite separately, Poupyrev et al. suggests that ‘when implementing object selection, it is essential to incorporate adequate feedback’. It is stated, that for intuitive and easy interaction the user must be made aware of when they have chosen an object for selection, and then they must know that they have successfully executed a selection command and additionally the user must be able to determine the current selection state of all objects.

Go-Go Technique: When interacting with an object within a virtual environment in natural manner, interaction is easy only with objects that are located within the users arms reach. However, a problem arises if the user wants to interact with remote object they cannot reach. This occurs when the distance between the virtual object and user is larger than the physical length of the users arm. Under such circumstances the user has to physically move closer to the virtual object to interact with it, so if the virtual object is outside of the interaction space the user has no way of accessing it. A technique called the Go-Go technique, originally invented by Poupyrev et al, uses non-isomorphic selection to allow the users’ virtual arm the ability to dynamically grow to the desired length.

The Go-Go technique extends the virtual hands reaching distance by using a non-linear mapping function applied to the users’ real hand extension. Poupyrev et al explains that the space around the user is split into two concentric regions. When the users’ real hand is within the closest region around the use, the distance to the hand is smaller than a specified threshold within the interaction space, the mapping is one-to-one and the movements of the virtual hand correspond to the real hand movements. If the user extends their hand further than this threshold, the mapping becomes non-linear and the virtual arm extends at an exponential rate, thus allowing the user to access and manipulate remote objects.

Thus selection of remote objects can be now be made by the virtual hand that now has the ability extend beyond its physical limitations. Using this technique along with visual feedback provides a natural mapping (though nonlinear) of physical movement the arm to the resulting movement of the virtual hand. (Flasar, 2000) states that this technique makes the manipulation of remote objects ‘simpler and faster’, as interaction with them is achieved by ‘natural hand and arm motions’.

Ray-Casting: In this technique the user indicates a target by casting a ray into the virtual environment from their input device. The metaphor is usually used for object selection. Instead of attempting to directly specify the desired position, the spatial input device is used to cast a ray into the virtual environment, Poupyrev et al. states that by allowing the user to hold the input device in a comfortable position and rotate it to change the ray direction.

In the literature, it is worth noting the observations Poupyrev et al. (1996) have made whilst evaluating two of above mentioned interaction techniques in controlled experiments. One of the relevant observations made from the experiments is that in a selection task the ray-casting technique 'generally demonstrated better performance' from the participant than Go-Go. They suggest that this was since the ray-casting technique required less physical movement from the user. However, an exception reported by Poupyrev et al was that when a very high degree of precision when selecting an object was required, in the case of selecting small objects or those objects were located far away, ray-casting was not as effective as Go-Go. This is apparently an intuitive observation; as in the real-world pointing at small objects is especially difficult when they are further away. Being usable to specify the desired amount of precision is something that can frustrate the user. The development of overcoming this issue whilst keeping our newly acquired degree of freedom could potentially be very useful.

It was also observed that the ray-casting techniques relied on the user recognising that the ray had collided with a virtual object. In Poupyrev et als' experiment, the ray-casting was said to be implemented as a short ray the user applied to point at objects. In Bowman's (1996) experiments, the ray-casting was implemented with an infinite ray. Here, the user could clearly see the intersection of the ray with the object. The results showed that this is a more effective implementation of the technique. The results of this experiment reiterate the validity of the heuristic described in section 2.1.3 that 'a constant knowledge of system status should be communicated by the interface'.

(Pierce et al., 1997, Poupyrev, et al. 1996, Bowman et al. 1999) Also evaluates on their experiment results; that ray-casting was found to be not very efficient for positioning with a change in distance from the user, yet the technique can be very

efficient when the starting and target positions of the object are located at a constant distance.

Manipulation

There are various interaction techniques already developed and evaluated for the manipulation of virtual objects. Including the HOMER technique, the scaling factor and world-scale-grab, these are discussed below.

Scaling Factor: Selected objects usually start by transforming in one-to-one correspondence with the user's hand (isomorphic interaction), however to allow a greater range of freedom in the objects motion, an amplification factor can be applied to the transformation of the object.

There are several different methods for the implementation of the scaling factor. The scaling factor can be specified by a single hand, the movement of the hand determining the scaling factor of the selected object. Mine (1995) explains that; 'movement of the hand up could signify a scale-up action, movement of the hand down could signify a scale-down action, and the range of hand motion could determine the magnitude of the scaling action' The addition of a second hand to the interaction gesture allows a reference for the scaling. So one hand could specify position on an object and the other could extend away in the desired direction. Mine states that this method is particularly well suited for non-uniform scaling.

Both the HOMER and world-scale grab techniques are based on an observation that selection and manipulation of virtual objects are sequential tasks. In other words, an object has to be selected in order to be manipulated. In the implementation of these techniques the interface changes mode after selection of an object to the manipulation mode and back to the selection mode when the manipulated object is released by the user.

HOMER Technique: HOMER (Bowman et al., 1997) stands for Hand-centred Object Manipulation Extending Ray-casting technique. The ray-casting selection technique is firstly used to choose an object, then once selected; the users' virtual hand attaches itself to the object and the interface enters the manipulation mode. The technique automatically extends the users' virtual reach or the user to virtual-

hand distance. When in manipulation mode, the user can translate the position of the selected by moving and extending their arm in the interaction space.

Poupyrev et al. (1997) points out that the scaling coefficient used in the HOMER techniques depends on the initial distance between the users' hand and the object in virtual space when selected. This coefficient is larger if the object is further away when the user selects it, therefore the further away the virtual object the larger the amplification is, so the larger the range of manipulation. It's observed that there is a problem if the object is initially located within reach of the user; if the user wants to move the object far away, the scaling coefficient is close to one so the range of manipulation doesn't extend beyond the reach of the user, this renders it impossible to move the object to a distance further away in this case.

Scaled-world grab: A separate technique is the scaled-world grab, developed by Pierce et al. (1997). The user first selects an object using an image-plane technique (Pierce et al., 1997). Once an object is selection, Pierce et al. explains that the technique 'scales down the whole virtual environment around the users' virtual viewpoint', Once this occurs, all objects are within the users reach to be easily manipulated by the user virtual hand. The scaling coefficient for each object is calculated so that the visual size of each individual object in the environment remains unchanged. Therefore, the user is unaware that the scaling has taken place, as there are no real-world clues.

Poupyrev et al. point out that both world-scale grab and the HOMER techniques do not maintain a consistent 'kinaesthetic correspondence' between the user's actual movement and the resulting movement of the virtual hand in the virtual environment. As the scaling coefficient depends on the position of the selected object, this leads to the same displacement of the hand not resulting in a consistent displacement of the virtual hand. 'The inconsistent visual feedback in motor movements usually results in a decrease in the operator performance, since the user cannot effectively built a "kinaesthetic model" of the hand motion' (Smith, 1993). This also goes against the initial heuristic principles of designing an interface, in that consistency in interface procedures is not kept, as already stated this causes actions within the interface to have misleading reactions for the user.

It is also worth noting in this review the observations of Poupyrev et al on the evaluation of techniques for interaction tasks. This is said to be difficult due to the diverse variables that affect user performance when executing a manipulation task. For example, the direction of movement applied by the user, whether inward or outward, left to right, has an impact on user performance, Poupyrev et al states that a reason for this could be the use of different groups of muscles and anatomical constraints that are active in different task conditions.

2.4 Existing Systems

This first notable research into the usability of 3D interfaces in an existing system involved the virtual reality environments CAVE and Powerwall. Observations from the user study state that, it was found 'users generally preferred a lack of physical stress over system responsiveness when manipulating virtual objects' (Sreeram et al, 2007, p. 228). Various reasons for this could include; a lack of precision and control in larger movements, user fatigue and so on. Observations of this experiment also found that, 'users generally desired freedom of movement for in-world navigation, placing great importance on making interaction with the virtual environment as immersive as possible.' (p. 228) It has been previous suggested that the more immersive the interface and its interactions, the more engaging the user becomes with it, and this is because the user is more focused on the content represented by the interface rather than the task of navigating it.

Although of a slightly different form, perhaps the most popular example of gesture-based interaction is the Nintendo Wii. Gamers hold one of a variety of controllers and make different motions to control an application; typically a game. It has been widely successful and despite being used almost exclusively in a private or shared space, offers a strong mental link to systems similar to the one presented in this report. However, users often find the use of the Wii interface difficult to map to the screen, especially if they are left handed. There is a distinct lack of geometric constraints on the controllers' pointer and the user is frequently unsure of where the controller is pointing. This occurs because the feedback, although meant to be unobtrusively minimal, sometimes cannot be seen on the texture of the game interface.

Focusing on the benefits of real-world use of geometric constraints, Jacob et.al (2008) mentions that real-world physics in the form of inertia and springiness is found on the iPhones' interface. 'When scrolling to the bottom of window, it appears to be connected to the bottom of the screen as if by springs. When flicking through the contact list, a fast flick will keep the contacts scrolling after the user's finger has been removed, as if the list itself has mass.' (Jacob et al. 2008, p.7) Geometric constraints or an adoption of real-world experience has made the behaviour of the interface firstly predictable to the user, and also more controllable without limiting the actual navigation of the interface.

2.5 Summary

In this section important points raised in the literature review will be summarised. Firstly, interaction should be precise so what the user believes themselves to be interacting with corresponds with what the interface system registers the user is trying to interact with. This is even more important with direct manipulation interfaces where the interface graphics are mapped with the interaction space. If the mapping between the interaction space and the interface do not match this leads to the sense of immersion and causality being lost.

As well as this, the interaction response should be fast. To maintain the 'sense of immersion' and causality mentioned above, the system must respond the user's input in real-time. O'Hagan et al (2002) suggests that the response time to interaction should be in the order of 30Hz if it is to be classed as 'real-time'.

The users' cognitive load (or relying on working memory) should be minimised. It is suggested that this can be achieved by using direct manipulation interfaces that feature a natural representation of objects and actions to hide the feeling of performing tasks through an intermediary. The main principle here is that by allowing the user to directly perceive and interact with virtual objects will lead to a more natural and effective interface. (O'Hagan et al, 2002)

To conclude, a wide range of 3D gesture-based interaction techniques have been developed and evaluated. The early notion that 3D interaction with such interfaces should be an exactly mimic our interaction with the real-world (isomorphism) is not always the most effective method. In fact, the techniques that deviate from a

realistic representation of real-world interaction can extend our physical limitations and make interaction tasks easier and more effective to perform.

Also, gesture-based interaction techniques do not necessarily need to be 6DOF ones or be controlled by totally unrestricted manipulation. Geometrical constraints of interaction tasks can make them easier for the user to perform and actually add *more* control and precision.

Poupyrev et al. states that there is not one universal interaction technique, designing such interfaces is a trade-off between the tasks that need to be performed and how the system applies 'affordances' of the techniques. Techniques should be tuned so that they maximize user performance in the targeted task conditions within the system.

To finish, it has been noted that appropriate and liberal feedback of the state and cause interactions is absolutely essential; Self-regulation solely relies on this feedback from the interface. As well as, accurate spatial and temporal correspondence from user input to interface heavily depend on it.

2.6 Research Models

Beaudouin-Lafon (2001) describes 'an interaction model as a set of principles, rules and properties that guide the design of an interface.' (p. 1) The interaction model defines how the designer implements the interaction techniques used within the interface into a meaningful and consistent experience; the model specifies a set of aesthetics guidelines for the construction of the interaction technique from the users' perspective. The evaluation of specific interaction designs can be based on the interaction models properties. (Beaudouin-Lafon, 2001)

However, little existing literature is concerned with how a 3D gesture-based interface should be. Nielsen et al. suggests firstly defining the desired functions of the 3D gesture-based interaction interface. Secondly, examine gestures the user would employ to perform each action and finally, test the successes and issues of the 3D gesture-based interface by conducting user studies. (Nielsen et al, 2001)

3.0 METHODS OF INVESTIGATION

In order to conduct valuable research addressing the main research questions previously stated; the following section proposes the following methods of investigation and the rationale behind them.

In the broad sense the method of investigation collects empirical evidence in order to determine the effectiveness of the intuitive nature of each gesture-based interaction with relation to the reaction from the interface. The system used in this research was developed with regular testing with unfamiliar subjects to provide continued feedback establishing whether there is a correlation of what the user expects from the interface with each gesture-based interaction and the response on the interface or virtual object.

More specifically, in order to examine how users perceive and learn to use different gesture-based interactive 3D interfaces, it is proposed to not only to use the regular tests with a small number of subjects on each development of the interface but with that a formal user study of such a situation should be conducted. Ideally the study would compare different gesture-based interaction techniques in terms of speed and user experience. Assessing the output would use a series of different methods.

Firstly, by observing the user perform each interaction technique with the interface without any prompting; witnessing each interaction task being learnt and how quick the discovery of the interaction technique was to each user, if at all. The observer would pay particular attention to the speed in which the user learns to use each interaction task and how efficiently they then execute it once learnt. The evaluation of these observations would be compared the theory that the faster the user takes to learn the gesture of the technique, the easier and more intuitive it is to learn (as set out in section 2). Also, common sense gives us the simple theory that, the more subjects that do come to successfully learn the interaction technique without prompt, the easier and more intuitive the technique. Here the amount of successful subjects provides us a measure. Nevertheless, although the results of this method would indicate what interaction techniques are easier and more intuitive to learn relative to each other, there is no qualitative assessment describing how and why this is.

Another method to be discussed is to interview the subject after conducting the tests. This would hope to give clarity to what features and interaction techniques allow users to work intuitively with the system without prior experience with it, as the user can describe their experience in their own words, point out what worked for them and what didn't, discuss where and why they assumed certain affordances when using the interface? Obviously, it is probably best to be aware that the limited memory of the average user could distort the responses to the user a little, especially if the interview is conducted a while after the actual test.

Perhaps useful at this stage, would be to ask the user to carry out a real-time talk through when they are first confronted with the system, this would allow for a more personal insight into what the user is thinking when confronted with the system. Asking for comments on, firstly what they notice about the interface on first sight and what they will try first, and then ask them to describe what interactions they guess execute what command in the system once they have started to try a few interactions. However, this could distract the user from the task of interacting with the interface, potentially negatively affecting the results of the experiment. That said the results would shed light on what the user initially expects from such an interface without being influenced by the way this particular interface functions.

Performance tasks would serve as an indication of how well the user understands the interface and how easy and efficient navigating the interface is to the user. Using this method, an evaluation can be drawn from comparing the result with the theory that; the easier the interaction task to successfully execute consistently a number of times by the user, the more efficient and easy to use the interaction technique is. Measurement factors in this case include the speed at which the task is completed, the number of errors in the attempt and how consistent the execution of the task is when asked to repeat the performance of the task. A theory is that the more times the user can execute the task with the least amount of errors the easier the technique is to use and the better understanding the user possesses of the interface, whereas the speed of execution signifies the techniques efficiency. It is not assumed that these factors necessarily overlap.

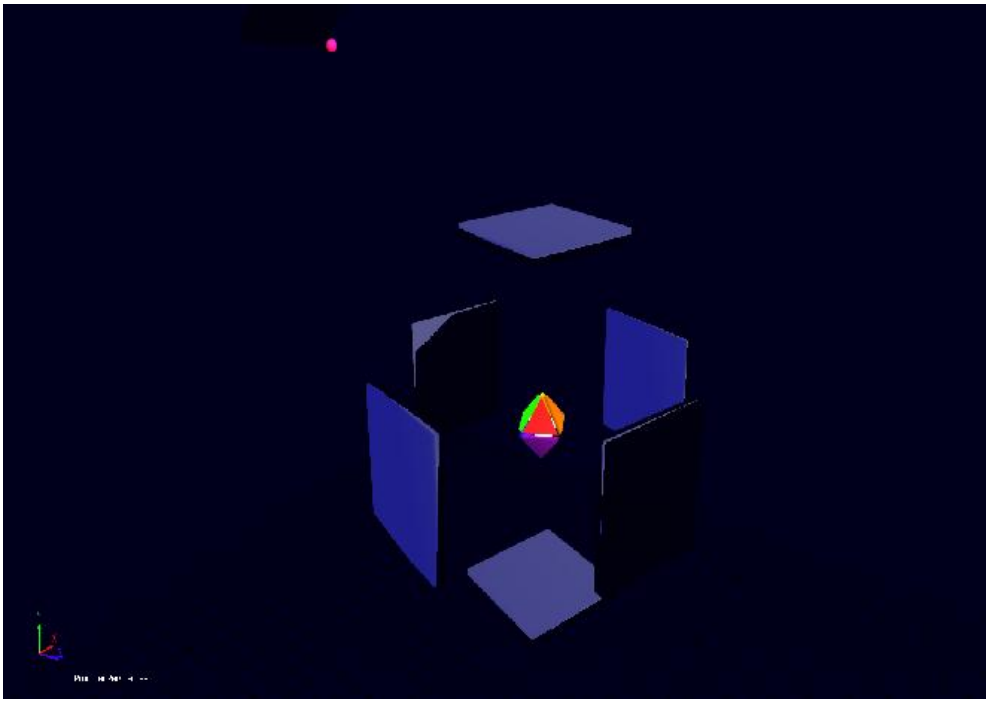
The theory set out by Preece et al, in section 2 is used to compare to empirical results to. Preece et al, states that the easier, the more efficient and the less error prone the interaction techniques is the more engaging and pleasurable the user finds the interface to use (Preece et al, 2001), one obvious reason for this is the reduced amount of frustration felt by the user. So, how happy and engaged the user is with the interface is a measure that could indicate its success. In order to attain this information the subject is asked to complete an anonymous questionnaire.

The questionnaire asks questions such as:

How entertaining is the interface to use?
Would you like to see more gesture-based 3D interactive interface systems in the future?

It should be noted that the use of questionnaires is not an entirely ideal method to use. The use of leading questions as shown above could inadvertently influence the answer given. Yet, it is hard to produce neutral questions without risking a vague answer.

In order to be able to conduct the user study, the development of a functioning 3D interface using gesture-based interaction to control was needed:- SpaceBall is a game developed to aid the conduction of this research. It is based on an existing popular game called *Breakout*. The aim of the game is to bounce the SpaceBall off of the paddles in order to hit and eliminate the boxes in the centre, once all of these boxes have been hit, the game is won. If however the ball bounces outside of the controller sphere, the player loses one of the three balls available to them. If the player loses of their balls, the game is lost. The player can hit the ball by manipulating the controller sphere to bat the ball into the centre of the sphere in order to collide with a box. The sphere is constrained to allow for rotational manipulation only.



The game is meant to facilitate simple gesture-based interaction techniques for the user study in an engaging way. It was decided that game play would be a good basis for the user study as it obvious what user has to achieve in terms of interaction tasks, plus the game aspect of the system would give the user an understanding as to its raison d'être and thus stop any unfounded assumptions as to its functionality that could alter the results.

As well as this, a series of three, visually identical, menu screens were developed, the main aim of this was to explore what '*selection*' techniques (different for each screen) was the most effective and intuitive to use. With the hope of shedding light on what factors help promote easy interaction between system and user.

In the user study on the developed user interfaces and SpaceBall game, a hybrid of the above mentioned methods is to be used. During the study 12 participants were asked to attend, the only previous knowledge the participants had of the study was that it included gesture-based interaction interfaces.[2]

Each subject was to be first individually presented with the menu screens mentioned above, in no particular order.

[2] Ideally the study would involve more than 12 participants to represent a broader range of results. However, in this case, there wasn't the resources available.

One selection techniques was ‘dwell’, where the user must hover over a certain menu icon that appears onscreen for more than a fixed period of time, the selection of the menu icon (represented by a colour change in the object) was mapped directly onto the users hand in using an isomorphic approach. Only once the users hand has been in position for long enough does the interface recognise that the icon has been selected by the user.

The second selection technique tested was the, arm extension method. Again, the mapping was the same as the ‘dwell’ technique, except that if one extended their hand in the z direction (forwards or away from their body) did the interface recognise that the user was selecting the icon.

Lastly, the third technique was based around the ‘double click’ mouse metaphor. The hand to icon mapping and feedback is again the same as the previous two techniques, but the user has to extend their arm (again in the z direction) forward twice before the interface recognises that the user has selected that icon.

Before beginning the experiment the user was asked what they expected from the interface, then once they started interact they were asked to talk a little about what they were attempting and why, and what they expected whilst trying to navigate the interface. At this point the interaction was quietly observed and noted.

Between menus the user is confronted with a transitional state, where the camera slowly zooms into the next interface state and no input is taken. This leads to the SpaceBall game space. Users are presented with the controller sphere and left to their own devices to figure out how to attempt to interact with it. The interaction technique was a metaphor based around rotating real objects using two hands. The performance task here is to attempt to play the SpaceBall game.

Upon completion the subjects were informally interviewed, they were asked questions relating to their talk-through or observed interactions with the system (there were no set questions). Also they were asked to fill out an anonymous questionnaire shortly after. The questions and resulting graphs can be found in Appendix B.

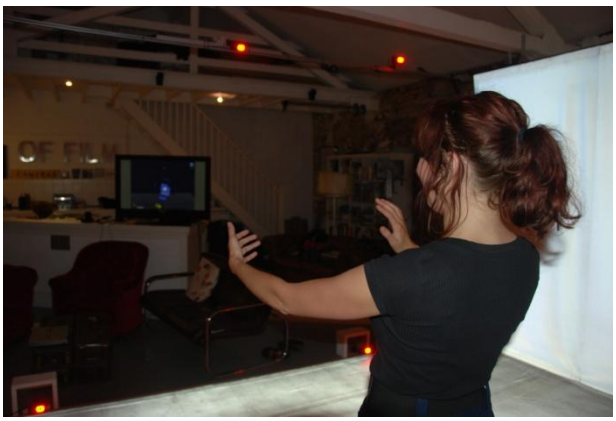
4.0 RESULTS & EVALUATION

In this section the results of the user study will be presented and discussed in relation to the literature review in section 2.

The first observation made from the study proved the theory - as discussed in the literature review - that realism is a particularly intuitive approach. During the experiment when rotating the sphere many subjects expected that interaction would mimic real-life physics, including inertia. Without prompt, they would assume this to be the case with the interface, attempting to start spinning the sphere with a swiping gesture, such as with a globe on a pedestal. Users also expected more detailed functionality from this interaction method, many were surprised when they discovered that the interface didn't support such interactions, this lead to frustration, highlighting the importance of such subtleties.

When conducting interviews with the subjects, the simple question of whether or not they were aware of the way the iPhones' touch screen interface functioned was asked. Those users who were familiar with the touch screen interface gestures expected the interaction with this interface to behave the same. For example, they would start employing the scrolling gesture, and again were surprised when it didn't function in the same way, whereas; those who weren't well versed in this type of technology more commonly expected the gestures to be similar to WIMP style interfaces, such as the 'point and click' metaphor, where usually the hand served as the pointer and clicking gesture was the pointing of a finger toward the virtual object. This worked well in the user study with the 'arm extension' technique (employed in the second menu interface). This could be evidence supporting the theory that cultural influences affect how users expect interface to function. It also points to the concept that existing interfaces create mental models for users to apply when presented with something unfamiliar but similar.

The users vocalised a frustration with the lack of physical feedback in the interaction space, such is the nature of un-instrumented interaction. Mine says 'The absence of constraints is one of the biggest problems encountered in the manipulation of virtual objects. Without constraints, users are restricted to gross interactions and are unable to perform any form of precise manipulation.' (Mine, 1995, p. 2) A few users stated that they would have preferred plenty of visual feedback as a guide when performing a gesture, in the same way it was expressed that the response from the system to the user had to be instantaneous and incredibly obvious, often wanting multiple clues to reveal what was happening. Perhaps it could be suggested that displaying hints to indicate how to cancel or complete the actions would help, such as a visual indication of when a threshold is being approached or something to this effect. Jacob et al.(2008) reports a similar issue: 'The iPhone has a much larger screen but sacrifices the passive haptics associated with hard buttons and the persistence that can allow for efficiency gains and lower cognitive load' (Jacob et al. 2008 p. 7)



It was found that the universal interaction space would vary in effectiveness as people had different reaches, depending on the length of their limbs. Thus, it was observed that interaction with the interface on screen was easier when mapped to the users' height and outward reach, but z-transform gestures didn't have the same effect. It could be argued that this is because most subjects were very experienced with the 2D WIMP GUI where the input device (i.e. the mouse) is mapped to 2D space. To overcome the issue, interactions in the z direction had to be mapped relative to the users root, instead to the virtual space presented onscreen.

It was suggested by a user that the need for thresholds when interacting with virtual objects was needed. A requirement mentioned when the subject was interacting with the rotation gesture in the SpaceBall game. In this instance, a threshold would have been useful only enabling users to interact with the sphere if their hands were outstretched (or above waist height) and at certain proximity from one another, this

is predicted to be the most likely pose for interaction. When implemented, this solved the problems of 'accidentally' interacting which previously lead to errors.

The issue of having the same selection space between menus was a problem when developing the interface. For example, the same positional coordinates from one menu to another sometimes cause the user to be in the same activating space unwittingly; this resulted in unforced user errors that frustrated the user. So, the use of transitions between interface stages were essential, firstly to solve the above problem and secondly, to allow the user to 'get into position'.

The one interaction technique that worked very well in the performance task was the control of the orientation of the sphere. Subjects reported that rotation of the

sphere was very responsive to the hand gesture, it was apparently very easy alternating between one handed or two handed gestures. It was not expected, but users' reported a very good degree of precision when using the gesture. Still, the gesture can be improved as it was limited by the rotations perform the limited range of our hands, so allowing absolute as well as relative motions would extend our freedom. One way of doing this is through cumulative rotations achieved by repeating *rotate-and-pause* gestures.

The greater the number gestures used or the more complex the gestures were when interact with the interface, the longer the time to learn the efficient use of the interface was. Although, a higher complexity level of interface functions and interactions were not very easy for an unfamiliar user to use, once the interface was known, the interaction was faster and more efficient to use. It can be concluded that it is often a choice between designing simpler but intuitive interface system or designing complex systems that allow for fast navigation, but only from experienced users. Perhaps, to overcome this issue; alternative ways of achieving the same task could be simultaneously implemented; one simple and intuitive for

novice users and one complex but fast to execute for the benefit of more experienced users

There are a few issues with the system design when developing the interface. Firstly the Organic Motion Stage motion capture system was very susceptible to light, plus it only tracks one person at a time, limiting the interface to one person interaction only. Other drawbacks included the user always being required perform a T-pose (see Figure 1) when entering the interaction space. Occlusions are also a system problem and constrictive to gesture development. For example, as it tracks ones silhouette, so crossing arms or sitting down causes the skeleton to behave erratically. The lack of fingers or wrist rotations also limits the scope of gestures that can be achieved with the system.

Lastly, it was expected the user would get tired after a certain amount of time interacting with the system, as it required physical actions that could become strenuous to the user (On the other hand, this may solve the nation's current obesity crisis). However, none of the test subjects reported any fatigue, which leads one to think it wasn't a problem as originally thought.

Figure 1: T-Pose

5.0 CONCLUSION

5.1 Report Summary

The main themes covered in this report are concerned with the gestural-based interaction techniques that allow 3D user interface systems to provide their users with an untethered, intuitive method of interaction.

In the literature review I discussed how current universal interface principles provide a model for designing new 3D interfaces. Initially the concept of interface metaphors were explored; it was stated that interface metaphors that provide a user with an interaction technique already familiar to them would allow for an easy mapping to a new domain like the 3D user interface. This familiarity could be based on:

- Analogy of a physical entity in the real-world
- Applying a mental model of a known existing interface
- From cultural influences, such as user interface concepts seen in film

A universal interface framework based around heuristics was then presented, the most relevant principle that proper feedback of system states should be communicated properly to the user, and the results of the user study have highlighted the importance of this issue. It was argued that the reason for this issue to be of such dominant concern when designing 3D untethered interfaces is because, as we have seen in this report, a lack of physical or traditional feedback is presented to the user but also as the extra degree of freedom afforded by such an interface makes navigation of the virtual environment more difficult. So it is concluded that the need for appropriate feedback is all the more important in this instance.

We have also seen that the addition of another dimension contributes to the user lacking precision in their interactions. A solution presented by the literature reviewed in this paper offers the use of geometric constraints and some aspects of an isomorphic approach to add control to the interaction technique. The success of this solution was shown in the existing systems explored, such as iPhone applications.

Furthermore, it was discussed that the use of realism in interaction techniques offers similar benefits to those of the interface metaphor- as the user is already familiar with the concept so can easily transfer this knowledge when using the interaction techniques. The results of the user study demonstrate an actual implementation of this concept and how it aids the users' ability to intuitively understand how an interface functions.

On the other hand, the user study demonstrated where the advantage of using a non-isomorphic approach could be benefited from; such as the physical limitations when manipulating the orientation of a virtual object. We have seen that rotating an object in virtual space is not a simple as expected. The use of quaternion mathematical frameworks was required. Once this was established it was seen that rotation manipulation can be very easy, especially when the orientation is mapped relative to the users other hand in a two-handed gesture.

To conclude, the hypothesis that the gestures which are representative of known actions would serve as the most intuitive when applied to a 3D gesture-based interface, is somewhat correct. Where the use of concepts such as interface metaphors and isomorphism in interaction techniques has resulted in them being be easy to learn and understand. However, the research has pointed out that non-isomorphism can aid the problems brought about by untethered interaction by improving precision and efficiency.

To finish, this report has addressed the main factors in which an intuitive interface should be concerned with. A discussion of the efficiency of certain interaction techniques in providing natural and intuitive interaction with a 3D user interface has been presented. And lastly, this report has documented a user study testing the hypothesis put forward.

5.2 Further Work

Automatic gesture recognition using neural networks, hidden Markov models and other methods - similar to the adaptive gesture recognition functions stated in Caridakis et al - could be used to avoid making users learn another interaction technique. A preferential solution would be to use an adaptive gesture detection algorithm such as the ones mentioned above. An adaptive gesture detection algorithm could conceivably profile the types of gesture the user is making, taking

into account different natural responses such as correction though big, small, fast and slow movements. This would hopefully lead to a more intuitive experience that would in turn promote increased usage. (Hardy, 2008)

6.0 REFERENCES

Albuquerquw, G., Kim, H., Havemann, S. & Fellner, D. W. (2004) *Tangible 3D: Immersive 3D Modelling through Hand Gesture Interaction.* Technical Report: Technical University of Braunschweig TUBS-CG-2004-07

Beaudouin-Lafon, M. (2000, April) *Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces.* Paper presented at ACM Human Factors in Computing Systems (CHI 2000). Den Haag, The Netherlands.

Benhajji, F & Dybner, E. (1999) *3D Graphical User Interface.* Master's Thesis: Stockholm University.

Bowman, D. A., Chen,J., Wingrave, C.A., Lucas, J., Ray, A., Polys, N. F.,

Li, Q., Haciahmetoglu, Y., Kim,J., Kim, S., Boehringer, R. & Ni, T. (2006*) New Directions in 3D User Interfaces.* The International Journal of Virtual Reality, 2006, 5(2):3-14.

Bowman, D. A., Kruijff, E., LaViola, J. & Poupyrev, I. (2004) *3D User Interfaces: Theory and Practice*. Boston: Addison-Wesley.

Bowman, D. A. (1999) *Interaction Techniques for Common Tasks in Immersive Virtual Environments: Design, Evaluation, and Application.* PhD Dissertation. Georgia Institute of Technology.

Buckland, M. (2005) *Programming Game AI by Example.* Texas:Wordware Publishing, Inc.

Burton, N., Kilgour, A. C. & Taylor, H. (1997) *A Case Study in the Use of VRML 2.0 for Marketing a Product, Proceedings of From Desk-Top to Web-Top: Virtual Environments on the Internet, World* Wide Web and Networks*, International Conference*, Bradford pp. 1-24.

Csisinko, M. & Kaufmann, H. (2007) *Towards a universal implementation of 3D user interaction techniques.* Institute of Software Technology and Interactive Systems: Vienna University of Technology.

Daiber, F., Schöning, J. & Krüger, A. (2009) *Whole Body Interaction with Geospatial Data.* In Lecture Notes in Computer Science. Springer: Berlin.

Flasar, J. (2000) *3D Interaction in Virtual Environment.* Faculty of Informatics. Masaryk University. Available from http://cg.tuwien.ac.at [Accessed 06/08/10]

Gutschmidt, T. (2003) *Game Programming with Python, Lua and Ruby.* Boston MA: Premier Press

Jacob, R. J. K., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovery, E. T. & Zigelbaum, J. (2008, April) *Reality-Based Interaction: A Framework for post-WIMP Interfaces.* Paper presented at ACM Human Factors in Computing Systems (CHI 2008). Florence, Italy.

Hand, C. (1997) *A Survey of 3D Interaction Techniques.* Computer Graphics Forum. Volume 016, (1997) number 005 pp. 269–281 Oxford: Blackwell Publishers.

Hardy, J. (2009) *Real-world responses to interactive gesture based public displays.* Dissertation: Lancaster University.

Kim, J., Park, J. & Lee, H. K. C. (2007, August) *HCI(Human Computer Interaction) Using Multi-touch Tabletop Display*. Paper presented at the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. Victoria, Canada.

Kirby, D. (2010) *The Future is Now: Diegetic Prototypes and the Role of Popular Films in Generating Real-world Technological Development.* In Social Studies of Science. 40 (1): 41-70. Sage Publications

Kolsch, M. & Martell, C. (2006, March) *Towards a Common Human Gesture Description Language*. Workshop on Mixed Reality User Interfaces, *Proceedings of IEEE Virtual Reality*, Alexandria, VA.

Liang, J. & Green, M. (1994*). JDCAD: A Highly Interactive 3D Modelling System.*

Computer & Graphics, vol. 4, pp. 499-506.

Kwon, D. Y., & Gross, M. (2007) *A Framework for 3D Spatial Gesture Design and Modeling Using a Wearable Input Device.* Proceedings of the 11th IEEE International Symposium on Wearable Computers (Boston: USA, October, 2007), pp. 95-101.

Lutz, M. (2009) *Learning Python.* Cambridge: O'Reilly

Mapes, D.P. and Moshell J.M (1995) *A Two-Handed Interface for Object Manipulation in Virtual Environments.* Presence 4(4): 403-416.

Mine, M. R. (1995) *Virtual Interaction Techniques.* University of North Carolina Computer Science Technical Report TR95-018

Moeslund, T. B. (2000) Interacting with a Virtual World through Motion Capture. In [Eds.] Granum, E., Holmqvist, B., Kolstrup, S., Madsen, K. H. & Qvortrup, L. (2000) *Virtual Interaction: Interaction in Virtual Inhabited 3D Worlds.* Springer: New York.

Nielsen, M., Störring, M., Moeslund, T. B. & Granum, E. (2003) *A procedure for developing intuitive and ergonomic gesture interfaces for HCI.* Aalborg University, Laboratory of Computer Vision and Media Technology

O'Hagan, R. G., Zelinsky, A. & Rougeaux, S. (2002) Visual Gesture Interfaces for Virtual Environments. *Interacting with Computers*, 14, 231-250

Parks, J. (2007) *Python Scripting for MotionBuilder Artists Masterclass CD-ROM.* SIGGRAPH

Pierce, J., Forsberg, A., Conway, M., Hong, S., Zeleznik, R. & Mine, M. (1997) *Image Plane Interaction Techniques in 3D Immersive Environments*. Proceedings of the ACM Symposium on Interactive 3D Graphics, pp. 39-44.

Poupyrev, I., Billinghurst, M., Weghorst, S., & Ichikawa. T. (1996, November) *The Go-Go Interaction Techniques: Non-linear Mapping for Direct Manipulation in VE.* Paper presented at ACM Symposium on User Interface Software and Technology (UIST), Seattle, US.

Preece, J., Rogers, Y. & Sharp, H. (2002) *Interaction Design: beyond human-computer interaction.* John Wiley & Sons, Inc: Massachusetts.

Sato, Y., Saito, M. & Koike, H. (2001) *Real-Time Input of 3D Pose and Gestures of a User's Hand and Its Applications for HCI.* IEEE Virtual Reality Conference 2001: Yokohama, Japan.

Smith, R. E. (1993) *Enhancing human performance: A psychological skills approach*. Minneapolis/St. Paul: West Publishing Company

Sreeram, S., Zurita, E. S. & Plimmer, B. (2007, November) *3D Input for 3D Worlds.* Paper presented at OzCHI 2007, Adelaide, Australia.

Sturman, D. J. and Zeltzer, D. A survey of glove-based input, *IEEE Computer Graphics and Applications*, Vol. 14, January 1994, pp. 30-39.

Sutphen, S., Sharlin, E., Watson, B. & Frazer, J. (2000) *Reviving a Tangible Interface Affording 3D Spatial Interaction.* WCGS 2000 (Western Computer Graphics Symposium), Panorama, British Columbia, Canada, March 2000.

Vogel, D., & Balakrishnan, R. (2004) *Interactive Public Ambient Displays: Transitioning from Implici to Explici, Public to Personal, Interaction with Multiple Users.* UIST: Santa Fe

Weiser, M. (1991) *The Computer for the Twenty-First Century*. Scientijk American, 265 (3), 94- 104.

Yang, X. B., Choi, S. H., Yuen, K. K. & Chan, L. K. Y. (2010) An Intuitive Human-Computer Interface for Large Display Virtual Reality Applications. *Computer-Aided Design & Applications, 7(2), 269-278: CAD Solutions, LLC.*

APPENDIX A: GLOSSARY OF TERMS AND ACRONYMS'

GUI - Graphical User Interface

HCI – Human Computer Interaction

HMD – Head Mounted Display

ORSDK – Open Reality Software Development Kit

UI - User Interface

WIMP Interfaces:- Most GUIs usually feature windows, icons, menus, and pointers (WIMP). The window is useful for dividing the screen into different sections, where each section can be used for different tasks. Icons are contained within each window instance; these are small pictures of system objects representing commands, files, folders of files, applications and so on. Menus are used to consolidate a set of commands that the user can select from a list. The pointer is the symbol that is shown on screen, it is mostly always controlled by a mouse, it uses the 'point and click' metaphor to select and execute objects and commands.

The Gestalt Law of Psychology- Gestalt psychology is a school of psychology that emphasises patterns of organisation. "gestalt" is a German word, which translates to "form, shape, pattern, or organised whole". The basic belief is that the whole is greater than the sum of its parts. The gestalt psychologists catalogued a number of laws of perceptual grouping which provide evidence for the nature of low-level visual processing, as well as a clue to the nature of the visual representation. They suggest that people group and interpret stimuli in accordance with four laws of perceptual organisation: proximity, similarity, closure and continuity (Smith, 1993)

Mathematical framework of the go-go techniques:- The conversion between the real and transformed coordinate system is defined by the following equation expressing the relation between the real (*Rr*) and virtually transformed (*Rv*) vector lengths of the virtual hand-users hand distance:

$$Rv = \begin{cases} Rr & Rr < D \\ Rr + k \cdot (Rr - D)^2 & Rr \geq D \end{cases}$$

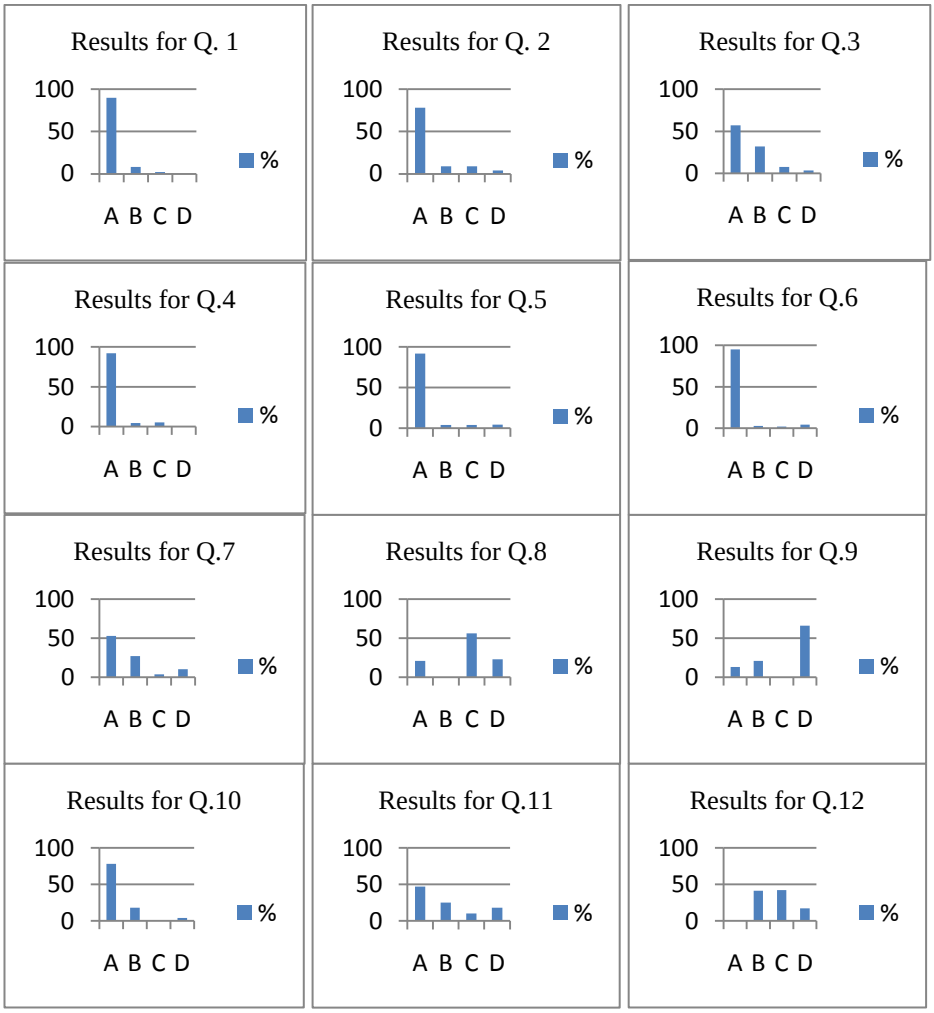
D determines the radius of the sphere around the user separating near-by objects and those located too far away to be in the user's reach. *k* is a coefficient in the range $0<k<1$ specifying the scaling factor for the non-linear coordinate transformation part. (Csisinko, 200)

APPENDIX B: USER STUDY OF SPACEBALL GAME AND INTERFACE

Table B.1 User Study Questionnaire

Questions	A	B	C	D
1. Overall, how do you rank SpaceBall a new concept for a 3D game?	Very exciting	Exciting	Moderately exciting	Unexciting
2. Overall, how do you rank SpaceBall as a concept of interacting with 3D interface?	Very exciting	Exciting	Moderately exciting	Unexciting
3. In your view how does this technology promote 3D gesture-based interaction?	Very helpful in promoting 3D gesture-based interaction	Some help in promoting 3D gesture-based interaction	Little help in promoting 3D gesture-based interaction	No help in promoting 3D gesture-based interaction
4. Overall, how do you feel about using 3D gestures to interact with the system?	Comfortable and intuitive	Moderate, same as controller based game play	Difficult and sometimes uncomfortable	Nervous and uneasy.
5. Would you like to see more gesture-based 3D interactive interface systems such as this in the future?	Yes, looking forward to trying it.	Yes, it might be a good idea	I don't really care	No, I don't think it works well
6. How entertaining is the SpaceBall game to you?	Very fun! I really enjoyed it.	Good fun for playing occasionally	It is about the same as other games	I don't like this game, it's not entertaining
7. How did you feel about using the 'Dwell' selection technique?	Comfortable and intuitive	Moderate, some initial difficulties	Very difficult and uncomfortable	It did not work
8. How did you feel about using the 'Arm extension' selection technique?	Comfortable and intuitive	Moderate, some initial difficulties	Very difficult and uncomfortable	It did not work
9. How did you feel about using the 'Double Tap' selection technique?	Comfortable and intuitive	Moderate, some initial difficulties	Very difficult and uncomfortable	It did not work
10. How easy was it to rotate the SpaceBall controller sphere?	Perfectly easy	Very easy after initial uncertainty	Not very easy	It did not work
11. Did you feel it was obvious how to rotate the sphere?	Blindingly obvious	Obvious	Not very obvious	Very unclear
12. How easy did you find playing the SpaceBall game?	Too easy	Easy	Moderately difficult	Difficult to impossible

B.2 User Study Result Graphs



APPENDIX C: OVERVIEW OF MOTION CAPTURE TECHNOLOGIES

Acoustic

Acoustic tracking devices use high frequency sound emitted from a source component that is placed on the hand or object to be tracked. Microphones placed in the environment receive ultrasonic pings from the source components to determine their location and orientation. In most cases, the microphones are placed in a triangular fashion and this region determines the area of tracked space. One of the most interesting problems with this type of tracking is that certain noises such as jingling keys or a ringing phone will interfere with the device.

Mechanical

Mechanical trackers have a rigid structure with a number of joints. One end is fixed in place while the other is attached to the object to be tracked (usually the user's head). The joint angles are used to obtain position and orientation records. The Fakespace BOOM uses this type of tracking technology.

Advantage include; portable - needs no dedicated space, quick setup - can be calibrated quickly using presets, unaffected by metal objects, no occlusion, line of sight not required, theoretically no distance limit, low lag. Disadvantages include; requires magnetic sensor for accurate root position, gyroscope required for root rotation (available in Gypsy 4), exoskeleton may be undesirable, may be inflexible in capturing all movements, markers set in place, only rotational information captured - hierarchy of rotations used for positional, cheaper than optical, not widespread, smaller support community and may wear down.

Magnetic

Magnetic tracking uses a transmitting device that emits a low frequency magnetic field that a small sensor, the receiver, uses to determine its position and orientation relative to a magnetic source. These trackers can use extended range transmitters which increase the

range of the device from around an 8 foot radius to anywhere from a 15 to 30 foot radius. The tracker shown in the picture is called the Ascension MiniBird. It uses a smaller emitter and receivers and has better accuracy than the regular system. However it's range is limited to about a 4 foot radius. It is primarily used in medical applications where range of the device is not a factor.

Advantages include; 6dof natively, very mature technology – widespread, wide range of options, line of sight not required, usually more affordable. Disadvantages includes; Electromagnet used, accuracy affected by metal objects, sensors and wires may be bulky (1" cube and wire), some latency, small area unless expensive extenders used, large number of sensors expensive.

Inertial

Inertial tracking systems use a variety of inertial measurement devices such as gyroscopes, servo accelerometers, and micro-machined quartz tuning forks. Since the tracking system is in the sensor, range is limited to the length of the cord which attaches the sensor to the electronics box. Two of the big limitations of these devices is that they only track orientation and are subject to error accumulation. The InterSense IS300 handles error accumulation by using a gravitometer and compass measurements to prevent accumulation of gyroscopic drift and also uses motion prediction algorithms to predict motion up to 50 milliseconds into the future.

Advantages include; self-contained, cost and reliability improving, uses Earth's magnetic field rather than transmitter. Disadvantages include: drift, interface more difficult, costly, no native root translation.

Hybrid

Hybrid trackers attempt to put more than one tracking technology together to help increase accuracy, reduce latency, and, in general, provide a better virtual environment experience. An example is the InterSense IS600. It combines inertial and ultrasonic tracking technologies which enables the device to attain 6 DOF. The major difficulty with hybrid trackers is that the more components added to the system, the more complex the device becomes. Other types of hybrid tracking include the combination of video cameras and structured digital light projectors.

Combining these two technologies allow for the capture of depth, colour, and surface reflectance information for objects and participants in the environment.

Optical

At least two cameras view markers in 2D and triangulate for 3D. Light or reflective sources on the tracked object have position determined by multiple cameras. Two types Passive and Active. E.g. Optitrack (passive) or the Phasespace (active).

Camera-based tracking involves the notion of triangulation. Triangulation is the process by which you deduce a position in space based on two or more data sources which taken separately, do not convey enough information to solve the equation alone. To perform optical (camera) based tracking, you need, first of all, a fast computing device able to solve the triangulation equations in real-time (for example, fast enough to provide 16ms or faster rates, steadily). At its root, the system is composed of many equations. By solving a multi-equations system, you can solve for the x, y and z components of one or many points in space. Each equation is linked to a 2D image of the scene or to some other knowledge you have from the scene. This is what we call in mathematics the known and unknown. The first being the information provided from known conditions or other information sources and the later being the x, y and z variable you want to solve for. Multiple camera triangulations: this type of system uses two or many cameras that are separated by known distances. Each camera sees the same scene from a slightly different angle or view point. Active or passive target points are placed on the tracked object. The same point is seen by all cameras with slight shifts due to perspective changes. These shifts are computed by the computer using the equations. If you can solve all the unknowns from this set of equations, you end up with the x, y, z coordinates of the point. When using dynamic targets, we use high output LEDs that can be turned on or off at specific times so that the system can hide those targets it is not currently interested in and so on. For passive targets, since they are always all visible, it takes an even more sophisticated vision algorithm to know which is which in the 2D images as seen from the cameras

Advantages include; Accurate and fast (.2mm and up to 10,000 fps), not affected by metal, no electromagnets, infrared is invisible to photography, mature technology, flexible placement of markers (if any), flexibility in using props. Disadvantages include; Ideally needs dedicated space, longer calibration times,

occlusion problems, lines of sight required, can be expensive (active usually cheaper), markers need to be reapplied each session, active units may be bulky

APPENDIX D: LILA SYSTEM DESCRIPTION

This section will describe the LILA system setup details. Below is a diagram representing the full hardware system. Presented, is a description of the Organic Motion Stage hardware components, its functions and connections as well as some setup procedures. Then, a full pipeline of Autodesks' MotionBuilder software and its ORSDK C++ Plug-ins is described. This is followed by a workflow of the SpaceBall game engine architecture.

Hardware: Organic Motion Stage Motion Capture

The Organic Motion Stage is an advanced optical tracking system. The system uses 14 gray-scale cameras at 120fps, equally positioned around the interaction space, which is of a reflective background and can be up to 4m x 4m x 2.5m. These are fed into a specialised vision processor that syncs the cameras and tracks a 3D mesh from the silhouette of the user. A digital skeleton consisting of 21 bones with six degrees of freedom is then calculated and outputted in the form of Biovision Hierarchical motion capture data (.bhv). This is fed to a computer running the MotionBuilder 9 Organic Motion plug-in via an Ethernet connection. Calibration of the system is automatic upon start-up (see Figure D.2), actor calibration is also automatic upon acquiring a T-Stance from the user (see Figure D.1). The scene within MotionBuilder 9 software is outputted to the projection screen shown to the user in the interaction space.

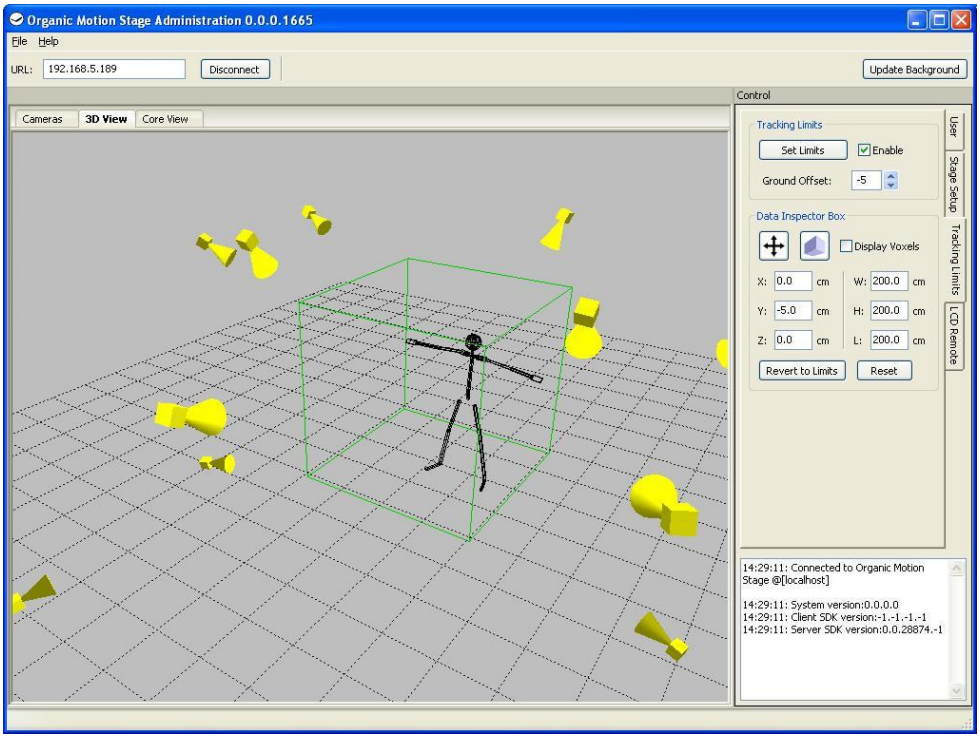


Figure D.1

Figure D.2

Software: MotionBuilder 9 and ORSDK C++ Plug-ins

Autodesks' MotionBuilder software is a real-time 3D character animation pipeline. The software is of particular interest to this project because of its specialism in

motion capture and real-time animation rendering with an inbuilt physics engine. The interface and 3D game is designed within a scene in MotionBuilder.

Open Reality is a C++ software development kit (SDK) allows the integration of new components into MotionBuilder software. The Open Reality SDK gives access to devices, constraints and manipulators within MotionBuilders' real-time architecture. The ORSDK is threaded with this architecture, producing a platform to develop fully manipulative elements without any lag or interrupting the physics simulation of the animation. A plug-in for MotionBuilder is developed in C++ using the Open Reality SDK. Objects are compiled into Dynamically Linked Libraries. These plug-in files are placed in the plug-ins sub-directory for execution with MotionBuilder, which detects them on start-up.

SpaceBall 3D Game

The SpaceBall game engine is built from an ORSDK plug-in that resides within a box in the relation constraint hidden in the scene; this means it is called on every frame of the animation. Boxes within the relation constraint are nodular, meaning they take input and output through nodes specified in the C++ classes. However, the relation box has access to all animation models in the scene and most functionality within the software in real-time, as well as its own namespace to store game and system variables or states.

The SpaceBall Game plug-in is built based around a State-Driven Engine programming model. (Buckland, 2001) Depending on the state the user is in within the game (i.e. interface menu, level 1/2 etc) determines the functionality of the plug-in. For example, if the player is in the interface menu they can interact with menu buttons by directly manipulating the buttons on screen (mapped to the interaction space) by gesturing, the program will react accordingly, changing states if required. Whereas, if the player is within the game, they can interact with the control sphere (a group of models within the screen), the program is continuously detecting collisions. If a collision occurs, one of the models disappears (as is the nature of the SpaceBall game) this requires the plug-in the remove the model from the scene by accessing a handle to the software object and changing its parameters. Once all the model have disappeared (requiring some skill from the player), the player has won and another game state change is required (i.e. next level or back to menu).

APPENDIX E: MOTIONBUILDER 9 ORSDK PLUG-IN C++ CODE

Header File

```
#ifndef __ORBOX_TEMPLATE_BOX_H__
#define __ORBOX_TEMPLATE_BOX_H__

//--- SDK include
#include <Ogre.h>
#include <vector>
#include <fbsdk/fbsdk.h>


//--- Registration defines
#define ORBOXTEMPLATE__CLASSNAME    ORBox_Template
#define   ORBOXTEMPLATE__CLASSSTR     "ORBox_Template"

/**    Template for FBBox class.
*/
class ORBox_Template : public FBBox
{
    //--- box declaration.
    FBBoxDeclare( ORBox_Template, FBBox );

public:
    //! creation function.
    virtual bool FBCreate();

    //! destruction function.
    virtual void FBDestroy();

    //! Overloaded FBBox real-time evaluation function.
    virtual bool AnimationNodeNotify(HFBAnimationNode pAnimationNode,HFBEvaluateInfo pEvaluateInfo);

    //! FBX Storage function
    virtual bool FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );

    //! FBX Retrieval function
    virtual bool FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );

private:

};

// Added by Nicola
//--- Registration defines
#define    ORBOXTEST__CLASSNAME    ORBox_Test
#define ORBOXTEST__CLASSSTR       "ORBox_Test"


/**    Template for FBBox class.
*/
class ORBox_Test : public FBBox
{
    //--- box declaration.
    FBBoxDeclare( ORBox_Test, FBBox );

public:
    HFBModel                getCube();
    void                    Activation(HFBModel pCube);
    void                    Activated();

    //! creation function.
    virtual bool FBCreate();

    //! destruction function.
    virtual void FBDestroy();

    //! Overloaded FBBox real-time evaluation function.
    virtual bool AnimationNodeNotify(HFBAnimationNode pAnimationNode,HFBEvaluateInfo pEvaluateInfo);

    //! FBX Storage function
    virtual bool FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );

    //! FBX Retrieval function
    virtual bool FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );

    virtual void OnInsideBox(HFBModel pCube);

protected:

    HFBAnimationNode        mVHandOne;            //!< Input Node: HandOne translation vector
    HFBAnimationNode        mVHandTwo;            //!< Input Node: HandTwo translation vector
    HFBAnimationNode        mR;            //!< Output Node: Rotation.
    Ogre::Quaternion        mTotalRotation;        //!< Store the total rotation
    Ogre::Vector3           mPreviousHandDiff;        //!< Store the previous hand dif
    double                  mInteger, cameraCount, cameraMax, lH1[3], lH2[3], distance, lRot[3],
                            prevHand1[3], prevHand2[3], speedRX, speedLX;
    bool                    mActivation, mActivated, insideBox, tapOne, tapOff, spin, spin2;
    HFBModel                lcamera, lcamerainterest, lcube, lroot; // Pointers to cube and camera
    FBVector3d              lcubeposition, lcameraposition, lposition, rootTranslation;; // position
```

```
    int                     max, script, counter, initialCounter, spinCounter, spinCounter2;
    std::vector<HFBModel>   lcubevector; // vector of cube pointers
    FBColor                 lcolor;
    FBRVector               lRotation;
    Ogre::Vector3           lHand1, lHand2, lDirection, lUp;
    Ogre::Quaternion        lRotThisFrame;
    FBQuaternion            lOutputQuat;

};

// Added by Nicola
//--- Registration defines
#define ORBOXTESTARMEXTENSION__CLASSNAME    ORBoxTest_Extend
#define ORBOXTESTARMEXTENSION__CLASSSTR     "ORBoxTest_Extend"



class ORBoxTest_Extend : public ORBox_Test
{
    //--- box declaration.
    FBBoxDeclare( ORBoxTest_Extend, ORBox_Test );

public:
    virtual void OnInsideBox(HFBModel pCube);
};

// Added by Nicola
//--- Registration defines
#define ORBOXTESTDOUBLETAP__CLASSNAME    ORBoxTest_DoubleTap
#define ORBOXTESTDOUBLETAP__CLASSSTR    "ORBoxTest_DoubleTap"



class ORBoxTest_DoubleTap : public ORBox_Test
{
    //--- box declaration.
    FBBoxDeclare( ORBoxTest_DoubleTap, ORBox_Test );

public:
    virtual void OnInsideBox(HFBModel pCube);
};

// Added by Nicola
//--- Registration defines
#define ORBOXTESTGAME__CLASSNAME    ORBoxTest_Game
#define ORBOXTESTGAME__CLASSSTR    "ORBoxTest_Game"



class ORBoxTest_Game : public FBBox
{
    //--- box declaration.
    FBBoxDeclare( ORBoxTest_Game, FBBox );

public:
    virtual bool FBCreate();
    virtual void FBDestroy();
    virtual bool AnimationNodeNotify(HFBAnimationNode pAnimationNode, HFBEvaluateInfo pEvaluateInfo);
    virtual bool FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );
    virtual bool FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );

protected:

    HFBAnimationNode        mVectorIn;
    HFBAnimationNode        mAllCollided;
    HFBAnimationNode        mDifference;
    HFBAnimationNode        mSmallest;
    double                  lAllCollided, Dot, lDifference[3], min, length[3], count;
    Ogre::Vector3           lX0, lX1, lX2, plane1, plane2, plane3;

};

//--- Registration defines
#define ORSPACEBALL__CLASSNAME    ORSpaceBall
#define   ORSPACEBALL__CLASSSTR     "ORSpaceBall"

/**    Template for FBBox class.
*/
class ORSpaceBall : public FBBox
{
    //--- box declaration.
    FBBoxDeclare( ORSpaceBall, FBBox );

public:
    virtual bool FBCreate();
    virtual void FBDestroy();
    virtual bool AnimationNodeNotify(HFBAnimationNode pAnimationNode,HFBEvaluateInfo pEvaluateInfo);
    virtual void ControlSphere();
    virtual bool didSpaceBallDirectionChange();
    virtual void killNearestTetra();
    virtual bool FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );
    virtual bool FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat );

protected:

    HFBAnimationNode   mLeftHand;
    HFBAnimationNode   mRightHand;
    Ogre::Vector3      OgreLeftHand, OgreRightHand, OgrePreviousLeftHand, OgrePreviousRightHand,
                       OgreHandDirection, OgrePreviousHandDirection, OgreSpaceBallTranslation,
                       OgrePreviousSpaceBallTranslation1, OgrePreviousSpaceBallTranslation2,
                       OgrePreviousSpaceBallTranslation3, OgreTetraTranslation;
    Ogre::Quaternion   lRotThisFrame, lTotalQuaternion;
```

```
    FBQuaternion        lOutputQuat;
    FBRVector           lRotationSphere;
    FBVector3d          rootTranslation, spaceBallTranslation, tetraTranslation;
    double              lLeftHand[3], lRightHand[3], speedLeftX, speedRightX, min, length, lcollision;
    HFBAnimationNode    mRotationSphere;
    HFBAnimationNode    mTest;
    int                 initialCounter, spinCounterLeft, spinCounterRight;
    HFBModel            lroot, ltetra, lspaceBall, index;
    std::vector<HFBModel>   ltetravector; // vector of tetra pointers
    bool                spinLeftBool, spinRightBool, directionChange;
};

#endif /* __ORBOX_TEMPLATE_BOX_H__ */
```

Main Classes

```
//--- Class declaration
#include "orbox_template_box.h"
#include <math.h>


//--- Registration defines
#define ORBOXTEMPLATE__CLASS       ORBOXTEMPLATE__CLASSNAME
#define ORBOXTEMPLATE__NAME        ORBOXTEMPLATE__CLASSSTR
#define ORBOXTEMPLATE__LOCATION    "Plugins"
#define ORBOXTEMPLATE__LABEL       "OR - Box Template"
#define ORBOXTEMPLATE__DESC        "OR - Box Template Long Description"

#define ORBOXTEST__CLASS        ORBOXTEST__CLASSNAME
#define ORBOXTEST__NAME         ORBOXTEST__CLASSSTR
#define ORBOXTEST__LOCATION     "Plugins"
#define ORBOXTEST__LABEL        "OR - Box Test"
#define ORBOXTEST__DESC         "OR - Box Test Description"

#define ORBOXTESTARMEXTENSION__CLASS     ORBOXTESTARMEXTENSION__CLASSNAME
#define ORBOXTESTARMEXTENSION__NAME      ORBOXTESTARMEXTENSION__CLASSSTR
#define ORBOXTESTARMEXTENSION__LOCATION  "Plugins"
#define ORBOXTESTARMEXTENSION__LABEL     "OR - Box Test Arm Extension"
#define ORBOXTESTARMEXTENSION__DESC         "OR - Box Test Arm Extension Description"

#define ORBOXTESTDOUBLETAP__CLASS    ORBOXTESTDOUBLETAP__CLASSNAME
#define ORBOXTESTDOUBLETAP__NAME     ORBOXTESTDOUBLETAP__CLASSSTR
#define ORBOXTESTDOUBLETAP__LOCATION "Plugins"
#define ORBOXTESTDOUBLETAP__LABEL    "OR - Box Test Double Tap"
#define ORBOXTESTDOUBLETAP__DESC     "OR - Box Test Double Tap Description"

#define ORBOXTESTGAME__CLASS        ORBOXTESTGAME__CLASSNAME
#define ORBOXTESTGAME__NAME         ORBOXTESTGAME__CLASSSTR
#define ORBOXTESTGAME__LOCATION     "Plugins"
#define ORBOXTESTGAME__LABEL        "OR - Box Test Game"
#define ORBOXTESTGAME__DESC         "OR - Box Test Game Description"

#define ORSPACEBALL__CLASS              ORSPACEBALL__CLASSNAME
#define ORSPACEBALL__NAME               ORSPACEBALL__CLASSSTR
#define ORSPACEBALL__LOCATION           "Plugins"
#define ORSPACEBALL__LABEL              "SpaceBall"
#define ORSPACEBALL__DESC               "SpaceBall Description"

//--- implementation and registration

FBBoxImplementation    (    ORBOXTEST__CLASS        );    // Box class name
FBRegisterBox        (    ORBOXTEST__NAME,            // Unique name to register box.
                          ORBOXTEST__CLASS,           // Box class name
                          ORBOXTEST__LOCATION,        // Box location ('plugins')
                          ORBOXTEST__LABEL,           // Box label (name of box to display)
                          ORBOXTEST__DESC,            // Box long description.
                          FB_DEFAULT_SDK_ICON         );    // Icon filename (default=Open Reality icon)

FBBoxImplementation    (    ORBOXTESTARMEXTENSION__CLASS        );    // Box class name
FBRegisterBox        (    ORBOXTESTARMEXTENSION__NAME,            // Unique name to register box.
                          ORBOXTESTARMEXTENSION__CLASS,           // Box class name
                          ORBOXTESTARMEXTENSION__LOCATION,        // Box location ('plugins')
                          ORBOXTESTARMEXTENSION__LABEL,           // Box label (name of box to display)
                          ORBOXTESTARMEXTENSION__DESC,            // Box long description.
                          FB_DEFAULT_SDK_ICON         );    // Icon filename (default=Open Reality icon)

FBBoxImplementation    (    ORBOXTESTDOUBLETAP__CLASS        );    // Box class name
FBRegisterBox        (    ORBOXTESTDOUBLETAP__NAME,           // Unique name to register box.
                          ORBOXTESTDOUBLETAP__CLASS,          // Box class name
                          ORBOXTESTDOUBLETAP__LOCATION,       // Box location ('plugins')
                          ORBOXTESTDOUBLETAP__LABEL,          // Box label (name of box to display)
                          ORBOXTESTDOUBLETAP__DESC,           // Box long description.
                          FB_DEFAULT_SDK_ICON         ); // Icon filename (default=Open Reality icon)

FBBoxImplementation    (    ORBOXTESTGAME__CLASS        );    // Box class name
FBRegisterBox        (    ORBOXTESTGAME__NAME,            // Unique name to register box.
                          ORBOXTESTGAME__CLASS,           // Box class name
                          ORBOXTESTGAME__LOCATION,        // Box location ('plugins')
                          ORBOXTESTGAME__LABEL,           // Box label (name of box to display)
                          ORBOXTESTGAME__DESC,            // Box long description.
                          FB_DEFAULT_SDK_ICON           );// Icon filename (default=Open Reality icon)


FBBoxImplementation    (    ORSPACEBALL__CLASS        );   // Box class name
FBRegisterBox          (    ORSPACEBALL__NAME,            // Unique name to register box.
                            ORSPACEBALL__CLASS,           // Box class name
                            ORSPACEBALL__LOCATION,        // Box location ('plugins')
                            ORSPACEBALL__LABEL,           // Box label (name of box to display)
                            ORSPACEBALL__DESC,            // Box long description.
```

```
                            FB_DEFAULT_SDK_ICON       ); // Icon filename (default=Open Reality icon)

/************************************************
*    Returns Cube
************************************************/
HFBModel ORBox_Test::getCube() {

    for ( int i = 0; i < lcubevector.size() ; i++ )
        {
            // get cube position and scaling
            lcube = lcubevector[i];
            lcube->GetVector(lcubeposition);
            FBVector3d lscale = lcube->Scaling;

            // check if hand is in cube bounds
            if ((lH1[0]<=lcubeposition[0]+lscale[0]) && (lH1[0]>=lcubeposition[0]-lscale[0])
                && (lH1[1]<=lcubeposition[1]+lscale[1]) && (lH1[1]>=lcubeposition[1]-lscale[1]))
            {
                insideBox = true; // activate flag
                break; // break loop once found
            }
            else
            {
                // reset cube colour
                FBMesh lmesh(lcube);
                lcolor = FBColor(0.0,0.0,0.0);
                                        lmesh.Materials[0]->Diffuse = lcolor;
                insideBox = false; // deactivate flag
            } // end of "check if in box" statement
        } // end of for loop
    return lcube;
}

/************************************************
 *    Execute Activation
 ************************************************/

void ORBox_Test::Activation( HFBModel pCube )
{
    lcube = pCube;

    if ( cameraCount >= cameraMax ) // camera count statement
    {
        mActivated = true;
        mActivation = false;
    }
    else
    {
        cameraCount += 0.01;
        lcube->GetVector(lcubeposition);
        lcameraposition[0] = cameraCount*lcubeposition[0];
        distance = lcubeposition[1]-150;
        lcameraposition[1] = cameraCount*distance+150;
        lcameraposition[2] = 700-(cameraCount*580);
        lcamera->Translation = lcameraposition;
        lcamerainterest->Translation = FBVector3d(lcameraposition[0], lcameraposition[1], 0);
    } // end of camera count statement
}

/************************************************
 *    Execute Activated
 ************************************************/

void ORBox_Test::Activated()
{
    lHand1 = Ogre::Vector3(lH1[0],lH1[1],lH1[2]);
    lHand2 = Ogre::Vector3(lH2[0],lH2[1],lH2[2]);
    lDirection = lHand2-lHand1;
          double difference = lH2[0] - lH1[0];
          lRot[2] = difference;
          if ( difference < -130 )
          {
                    lcamera->Translation = FBVector3d(0, 150, 700);
                    lcamerainterest->Translation = FBVector3d(0, 150, 0);
                    cameraCount = 0;
                    script = 1;
                    mActivated = false;
          }

          //if ( lHand1[1] > 100 && lHand2[2] > 100 ) {

                    if (spin != true && lHand1.squaredDistance(lHand2)<3500 &&
lHand1.squaredDistance(lHand2)>100) {
                              lRotThisFrame = mPreviousHandDiff.getRotationTo(lDirection);
                              mTotalRotation = lRotThisFrame * mTotalRotation;
                              mPreviousHandDiff = lDirection;

                              lOutputQuat[3] = mTotalRotation.w;
                              lOutputQuat[0] = mTotalRotation.x;
                              lOutputQuat[1] = mTotalRotation.y;
                              lOutputQuat[2] = mTotalRotation.z;

                              FBQuaternionToRotation( lRotation, lOutputQuat );
                    }
          //}

          // have hands passed each other
          lroot = FBFindModelByName("OMTSkeleton:Root");
          rootTranslation = lroot->Translation;
          if (lH1[0] > rootTranslation[0]+30) {
                    speedRX = lH1[0] - prevHand1[0];
```

```
                    if ( speedRX < -5 ) {
                              spin = true;
                    }
          }
          if (lH2[0] < rootTranslation[0]-30) {
                    speedLX = lH2[0] - prevHand2[0];
                    if ( speedLX > 5 ) {
                              spin2 = true;
                    }
          }


          if (spin == true) {
                    if (spinCounter < 140) {
                              spinCounter++;
                              lRotation[1] = lRotation[1] - 10;
                    } else {
                              spin = false;
                              spinCounter = 0;
                    }
          }
          if (spin2 == true) {
                    if (spinCounter2 < 140) {
                              spinCounter2++;
                              lRotation[1] = lRotation[1] + 10;
                    } else {
                              spin2 = false;
                              spinCounter2 = 0;
                    }
          }

          lcube->Rotation = lRotation;
          lRot[0] = lRotation[0];
}

/************************************************
 *    Creation
 ************************************************/
bool ORBox_Test::FBCreate()
{
    // Create the input/output nodes.
    if( FBBox::FBCreate() )
    {
        mVHandOne             = AnimationNodeInCreate (    0, "HandLeft", ANIMATIONNODE_TYPE_VECTOR );
        mVHandTwo             = AnimationNodeInCreate (    1, "HandRight", ANIMATIONNODE_TYPE_VECTOR
);
        mR                    = AnimationNodeOutCreate (    0, "Rotation", ANIMATIONNODE_TYPE_VECTOR
);
        mPreviousHandDiff     = Ogre::Vector3::UNIT_X;
        mTotalRotation        = Ogre::Quaternion(1,0,0,0);
        mActivation           = false;
        mActivated            = false;
        max                   = 1.0;
        cameraMax             = 1.0;
        script                = 0;
        counter               = 0;
                    spinCounter                                 = 0;
                    spinCounter2                    = 0;
                    initialCounter                  = 0;
        tapOne                = false;
        tapOff                = false;
                    spin                                        = false;
                    spin2                                       = false;
        return true;
    }
    return false;
}


/************************************************
 *    Destruction.
 ************************************************/
void ORBox_Test::FBDestroy()
{
    /*
    *    Free any user memory associated to box.
    */
    FBBox::FBDestroy();

}

/************************************************
 *    Real-time engine evaluation
 ************************************************/
bool ORBox_Test::AnimationNodeNotify( HFBAnimationNode pAnimationNode, HFBEvaluateInfo pEvaluateInfo )
{
          if (initialCounter == 0) {

                    lcamera = FBFindModelByName("Camera");
                    lcamera->GetVector(lcameraposition);
                    lcamerainterest = FBFindModelByName("Camera Interest");
                    lcubevector.push_back(FBFindModelByName("Cube"));
                    lcubevector.push_back(FBFindModelByName("Cube 1"));
                    lcubevector.push_back(FBFindModelByName("Cube 2"));
                    lcubevector.push_back(FBFindModelByName("Cube 3"));
                    lcubevector.push_back(FBFindModelByName("Cube 4"));
                    lcubevector.push_back(FBFindModelByName("Cube 5"));
                    lcubevector.push_back(FBFindModelByName("Cube 6"));
                    lcubevector.push_back(FBFindModelByName("Cube 7"));
                    lcubevector.push_back(FBFindModelByName("Cube 8"));
                    lcubevector.push_back(FBFindModelByName("Cube 9"));
                    lcubevector.push_back(FBFindModelByName("Cube 10"));
```

```
                    lcubevector.push_back(FBFindModelByName("Cube 11"));
                    lcubevector.push_back(FBFindModelByName("Cube 12"));
                    lcubevector.push_back(FBFindModelByName("Cube 13"));
                    lcubevector.push_back(FBFindModelByName("Cube 14"));
                    lcubevector.push_back(FBFindModelByName("Cube 15"));
                    lcubevector.push_back(FBFindModelByName("Cube 16"));
                    lcubevector.push_back(FBFindModelByName("Cube 17"));
                    lcubevector.push_back(FBFindModelByName("Cube 18"));
                    lcubevector.push_back(FBFindModelByName("Cube 19"));
                    lcubevector.push_back(FBFindModelByName("Cube 20"));
                    lcubevector.push_back(FBFindModelByName("Cube 21"));
                    lcubevector.push_back(FBFindModelByName("Cube 22"));
                    lcubevector.push_back(FBFindModelByName("Cube 23"));
                    lcubevector.push_back(FBFindModelByName("Cube 24"));
                    lcubevector.push_back(FBFindModelByName("Cube 25"));
                    lcubevector.push_back(FBFindModelByName("Cube 26"));
                    lcubevector.push_back(FBFindModelByName("Cube 27"));
                    initialCounter = 1;
          }

    // Read the data from the input nodes
    mVHandOne->ReadData( lH1, pEvaluateInfo );
    mVHandTwo->ReadData( lH2, pEvaluateInfo );

        // if in first scene
    if ( mActivation==false && mActivated==false )
    {

        lcube = getCube();
        OnInsideBox(lcube);
    } // end of first scene statement


    if ( mActivated==true ) // if in second scene
    {
        Activated();
    }
    else
    {
        if ( mActivation==true ){ // camera move statement
            Activation(lcube);
        } // camera move statement
    } // end of second scene statement

          prevHand1[0] = lH1[0];
          prevHand1[1] = lH1[1];
          prevHand1[2] = lH1[2];
          prevHand2[0] = lH2[0];
          prevHand2[1] = lH2[1];
          prevHand2[2] = lH2[2];
    lRot[1] = script;

    // Write the rotation out.
    mR->WriteData( lRot, pEvaluateInfo );
    return CNT_STATUS_LIVE;

} // end of ORBox_Test

void ORBox_Test::OnInsideBox(HFBModel pCube) {
    if ( insideBox==true ) // inside box statement
    {
        // count to max
        mInteger = mInteger + 0.005;
        if ( mInteger >= max ) // count statement
        {
                            FBMesh lmesh(pCube);
            lcolor = FBColor(0.0,1.0,0.0);
            lmesh.Materials[0]->Diffuse = lcolor;
            mActivation = true;
            mInteger = 0;
            script = 0;

        }
        else
        {
            FBMesh lmesh(pCube);
            lcolor = FBColor( 0.0, mInteger, 0.0 );
            lmesh.Materials[0]->Diffuse = lcolor;
            mActivation = false;
        } // end of count statement
    }
    else // outside of box
    {
        mInteger = 0;
    } // end of inside box statement

}

void ORBoxTest_Extend::OnInsideBox(HFBModel pCube) {
    if ( insideBox==true ) // inside box statement
    {
        lroot = FBFindModelByName("OMTSkeleton:Root");
                    rootTranslation = lroot->Translation;
                    if ( lH1[2] < rootTranslation[2]+50.0 ) // count statement
        {
            FBMesh lmesh(pCube);
            lcolor = FBColor(0.0,1.0,0.0);
            lmesh.Materials[0]->Diffuse = lcolor;
            mActivation = true;
            mInteger = 0;
            script = 0;
```

```
        }
        else
        {
            FBMesh lmesh(pCube);
            lcolor = FBColor(1.0,0.0,0.0);
            lmesh.Materials[0]->Diffuse = lcolor;
            mActivation = false;
        } // end of count statement
    }
}

void ORBoxTest_DoubleTap::OnInsideBox(HFBModel pCube) {

    if ( insideBox==true ) // inside box statement
    {
        if ( tapOne==true )
        {
            counter++;
        }
        if ( counter > 500 )
        {
            tapOne = false;
        }
                lroot = FBFindModelByName("OMTSkeleton:Root");
                rootTranslation = lroot->Translation;
        if ( lH1[2] < rootTranslation[2]+50.0 ) // count statement
        {
            tapOne=true;
            if ( tapOff==true && counter<500 )
            {
                FBMesh lmesh(pCube);
                lcolor = FBColor(0.0,1.0,0.0);
                lmesh.Materials[0]->Diffuse = lcolor;
                mActivation = true;
                mInteger = 0;
                script = 0;
                tapOne = false;
                tapOff = false;
            }
        }
        else
        {
            FBMesh lmesh(pCube);
            lcolor = FBColor(1.0,0.0,0.0);
            lmesh.Materials[0]->Diffuse = lcolor;
            mActivation = false;
            if ( tapOne==true )
            {
                tapOff=true;
            }
        }
    }
    else
    {
        tapOff = false;
        tapOne = false;
        counter = 0;
    }
}

/************************************************
 *    FBX Storage.
 ************************************************/
bool ORBox_Test::FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat )
{
    /*
    *    Store box parameters.
    */
    return true;
}


/************************************************
 *    FBX Retrieval.
 ************************************************/
bool ORBox_Test::FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat )
{
    /*
    *    Retrieve box parameters.
    */
    return true;
}

/************************************************
 *    Creation
 ************************************************/
bool ORBoxTest_Game::FBCreate()
{
    /*
    *    Create the nodes for the box.
    */
    mVectorIn        = AnimationNodeInCreate (    0, "Vector In", ANIMATIONNODE_TYPE_VECTOR );
    mAllCollided    = AnimationNodeOutCreate (    1, "All Collided", ANIMATIONNODE_TYPE_NUMBER );
    mDifference        = AnimationNodeOutCreate (    2, "Difference", ANIMATIONNODE_TYPE_VECTOR );
    mSmallest        = AnimationNodeOutCreate (    3, "Smallest", ANIMATIONNODE_TYPE_NUMBER );
    lAllCollided    = 0;
    lX0 = Ogre::Vector3(0,0,0);
    lX1 = Ogre::Vector3(0,0,0);
    lX2 = Ogre::Vector3(0,0,0);

    Dot = 1;
    return true;
```

```
}


/************************************************
 *    Destruction.
 ************************************************/
void ORBoxTest_Game::FBDestroy()
{
    /*
    *    Free any user memory associated to box.
    */

}


/************************************************
 *    Real-time engine evaluation
 ************************************************/
bool ORBoxTest_Game::AnimationNodeNotify( HFBAnimationNode pAnimationNode, HFBEvaluateInfo
pEvaluateInfo )
{
    /*
    *    This class serve as a test bed for the implentation of different functions without
    *    having to alter the game engine code
    *
    *
    */
    HFBModel        lPlane1, lPlane2, lPlane3;
    FBVector3d        mPlane1, mPlane2, mPlane3;
    lPlane1 = FBFindModelByName("Plane");
    mPlane1 = lPlane1->Translation;
    plane1[0] = mPlane1[0];
    plane1[1] = mPlane1[1];
    plane1[2] = mPlane1[2];
    lPlane2 = FBFindModelByName("Plane 1");
    mPlane2 = lPlane2->Translation;
    plane2[0] = mPlane2[0];
    plane2[1] = mPlane2[1];
    plane2[2] = mPlane2[2];
    lPlane3 = FBFindModelByName("Plane 2");
    mPlane3 = lPlane3->Translation;
    plane3[0] = mPlane3[0];
    plane3[1] = mPlane3[1];
    plane3[2] = mPlane3[2];

    double            lVectorIn[3];
    int                lStatus;
    lStatus            = mVectorIn->ReadData( lVectorIn, pEvaluateInfo );

    if( lStatus != CNT_STATUS_DEAD ) {
        Ogre::Vector3 lNewPos = Ogre::Vector3(lVectorIn[0],lVectorIn[1],lVectorIn[2]);
        if((lNewPos-lX2).squaredLength() > 0.0000001f) {

            lX0 = lX1;
            lX1 = lX2;
            lX2 = lNewPos;

            if((lX0-lX1).squaredLength() > 0.0001) {
                Ogre::Vector3 lV0 = lX1- lX0;
                Ogre::Vector3 lV1 = lX2- lX1;
                lV0.normalise();
                lV1.normalise();
                if(lV1.dotProduct(lV0) < 0.85f) {
                    min = 10000000;
                    double index = -1;
                    length[0] = (plane1-lNewPos).squaredLength();
                    length[1] = (plane2-lNewPos).squaredLength();
                    length[2] = (plane3-lNewPos).squaredLength();

                    for ( int i = 0; i<3; i++ ) {
                        if ( length[i] < min ) {
                            min = length[i];
                            index = i;
                        }
                    }

                    if ( min < 5000 ) {
                        if ( index==2 ) {
                            if ( lPlane3->Components.GetCount() != 0 ) {
                                HFBComponent rb3 = lPlane3->Components[0];
                                lPlane3->Components.Remove(rb3);
                                lPlane3->Show = false;
                                }

                        }
                        if ( index==1) {
                            lPlane2->Show = false;
                        }
                        if ( index==0 ) {
                            lPlane1->Show = false;
                        }
                    }


                    lAllCollided += 1;
                    lDifference[0] = length[0];
                    lDifference[1] = length[1];
                    lDifference[2] = length[2];

                }
            }
```

```
        }

        mAllCollided->WriteData( &lAllCollided, pEvaluateInfo );
        mDifference->WriteData( lDifference, pEvaluateInfo );
        mSmallest->WriteData( &count, pEvaluateInfo );
        return CNT_STATUS_LIVE;
    }
    return CNT_STATUS_DEAD;
}


/************************************************
 *    FBX Storage.
 ************************************************/
bool ORBoxTest_Game::FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat )
{
    /*
    *    Store box parameters.
    */
    return true;
}


/************************************************
 *    FBX Retrieval.
 ************************************************/
bool ORBoxTest_Game::FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat )
{
    /*
    *    Retrieve box parameters.
    */
    return true;
}

// SPACEBALL GAME PLUGIN

/************************************************
 *    Creation
 ************************************************/
bool ORSpaceBall::FBCreate()
{
          if( FBBox::FBCreate() ) {

        mLeftHand                                           = AnimationNodeInCreate  (    0, "Left
Hand",        ANIMATIONNODE_TYPE_VECTOR );
        mRightHand                    = AnimationNodeInCreate  (    1, "Right Hand",
ANIMATIONNODE_TYPE_VECTOR );
        mRotationSphere               = AnimationNodeOutCreate (    2, "Sphere Rotation",
ANIMATIONNODE_TYPE_VECTOR );
                    mTest                                                   =
AnimationNodeOutCreate (    3, "Collision Count",  ANIMATIONNODE_TYPE_NUMBER );
                    initialCounter                          = 0;
                    spinCounterLeft                             = 0;
                    spinCounterRight                    = 0;
                    lcollision                                              = 0;
                    OgrePreviousSpaceBallTranslation1 = Ogre::Vector3(0,0,0);
                    OgrePreviousSpaceBallTranslation2 = Ogre::Vector3(0,0,0);
                    OgrePreviousSpaceBallTranslation3 = Ogre::Vector3(0,0,0);
                    spinLeftBool                                        = false;
                    spinRightBool                           = false;
                    return true;
          }
          return false;
}



/************************************************
 *    Destruction.
 ************************************************/
void ORSpaceBall::FBDestroy()
{
    /*
    *    Free any user memory associated to box.
    */
}


/************************************************
 *    Real-time engine evaluation
 ************************************************/
bool ORSpaceBall::AnimationNodeNotify( HFBAnimationNode pAnimationNode, HFBEvaluateInfo pEvaluateInfo
)
{

          if (initialCounter == 0) {
                    lroot = FBFindModelByName("OMTSkeleton:Root");
                    lspaceBall = FBFindModelByName("Spaceball");
                    ltetravector.push_back(FBFindModelByName("Tetra 01"));
                    ltetravector.push_back(FBFindModelByName("Tetra 02"));
                    ltetravector.push_back(FBFindModelByName("Tetra 03"));
                    ltetravector.push_back(FBFindModelByName("Tetra 04"));
                    ltetravector.push_back(FBFindModelByName("Tetra 05"));
                    ltetravector.push_back(FBFindModelByName("Tetra 06"));
                    ltetravector.push_back(FBFindModelByName("Tetra 07"));
                    ltetravector.push_back(FBFindModelByName("Tetra 08"));
                    initialCounter = 1;
          }

          mLeftHand->ReadData( lLeftHand, pEvaluateInfo );
    mRightHand->ReadData( lRightHand, pEvaluateInfo );
          OgreLeftHand[0] = lLeftHand[0];
```

```
          OgreLeftHand[1] = lLeftHand[1];
          OgreLeftHand[2] = lLeftHand[2];
          OgreRightHand[0] = lRightHand[0];
          OgreRightHand[1] = lRightHand[1];
          OgreRightHand[2] = lRightHand[2];
          ControlSphere();
          if (didSpaceBallDirectionChange()==true) {
                    killNearestTetra();
          }

          mRotationSphere->WriteData( lRotationSphere, pEvaluateInfo );
          mTest->WriteData ( &lcollision, pEvaluateInfo );
          OgrePreviousLeftHand = OgreLeftHand;
          OgrePreviousRightHand = OgreRightHand;
    return CNT_STATUS_LIVE;
}


/**********************************************
 *    FBX Storage.
 **********************************************/
bool ORSpaceBall::FbxStore( HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat )
{
    /*
    *    Store box parameters.
    */
    return true;
}


/**********************************************
 *    FBX Retrieval.
 **********************************************/
bool ORSpaceBall::FbxRetrieve(HFBFbxObject pFbxObject, kFbxObjectStore pStoreWhat )
{
    /*
    *    Retrieve box parameters.
    */
    return true;
}

void ORSpaceBall::ControlSphere() {

    OgreHandDirection = OgreLeftHand-OgreRightHand;

          if (spinRightBool!=true && spinLeftBool!=true) {
          // if hand are within threshold for interaction then alter rotation of sphere
                    // if above waist height
                    if (OgreLeftHand.y > 100 && OgreRightHand.y > 100) {
                     if close enough together
                              if (OgreLeftHand.squaredDistance(OgreRightHand)<3500 &&
OgreLeftHand.squaredDistance(OgreRightHand)>100) {

                                        lRotThisFrame =
OgrePreviousHandDirection.getRotationTo(OgreHandDirection);
                                        lTotalQuaternion = lRotThisFrame * lTotalQuaternion;
                                        OgrePreviousHandDirection = OgreHandDirection;
                                        lOutputQuat[3] = lTotalQuaternion.w * 0.5;
                                        lOutputQuat[0] = lTotalQuaternion.x;
                                        lOutputQuat[1] = lTotalQuaternion.y;
                                        lOutputQuat[2] = lTotalQuaternion.z;
                                        FBQuaternionToRotation( lRotationSphere, lOutputQuat );
                              }
                    }
          }
          rootTranslation = lroot->Translation;
          // if left hand extends past root by $$$ 60 (OM backwards)
          if (OgreLeftHand.x > rootTranslation[0]+60) {
                    speedLeftX = OgreLeftHand.x - OgrePreviousLeftHand.x;
                    // And if speed is higher than $$$ 5 in positive direction
                    if ( speedLeftX > 5 ) {
                              spinLeftBool = true;
                    }
          }
          // if right hand extends past root my $$$ 60 (OM backwards)
          if (OgreRightHand.x < rootTranslation[0]-60) {
                    speedRightX = OgreRightHand.x - OgrePreviousRightHand.x;
                    // And if speed is lower than $$$ -5 in negative direction
                    if ( speedRightX < -5 ) {
                              spinRightBool = true;
                    }
          }

          if (spinLeftBool == true) {
                    // spin for $$$ 140 frames
                    if (spinCounterLeft < 140) {
                              spinCounterLeft++;
                              lRotationSphere[1] = lRotationSphere[1] - 10; // in negative direction
(left)
                    } else { // reset
                              spinLeftBool = false;
                              spinCounterLeft = 0;
                    }
          }
          if (spinRightBool == true) {
                    // spin for $$$ 140 frames
                    if (spinCounterRight < 140) {
                              spinCounterRight++;
                              lRotationSphere[1] = lRotationSphere[1] + 10; // in positive direction
(right)
                    } else { // reset
                              spinRightBool = false;
```

```
                                spinCounterRight = 0;
                        }
                }

}

bool ORSpaceBall::didSpaceBallDirectionChange() {

                directionChange = false;
                spaceBallTranslation = lspaceBall->Translation; // spaceBall position
        OgreSpaceBallTranslation =
Ogre::Vector3(spaceBallTranslation[0],spaceBallTranslation[1],spaceBallTranslation[2]); // Ogre
conversion
                if ( (OgreSpaceBallTranslation-OgrePreviousSpaceBallTranslation3).squaredLength() >
0.0000001f) { // if spaceBall has moved position frame to frame
                        OgrePreviousSpaceBallTranslation1 = OgrePreviousSpaceBallTranslation2;
                        OgrePreviousSpaceBallTranslation2 = OgrePreviousSpaceBallTranslation3; // average
                        OgrePreviousSpaceBallTranslation3 = OgreSpaceBallTranslation;
                        if((OgrePreviousSpaceBallTranslation1-
OgrePreviousSpaceBallTranslation2).squaredLength() > 0.0001) { // if spaceBall has move position for
more than 3 frames
                                Ogre::Vector3 OgreVelocity1 = OgrePreviousSpaceBallTranslation2 -
OgrePreviousSpaceBallTranslation1;
                                Ogre::Vector3 OgreVelocity2 = OgrePreviousSpaceBallTranslation3 -
OgrePreviousSpaceBallTranslation2;
                                OgreVelocity1.normalise();
                                OgreVelocity2.normalise();
                                if(OgreVelocity2.dotProduct(OgreVelocity1) < 0.85f) { directionChange =
true; }
                        }
                }
                return directionChange;
}

void ORSpaceBall::killNearestTetra() {

                min = 1000000;
                length = 0;
                // find smallest length and assign pointer to index
                for ( int i = 0; i < ltetravector.size() ; i++ ) {
                        ltetra = ltetravector[i];
                        tetraTranslation = ltetra->Translation;
                        OgreTetraTranslation = Ogre::Vector3(tetraTranslation[0], tetraTranslation[1],
tetraTranslation[2]);
                        length = (OgreTetraTranslation-OgreSpaceBallTranslation).squaredLength();
                if ( length < min ) {
                                min = length;
                    index = ltetra;
                                lcollision = length;
                }
                  }

                  // if length is smaller than size of object hide tetra and remove rigid body
                  if ( min < 100000 ) {
                        if ( index->Components.GetCount() != 1 ) {
                                HFBComponent rb3 = index->Components[1];
                                index->Components.Remove(rb3);
                                index->Show = false;
                        }
                  }

}
```

Library Class

```
//--- SDK include
#include <fbsdk/fbsdk.h>

#ifdef KARCH_ENV_WIN
#include <windows.h>
#endif

//--- Library declaration
FBLibraryDeclare( orbox_template )
{
    FBLibraryRegister( ORBox_Template );
    FBLibraryRegister( ORBox_Test ); //Added by Nicola
    FBLibraryRegister( ORBoxTest_Extend ); //Added by Nicola
    FBLibraryRegister( ORBoxTest_DoubleTap ); //Added by Nicola
    FBLibraryRegister( ORBoxTest_Game ); //Added by Nicola
          FBLibraryRegister( ORSpaceBall ); //Added by Nicola
}
FBLibraryDeclareEnd;

/************************************************
 *    Library functions.
 ************************************************/
bool FBLibrary::LibInit()        { return true; }
bool FBLibrary::LibOpen()        { return true; }
bool FBLibrary::LibReady()        { return true; }
bool FBLibrary::LibClose()        { return true; }
bool FBLibrary::LibRelease()    { return true; }
```