Be The Controller: A Kinect Tool Kit for Video Game Control

Recognition of human motion using skeletal relational angles

Nicholas Hadjiminas
Department of Computing,
School of Informatics, City University
London, United Kingdom
n123al@hotmail.com

Christopher Child
Department of Computing,
School of Informatics, City University
London, United Kingdom
C.Child@city.ac.uk

***Abstract*— As technology evolves, more interactive video game controllers are being developed in order to provide players with greater interaction with games. Motion control is one of the most challenging and exciting topics in today's games industry and the related controllers have taken the industry by storm becoming a prerequisite for new generation game consoles.**

We propose a new method to quickly and accurately recognize human motion using skeletal relation angle recognition rather than body parts Cartesian co-ordinates position recognition in order to provide a more flexible, accurate and efficient way tracking human motion. This method was used to develop a tool kit that aims to help game designers easily identify and recognise a user's pose and gestures using Microsoft's Kinect motion controller, in order to then link these movements with actions in a game for example key presses can be liked to poses or gestures in order to produce key events.

Finally, we evaluated the usability of the tool kit and the success rate of skeletal relation angle recognition through an experiment. Ten representative users were selected and asked to complete a set of tasks using the "Be the Controller" project and a game (World of Warcraft) in order to simulate real conditions of use of the software and evaluate its usability. The data from this experiment helped to form some important conclusions: users found it easy to create their own poses and gestures; they were enthusiastic about the fact that they were able to bind their actions to key/mouse events; and were satisfied with the success rate of pose recognitions

Keywords- Pose and gesture recognition; Kinect project; Be the controller tool kit;motion control; human motio; skeletal relational angles recognition

I. INTRODUCTION

As technology evolves, more interactive video game controllers are being developed in order to provide players with greater interaction with games and thus offering more enjoyment and the ability for a player to become more immersed in the game on many different levels, (immersed, refers to how much the user is drawn into the game's world and feels as though he is actually in the gaming environment). Motion control is one of the most challenging topics in today's games industry whereas the related controllers have taken the industry by storm becoming a prerequisite for new generation game consoles [1]. Motion controllers use different types of mechanisms such as, accelerometers, infra-red (IR) sensors and cameras. These help detect the player's movements and identify his gestures in order to allow him to interact with the game and manipulate items on screen through real-time movement and gestures. Contemporary video game controllers use a camera, IR projector and IR sensors to track the user's motion without the need for the user to hold a controller. Microsoft's Kinect holds the Guinness World Record of being the "fastest selling consumer electronics device". It sold an average of 133 units per day with a total of 8 million units sold within the first 60 days of its release [2].

Despite the success of motion controllers they have not resulted in the significant and radical change to the game industry expected. This can be explained with a brief look at the current games market and the use of motion tracking controllers. One could describe the market for motion tracking games as casual, consisting of people with little or no experience with games, who do not like playing games for extended periods and who will probably not read the instruction manual before playing a game [3]. Probably the reason behind this is that motion controllers support more accessible simple motions compared to the traditional gamepads with their many action buttons. However, other game markets, such as the more traditional core games, appear not to share the immense success of casual games with motion tracking . There are only a few releases of core game (like Frist Pearson Shouter games) with motion tracking control and these are not as successful as their casual competitors. This is in direct contrast to recent statistics illustrating traditional core market games as being very successful. For example, Call of Duty: Black Ops has sold over 18 million copies [4]. Finally, a cursory analysis of the casual market's released games, shows that most users use the motion controller's capabilities in a restricted, controlled fashion. This observation regarding the current state of the motion controllers' market gives the impression of market stagnation and there is no obvious explanation as to why this is happening.

Nevertheless, several theories try to explain this stagnation. The first of these is that it is caused by the limitations of the motion controllers [5]. However, this explanation does not seem to be the case as there are several impressive open source implementations that use motion controllers developed by programmers' communities. Another reason considered more valid is that the motion controller market is small, thus the designing of games aimed at this market need considerably more experiences and the available Software Development Kits provide developers with only basic functionalities. For example, available SDKs provide developers with a mechanism to track the user's position in the environment but they do not provide them with a mechanism to identify user's actions. Subsequently, the conclusion is that not enough research has been carried out regarding motion controllers and which mechanisms are needed in order to allow developers to use the motion tracking controllers to their full capabilities. The lack of research regarding the full capabilities of motion controllers and ways to use them in order to identify players' actions were the motivation of the "Be The Controller: A Kinect Tool Kit for Video Game Control" project [6]. The current paper summarizes the work presented in that project

A. *What is the "Be the Controller tool kit" ?*

The primary goal of the Be the Controller tool kit is to help game designers easily identify and recognise a user's pose and gestures using Microsoft's Kinect motion controller, in order to then link these movements with actions in a game. The project is divided into two parts: the first part of the tool kit is a windows application that allows a developer to create a set of gestures and poses and then export these gestures and pose lists to files. The second part of the tool kit is a software library (dll file) that provides the developer with the infrastructure to load lists of poses and gestures to an application, which are then recognised and linked to the appropriate action in the application.

The tool uses a skeletal relation angle recognition method rather than body parts Cartesian co-ordinates position recognition method, which provide a more easy and flexible way to create and recognise pose and gesture. By using this method the tool can recognise poses unaffected by the positioning of the body's parts in the virtual world environment because it only compares the relationship of the parts, (if the positioning of two parts changes but they keep the same angle/relation between them they will still be recognised as the previous pose).

Furthermore, the tool achieved high performance by restricting the comparison to the active relations of poses and by simplifying gestures as a sequence of two poses instead of representing them as a set of continuous changes of the tracked relations.

Finally the "Be the controller App" application provides a way for users to convert the recognised poses and gestures into keyboard and mouse actions, allowing a user with no programing experience to use the system to control contemporary games with the Kinect controller.

II. HUMAN BODY MOVEMENTS

This section looks at the method we suggest to track the movements of a human for this project. First research about the movements of the human body is presented and then a description of the technique used to recognise the user's different actions using the positioning of human body parts is provided.

A. *Understanding Body Movements*

According to Bartlett, the human body movements are usually described in three dimensional space using a system of planes and axis [7]. The three planes of motion the human body passes through are:

- The sagital plane
- The frontal plane
- The transverse (horizontal) plane

The sagital plane is the plane that lies vertically and bisects the body into right and left parts. Sagital plane motions are the movements of human parts around the frontal axis.

The frontal plane also lies vertically however it bisects the body into the front and back parts. Frontal plane motions are the movements of human parts around the sagital axe.

The transverse is the plane that lies horizontally and bisects the body into upper and lower parts. Transverse plane motions are the movements of human parts around the vertical axe [7].

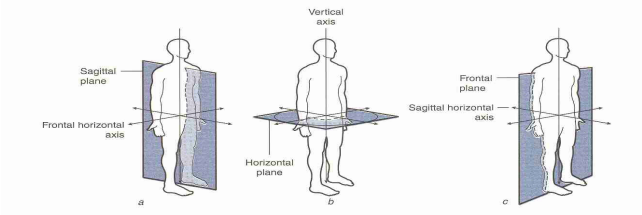

Figure 1. Planes of Human motion [7]

Axis of movement

Human movement at the body's joints happen in a plane, about an axis. The axis of movement for an object is a straight line that the object can rotate around. There are three axes of rotation for the body:

- Sagital axis
- Frontal axis
- Vertical axis

Human motion

The most common way of describing a movement is by referring to the movement's dominant plane. For example, we describe the walking activity as a sagital plane movement. This description does not accurately represent the real movement scheme; it only covers the gross direction of movement. At each skeletal joint movement will occur in several planes not only in a single plane that matches direction of movement. In the example of the walking activity, the hip will flex or extend across the sagital plane, adduct or abduct across the frontal plane and rotate internally or externally across the transverse plane. This scheme of motion applies to all the individual joints of body parts and because the movement on the three planes happens simultaneously, can be seen as one action with three components, a tri-planar motion [8].

All human motions follow this tri-planar scheme so all functional movements occur in three dimensions, though, it is more useful, from the mechanical point of view, when generalising about average motion pattern to describe motion in a single plane scheme. The following table shows examples of motion planes, axis of movements and motions.

Plane	Motion	Axis	Example
Sagital	Flexion/extension	Frontal	Walking Squatting Overhead press
Frontal	Abduction/abduction Side flexion Inversion/eversion	Sagital	Star jump Lateral arm raise Side bending
Frontal	Int-rotationn/ ext- rotation Horizontal flexion/extension Supination/pronation	Sagital	Throwing Baseball swing Golf swing

Table 1 .Examples of dominant planes, motions and axis in gross movements[8].

Although the single plane scheme is useful when generalising about average motion it cannot be used to describe the functional activities of life and sport such as running, jumping, walking, catching, throwing, twisting, hopping, skipping, rolling, climbing, kicking, pushing, pulling etc. because these activities require simultaneous movement in all planes of motion.

As multi-plane movement dominates activities of life and sport, this makes it essential to track movement in all three planes of motion in order to accurately recognise human actions.

B. Tracking Human actions

New generation controllers use sensors to detect the placement of player's body parts in the real word and provide the programmer with coordinates about the placement of the player in a virtual environment (world coordinates). The most common coordinate system used for representing positions in a virtual environment is the Cartesian coordinate system. Cartesian is a coordinates system that provides a method of rendering graphs and representing the positions of points on a two-dimensional or three-dimensional environment [9].

The most common approach used to recognise human actions is to try and track the placement of the body parts by their position in the virtual world. For example, a pose can be the position of an arm and forearm, and compare it with the current user's arm and forearm position trying to recognise this pose. The main problem with this technique is that the user's body parts must be in a position which exactly matches the coordinates of the stored pose. Even if a variation is allowed in each dimension dx, dy, dz allowing a diversion range $x-dx \leq x \leq x+dx$ in x-axis, $y-dy \leq y \leq y+dy$ in y-axis and $z-dz \leq z \leq z+dz$ in z-axis, which will create a cube of allowed coordinates around the base position (Figure 2), the method can only recognise poses around the initial captured position. For example, it cannot recognise a stretched hand action if the initial captured pose was on the front of the body and the user stretched his hand on the left side. This factor makes this method unsuitable for tracking the tri-planar human actions.

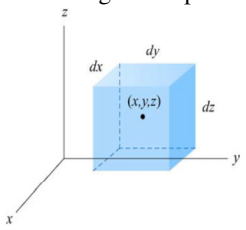

Figure 2. Cube of allowed coordinates around the base position.

The technique we suggest for recognising human actions is to track the relationship between each of the body parts. The relationship between body parts can be measured using the angle between the vectors represented and passing through each part. A vector starts at the joint of the parts and ends at end of the part. For example, a pose that tracks how outstretched a hand is, can be the angle between the two vectors starting from the arm and forearm joint, one end at forearm's end and the other one at arm's end. Furthermore, by allowing a variation ($d\theta$) on the angle (an angle counts as valid if it is inside the range $\theta-d\theta \leq \theta \leq \theta+d\theta$) the flexibility of the recognition can be controlled. This method can recognise poses unaffected by the positioning of the body's parts in the virtual world environment because it only compares the relationship of the parts, (if the positioning of two parts change but they keep the same angle/relation between them they will still be recognised as the previous pose). Each user's pose has a list of the angles, which represent the relationship between user's parts that construct the pose. Finally, a user's action is tracked as a sequence of two poses (start and end) within a certain amount of time.

III. IMPLEMENTATION OF THE RECOGNITION METHOD

The recognition method, we proposed in the previous section, supported the representation of the relation between human

body parts based on the angles between them. The recognition process was performed in three steps:

a) *Calculate the angle between the body parts*
b) *Define the direction of the angle*
c) *Compare the angles of the poses*

A. *Calculate the angle between two body parts*

Firstly, a vector to represent each part was created by subtracting the End Vertex values from the Start Vertex of the part (Vector 1 = Vertex B - Vertex A, Vector 2 = Vertex C - Vertex A). Vector is the movement from one point to the next [10].

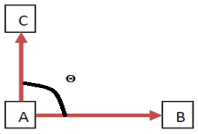

Figure 3. Vector's vertexes

Then the dot was used to determine the cosine of the angle [11]

$$\text{Vector1} \bullet \text{Vector2} = |\text{Vector1}| * |\text{Vector2}| * \text{cosine}(\Theta)$$

=>

$$\text{cosine}(\Theta) = \frac{\text{Vector1} \bullet \text{Vector2}}{|\text{Vector1}| * |\text{Vector2}|}$$

$$\frac{\text{Vector1} \bullet \text{Vector2}}{|\text{Vector1}| * |\text{Vector2}|} = \text{Normalize}(\text{Vector1}) * \text{Normalize}(\text{Vector2})$$

$$\Rightarrow \quad \Theta = \arccos(\text{Normalize}(\text{Vector1}) * \text{Normalize}(\text{Vector2}))$$

The angle calculated with this procedure is unsigned (without direction information). Direction information is important in order to separate two sets of vectors with the same angle between them but with different rotation.

B. *Define the direction of the angle*

The vectors created by the vertexes of the body parts define a surface. The Normal of this surface was used to identify the direction of the angle between the two vectors (Figure 4). The Normal determines a surface's orientation and is a vector that points directly away from and is perpendicular to the surface.

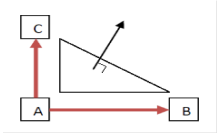

Figure 4. Surface Normal

The Cross Product was used in order to calculate the normal of these vectors. The result of the cross product is a new vector that is positioned perpendicular to the plane generated by the vectors. The cross product is not commutative and the right hand rule was used to determine the direction of the resulting cross product (Figure 5)

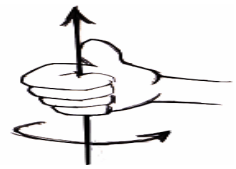

Figure 5. Right hand rule [11]

The next step was to normalise the vector of the normal.

$$Normal = \frac{Normal}{|Normal|}$$

Figure 6. Normalise a vector

C. *Compare the angles of two poses*

The final step of the recognition process was the implementation of a procedure that compares the angle of two body parts and the orientation of the surface defined by these parts.

An angle between two user's body part of the current pose was considered as matched if it was inside a range of values, created using the variation defined by the user when he stored the pose, around the stored pose angle Θstored ± (variation /100) *π/2

Finally, in order to compare the orientation of the two surfaces defined by these parts, the Dot product between the direction (normal) vectors of each surface was used. If the Dot product was positive the surfaces had the same orientation. For angles smaller than the calculated variation the orientation test was ignored in order to prevent failing match tests even if they are in the allowed range of angles.

IV. BE THE CONTROLLER TOOL KIT

We used the method analysed in the previous sections to develop a tool kit that aims to help game designers easily identify and recognise a user's pose and gestures using Microsoft's Kinect motion controller, in order to then link these movements with actions in a game. The project was divided into two parts: A pose and gesture generator application that allows the software developer to capture poses using the Kinect controller and then select which relation between the various human body parts they want to track, via an easy to use graphic interface. A software library that allows

a software developer to load poses and gestures created with the generator application, and inform the application when a recognised pose or gesture is performed by the user.

The actors of the system are: the software developer, the user and the application using the library. Finally, three elements comprising the core system:

- The Pose and Gestures generator - windows application (developed using Windows Presentation Foundation (WPF) system and C# (C Sharp))
- The Recognition library - dll file for windows (Developed using C#)
- The Kinect sensor

The figure 7 shows a high level overview of the tool kit.

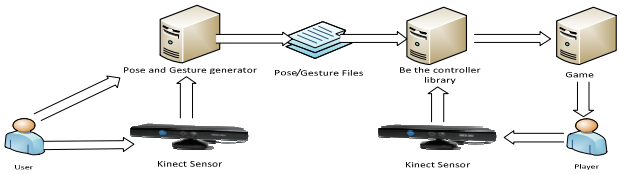

Figure 7. System high level overview

The main functionalities that are supported by the latest version of the tool kit :

Pose and Gesture Generator application

- Create pose: The system allows the user to create a new pose
- Test pose recognition: The system allows the user to test the recognition of an existing pose
- Compare a sequence of poses: The system allows the user to compare a sequence of poses with existing poses (base pose)
- Save pose list: The system allows the user to save the current pose list to a file
- Load pose list: The system allows the user to load a list of poses from a file to the system
- Create gesture: The system allows the user to create a new gestures
- Test gesture recognition: The system allows the user to test the recognition of an existing gesture
- Save gesture list: The system allows the user to save the current gestures list to a file
- Load gesture list: The system allows the user to load a list of gestures from a file to the system
- Accept voice commands (with speech recognition): The system allows the user to control the application using voice commands

'Be the Controller' library

- Initialise Kinect sensor: The system allows the programmer to initialize the Kinect sensor
- Load pose list: The system allows the programmer to load a list of poses from a file to the system
- Load gesture list: The system allows the programmer to load a list of gestures from a file to the system
- Announce the recognition of a pose or gesture: The system informs the programmer when a pose is recognised
- Announce the expiration a pose or gesture: The system informs the programmer when a gesture is recognised

V. PROJECT EVALUATION

A. *"Be The Controller App" application*

For the needs of the evaluation experiment, we developed an application that aimed to provide a way for users to convert the recognised pose and gestures into keyboard and mouse actions. This was implemented by converting the return value of a recognised pose or gesture to a simulation of keyboard key, mouse key press event or to activate the control of the mouse cursor position with the user's left hand. For the simulation of the keyboard/mouse events in this application the SendInput function of the Uesr32.dll library was used. This function provides the closest simulation of keyboard /mouse events, which makes the application compatible with the majority of games. A table with the allowed return values and the action represent each value is available in Appendix I. With the use of this application it was possible for a user with no programing experience to use the system , which aided not only game developers but also people that represent the "typical gamer" to participate in the evaluation test.

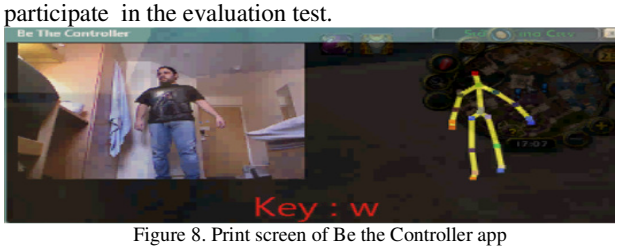

Figure 8. Print screen of Be the Controller app

B. *Evaluation Experiment*

In the phase of the evaluation of the project, we selected ten representative users and we assigned them to complete a set of tasks using the "Be the Controller" project and a well-known multiplayer online game, World of Warcraft (WOW) [12]. Before each user started the evaluation experiment, we asked him to answer a pre- questionnaire in order to identify the user characteristics. During the evaluation, a usability expert observed the users in order to collect as much relevant information as possible from them. Finally, at the end of the evaluation each we asked the user to answer another

questionnaire that aimed to record the way the user saw the project after the trial.

Based on the pre-questionnaire the characteristics of the ten users taking part in the experiment are that: all the users had previous video game experience, 60% of the users had played games on game consoles. Furthermore, 40% of the users had previous experience with motion tracking controllers but only 10% of them had played games with the Kinect controller. Finally, 70% of the users had some previous experience with World of Warcraft .

The data from the observation of the users during the trail helped to form some important conclusions about the use of Kinect in games and the usability of the project. First, it was clear that the use of Kinect offers more enjoyment and the ability for a player to become more immersed in the game on many different levels. This was decreased somewhat by the fact that the game used for the test was not designed to be played with Kinect. As a result a delay to the activation of an action in the game occurred from the moment the corresponding gesture was performed by the user (global delay = gesture perform time + game's action cast time).

Users found it easy to create their own poses and gestures and they were enthusiastic with the fact that they were able to bind their action to key/mouse events. In some cases, people were so excited with the experiment that they spent a lot of time creating poses and gestures that were very similar to the avatar's animation in the game.

Finally, users were satisfied with the success rate of recognitions of the project, but some did not like the limitations of the Kinect controller in the creation of some poses. For example, they were unable to capture poses in which one body part was in front of other body part. Moreover, it was clear that the Kinect is not suitable in controlling the mouse cursor because of the instability of the hand position point tracked by the Kinect SDK, which made it hard for the user to aim with the mouse cursor. A possible solution to this issue is instead of using the point of the hand directly to determine the cursors position on the screen to use hand's speed and direction of move to determine the next position of the cursor. This implementation is more accurate as the small and short movements of the hand's point due to the instability of its tracked position will produce insignificant movements of the cursor.

As mentioned before the users were asked to answer a post-questionnaire and the following table shows the average results for the answers given by the ten participants in the project. The following table shows the outcome of the post-questionnaire.

	Question	Answer (+2/-2)	Average
1	Your actions were always correctly recognized by the application	Strongly agree / Strongly disagree	1.8
2	You found it easy to create your own pose and gestures.	Strongly agree / Strongly disagree	1.7
3	The use of Kinect helped you to become more immersed in the game.	Strongly agree / Strongly disagree	1.5
4	How much effort was needed in order to learn how to use the Pose and Gesture generator application?	A lot of effort / No effort	-1.6
5	If you were a game developer creating a new MMO game, would you choose Kinect over traditional game controllers (joysticks, game pads etc.) as the primary controller of your game?	Yes / No	1.6
6	The voice commands were always executed correctly.	Strongly agree / Strongly disagree	1.1
7	Did the "Be the controller app" always matched correctly your action to the appropriate key/mouse command?	Always / Never	1.7
8	The procedure of creating pose and gesture was easy to understand.	Strongly agree / Strongly disagree	1.8
9	Kinect was able to "see" all the parts of your body all the time.	Strongly agree / Strongly disagree	0.9
10	Are you willing to spend £129 to buy Kinect controller in order to play games with motion tracking?	Yes / No	1.3

Table 2. Result of the post-questionnaire

The results of this post-questionnaire verified the inferences based on the observation of the users.

VI. DISCUSSION

A. *Evaluation Results*

The usability experiment helped to validate the conclusion that the use of motion tracking controllers in games is likely to be a key growth market. This can be inferred from the fact that Kinect holds the Guinness World Record for the "fastest selling consumer electronics device". People were excited by the idea of controlling a game without holding a controller, and with the new direct manipulation interfaces. The fact that led to this conclusion was that during the experiment, users were becoming immersed in the game, they seemed to enjoy it greatly and they spent much more time than the tasks of the experiment required, creating their own poses and gestures and performing more complicated actions in the game. Unfortunately, the overall immersion was reduced because the game used in the test was not designed to be played with Kinect, and this had as a result, a delay to the activation of an action in the game from the moment the corresponding gesture was performed by the user (global delay = gesture perform time + game's action cast time).

The poses and gestures generator was easy to use in order to create a desired pose/gesture. Users found it simple to create their own poses and gestures and they were enthusiastic with the fact that they were able to bind their action to key/mouse events. In some cases, people were so excited with the experiment that they spent a lot of time creating poses and gestures that were very close to the avatar's animation in the game.

Finally, confirmation that the chosen recognition method was appropriate for the recognition of human action was the fact that users were satisfied with the success rate of recognitions of the project. However, in some cases users identified limitations of the Kinect controller in the creation of some poses. For example, they were unable to capture poses in which one body part was in front of other body part. Moreover, it was clear that the Kinect is not suitable to control the mouse cursor because of the instability of the hand position point tracked by the Kinect SDK, which made it hard for the user to aim with the mouse cursor.

All the above conclusions prove that most of the project's aims were achieved, such as: Project Stability, Easy to use ,Good recognition rate, Compatible with game development tools like XNA, Best Performance

B. Potential Applications for the Research

This section proposes some potential uses of the project. Initially, two proposals are described and explained how to use this library to control contemporary games and finally a proposal to use this library for developing new generation games that are designed to use Kinect is introduced.

Use of the "Be the controller project" to control contemporary game

This proposal is aimed at developers of contemporary game and suggests a way to control games by binding the user's motion to the simulation of keyboard/mouse events. This can be accomplished with the use of the "Be the Controller Application", which converts the return value of a recognised pose or gesture to a simulation of keyboard, mouse key press event or activates the control of mouse cursor position with the user's hands (left or right). This proposal does not require any programing experience so it can be used by almost anyone. Furthermore, for the simulation of the keyboard/mouse events in the "Be the Controller App" the SendInput function of the Uesr32.dll library was used. This function provides the closest simulation of keyboard /mouse events, which makes the application compatible with the majority of games.

The main drawback of this method is that contemporary games are not designed to be played with Kinect, and have as a result, a delay to the activation of an action in the game from the moment the corresponding gesture is performed by the user. This fact can reduce the immersion of the user with the game.

Use of the Be the controller project to control contemporary game – Developer site

This proposal is aimed at developers of contemporary game and suggests a way to use Kinect and "Be the controller" library as the main input device of the game. In most of the games when a keyboard and mouse event occurs, the developer activates, through the game code, the appropriate action and the corresponding animation in the game [13]. Using the same scheme, the "Be the controller" library can play the role of a keyboard and mouse and be used to control the game. When a pose or gesture recognition event occurs the developer can activate the appropriate action and the corresponding animation in the game.

The main disadvantage of this implementation is that when a gesture is used to activate an action in a game, it is not possible to identify which animation to play during the time the user performs the gesture. A possible solution to this problem is to divide each animation into three parts, a start, middle, and end, and create a representative pose for each part. The start pose would represent the beginning of the animation and probably more than one animation would start with this pose. This makes it suitable to be used to group the animations into groups with the same start pose. The middle pose in most cases is unique and so can be used as an identifier for the animation because it is created based on the characteristic of the avatar's bones at the middle of the animation's duration. In most cases at this point of time, the animation is different from other animations with the same start. Finally, the end pose would determine the end of the animation and activate the appropriate action.

These three poses could be used to solve the problem of choosing the animation to play during the time the user performs a gesture as follows. When the start pose of a group of animation is detected, a generic animation that represents this group should be played until the middle pose of an animation is recognised, which will then define which animation matches the user's action. Furthermore, the middle pose should be used as the start pose of the gesture represented in this animation/action and the end pose should be the end of this gesture. When this gesture is recognised, the corresponding action to this user motion should be activated in game. This solution theoretically solves the problem of choosing the appropriate animation during a user's motion in most cases.

Use of the Be the controller project to control a game designed to use Kinect

A game designed to use Kinect sensor as its primary input creates the user's avatar animation in real time using the data received from the Kinect about the position of the user's body parts. The "Be the controller" library can be used in this scheme to convert the user's body to the main input of the game. When the library announces the recognition of a gesture

or a pose the developer can activate the appropriate action and any needed animation in game. In this proposal, there is no need to play any stored animation during the time the user performs a gesture because the user's avatar animation is created in real time relative to the user's body parts position. The main drawbacks of this real time animations scheme is that inaccuracies in the controller tracking may cause odd movements and that the real time animation will represent player's actions in game less gracefully compare to a custom design animation.

A more optimal and appropriate scheme for managing game animations is to combine stored animation with real time created animation using the data received from the Kinect sensor. The developer will use the "Be the controller" library to recognise the player's actions and will try to match the particular action with a stored animation analogous to matching sounds to known words in speech recognition software. If the library fails to recognise a specific motion or more than one animation matches the recognised action resulting in uncertainty as to which stored motion animation to activate, a real time created animation will be played. This scheme allows the combination of stored motion animation with real time created animation in a game and, as a result the visually effective and attractive design stored animation will increase the enjoyment of the game, while the real time animation will address any voids created by unrecognised motion and will make the gameplay feel more realistic.

VII. CONCLUSIONS

This project has attempted to develop a tool kit that can be used by an application developer to recognize a user's motion using the Microsoft Kinect controller, as a means for the user to interact with a virtual dynamic world. By applying a theory-based academic approach it was possible to maximise the productivity of the development process, and with the use of software engineering principles and design patterns, good software has been developed [14]. The study of research literature concerning human body movements supported the design and development of a human motion recognition method that gave a high success rate, which was one of the most important aims of this thesis. Moreover, the outcomes of the usability experiment showed that the use of user-centric design principles resulted in an easy-to-use tool kit.

Through the design and development process of this project, it became obvious that the Kinect controller is a powerful device with many capabilities and potential uses in the gaming world. Moreover, it was clear that the use of Kinect offers increased levels of enjoyment and the ability for a player to become more immersed in the game on many different levels. However, issues arise concerning the capabilities and the limitations of the Kinect, as a piece of hardware, which may ultimately decrease the immersion, and enjoyment of a game.

In conclusion, motion controls have a promising future in the game industry. In most game genres, which have good game and mechanism design, motion control could become the default input method of the game. However, more research is needed in order to make this kind of control a suitable input method for games with high complexity such as Massive Multiplayer Online games in which players have many spells and abilities and where fast, accurate responses are vital.

REFERENCES

[1] Miller, F, Vandome, A & McBrewster , J 2010, Game Controller , Alphascript Publishing, Mauritius.

[2] Guinness World Records, 2011, Guinness World Records, Kinect Confirmed As Fastest-Selling Consumer Electronics Device, viewed 10 August 2011 <http://community.guinnessworldrecords.com/_Kinect-Confirmed-As-Fastest-Selling-Consumer-Electronics-Device/blog/3376939/7691.html>.

[3] Adams, E & Rollings, A 2007, Game Design and Development: Fundamentals of Game Design, Pearson Education, Inc., Upper Saddle River, New Jersey

[4] Gamasutra 2011, Gamasutra, viewed 2 September 2011 <http://www.gamasutra.com/view/news/36985/Activision_Blizzard_Reports_Digital_Sales_Growth_18M_Black_Ops_Map_Pack_Sales.php>

[5] Carmody, T 2010, Wired, How Motion Detection Works in Xbox Kinect, viewed 2 August 2011 <http://www.wired.com/gadgetlab/2010/11/tonights-release-xbox-kinect-how-does-it-work/>.

[6] Hadjiminas N, 2011, Be The Controller: A Kinect Tool Kit for Video Game Control, City Univercity, London, United Kingtom

[7] Bartlett, R 2007, Introduction to Sports Biomechanics, 2nd edition, Routledge , London, United Kingdom.

[8] McGinnis, PM 1999, Biomechanics of Sport and Exercise, Champaign, Illinois

[9] Gregory, J, Lander , J & Whiting M 2009, Game Engine Architecture, A K Peters/CRC Press, Natick, Massachusetts

[10] Dunn F, Parberry, I 2002, 3D Math Primer for Graphics and Game Development, Wordware Publishing Inc, Plano, Texas

[11] Millington, I 2010, Game Physics Engine Development: How to Build a Robust Commercial-Grade Physics Engine for your Game, 2nd Edition, Morgan Kaufmann,Waltham, Massachusetts.

[12] Blizzard 2001, World of Warcraft, Blizzard Entertainment, Inc., Irvine, California.

[13] Rouse, R 2005, Game Design: Theory & Practice, 2nd edition., Wordware Publishing, US.

[14] Weilkiens, T 2009, Systems Engineering with SysML/UML: Modeling, Analysis, Design , Morgan Kaufmann, Waltham, Massachusetts.