# A NATURAL USER INTERFACE FOR VIRTUAL OBJECT MODELING FOR IMMERSIVE GAMING

by

**Siyuan Xu**

A Thesis

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Interactive Media & Game Development

July 2013

APPROVED:

______________________________________

Dr. Robert W. Lindeman, Major Advisor

# ABSTRACT

We designed an interactive 3D user interface system to perform object modeling in virtual environments. Expanding on existing 3D user interface techniques, we integrate low-cost human gesture recognition that endows the user with powerful abilities to perform complex virtual object modeling tasks in an immersive game setting.

Much research has been done to explore the possibilities of developing biosensors for Virtual Reality (VR) use.  In the game industry, even though full body interaction techniques are involved in modern game consoles, most of the utilizations, in terms of game control, are still simple. In this project, we extended the use of motion tracking and gesture recognition techniques to create a new 3DUI system to support immersive gaming. We set a goal for the usability, which is virtual object modeling, and finally developed a game application to test its performance.

# Table of Contents

## List of Figures

## List of Tables

# 1. INTRODUCTION

The purpose of this project is to design a 3D user interface for virtual object modeling. Expanding on existing 3D user interface techniques, we developed a low-cost human body gesture/posture recognition system and let the user perform virtual object modeling tasks in an immersive game setting.

Inspiration for this work comes from a physics-based game, called “Wake up the Box 4” [1]. In this game, the player needs to draw shapes and let gravity take its course in order to hit and wake up each level’s box. It is a puzzle game in which everything happens in a 2D plane. To provide a novel game experience, in this project we brought the idea into the 3D world: Players can use their hands to create custom 3D objects to solve the puzzles. (Figure 1)

We solved several problems in order to translate this idea from 2D to 3D. First of all, we built up a virtual world and allowed users to create virtual objects in this environment. Secondly, we developed a motion based user interface to let the user use their hands or body to interact with the world. Finally, we developed a system that gives users the ability to modify and manipulate these objects to solve the puzzles provided on each game level.

In this project, we extended the use of motion tracking, gesture recognition, and immersive object modeling to create more compelling and entertaining game contexts. Knowledge of motion recognition and immersive modeling was used. A 3D physics-based puzzle game was developed to evaluate this novel immersive 3DUI system.

This paper is organized as follows. Chapter 2 explains the research problem and gives the related work in the respective field. Chapter 3 gives an overview of the system architecture. In Chapter 4, we elaborate the proposed 3DUI system in details. In Chapter 5, we discuss a user study we conduct for the purpose of evaluation. Finally, in Chapter 6, we give a conclusion of all our effort and point out the future direction of the research.

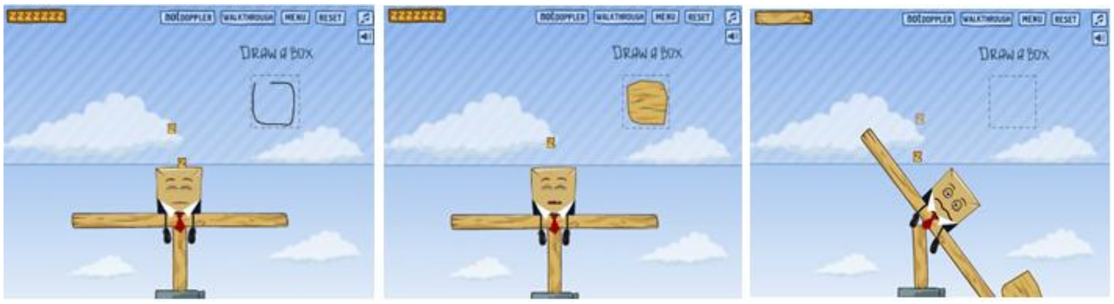


**Figure 1: Disturb the box's sleep by drawing objects inside the striped areas [1]**

## 2. PROBLEM STATEMENT

During the progress of building an immersive 3D object modeling application, we met with three problems. First of all, we needed to build a fully functional object modeling system. Secondly, we needed to develop a motion-based input system to support complex modeling operations. And finally, we needed a well-designed user interface to connect the two components together. In the game industry, even though full body interaction techniques are involved in modern game consoles, most of the utilizations, in terms of game control, are still simple. Most controlling systems rely on tracking hands or bodies positions rather than understanding human hands and bodies gestures. This places significant restrictions on the game applications which involve these techniques. While object modeling is beyond the scope of our research, our research interests lie in: (1) How to design a practical human body gesture/posture recognition system, and (2) How to design a competent user interface to let the user perform virtual object modeling tasks in a comfortable and interactive way.

### 2.1 PREVIOUS STUDY

Detecting human motion is a challenging problem due to variations in pose, lighting conditions and complexity of the backgrounds. In the past few years, many image-based methods have been proposed [2, 3, 4]. Some methods involve statistical training based on local features, e.g., gradient-based features such as HOG [2] and SIFT [5]. STIP [6] and MoSIFT [7] are another two popular video representations for motion recognition. Although many of these image–based methods have very high recognition rates, they have high computation and storage requirements and therefore cannot be achieved in an interactive way. In November 2010 Microsoft launched the

Kinect, which led to renewed research on depth-based motion recognition. With the Kinect, the depth data is utilized to constitute a skeleton model which consists of twenty joints all through the human body. Raptis et al. [8] presented a real-time gesture classification system to recognize dance gestures. The method uses 16 main skeleton joints and takes approximately four seconds for data collection. Dan Xu et al. [9] presented a natural human-robot interaction system based on dynamic hand gesture recognition. In their method, a start/end point detection method is proposed for extracting hand gesture from the hand trajectory. Wang et al. [10] presented a Hidden Markov Model (HMM) based dynamic hand gesture algorithm using Kinect. With a palm node, defined by the Kinect, valid points are extracted and analyzed to accurately identify defined gestures and reject non-defined gestures. Gu et al. [11] implemented a non-intrusive human gesture recognition system that was able to recognize gestures if they were performed faster or slower (within certain ranges) compared to the training data.

In the virtual object modeling area, Clark [12] was the first to use a head-mounted display (HMD) system to achieve immersive modeling. Before his work, HMDs were only used for exploring virtual worlds. His system allowed users to create parametric surfaces by manipulating control points on a wire-frame grid. The use of HMDs was mainly to improve interaction with models. Butterworth and Davidson [13] developed their “3dm” system which drew techniques of model manipulation from both CAD and drawing programs. It supported users’ natural forms of interaction with objects to give them better understanding of the models. Matsumiya and Takemura [14] developed a new free-form interactive modeling technique based on the metaphor of clay work. In this system, users could interactively design 3D solid objects with curved surfaces with their hands and fingers. *Denting* and *pinching* were the two main deformations that the system supported. The major merit of this work was that the manipulation method was direct. Kuester and Duchaineau [15] created a semi-immersive virtual environment for two-handed modeling, sculpting, and analysis tasks. Scene navigation and virtual menus were implemented to make more complicated modeling tasks possible.

In recent years, motion capture based techniques have been used for more complex modeling tasks. Leithinger et al. [16] designed a novel shape output system. The shapes that a user could create were much less restricted than in previous work. Direct touch interaction with 2.5D shape displays (2D + height) in real space was extended by a set of free-hand gestures. The system was able to form a 2.5D approximation of an object's shape, similar to a relief sculpture. (Figure 2: 2.5D shape display with gestural interaction) Tasks that could be implemented by gesture input included: *selection*, *translation, rotation*, and *scaling*. However, these conductions are still limited in 2D surface.

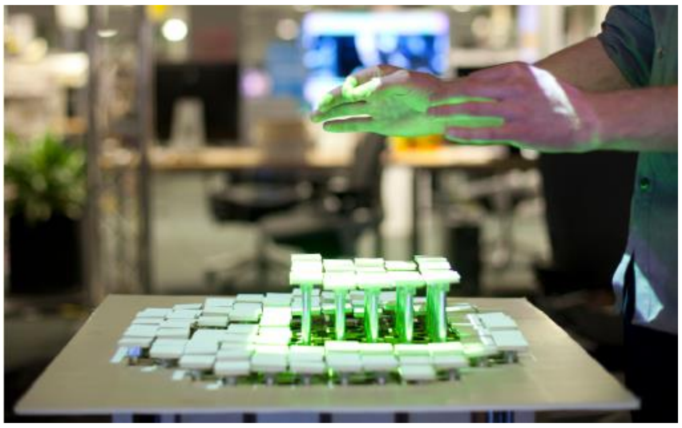


**Figure 2: 2.5D shape display with gestural interaction [16]**

## 2.2 OUR WORK

In computing, a Natural User Interface (NUI) is the common parlance to refer to a user interface that is based on natural elements [17]. The user uses simple voice or body gestures rather than artificial control devices to make the operation more natural. While simple immersive modeling is not a difficult task, how to perform such tasks naturally is what concerns us here. Many previous methods had different kinds of limitations to the user. For instance, some methods used too much computation time and the user had to wait for its response. Some methods required specific devices which are not common. Some methods would come to different results when the user performed the same activity at a different speed.

In this project, we made use of the Kinect camera and built a 3D NUI to support natural body gesture input. The 3D NUI contains a low computation cost human body gesture/posture recognition system that could run in an interactive way. Most importantly, our system gives the user flexibility when giving commands. The user is not limited to a fixed position and does not need to worry about giving wrong commands due to unintentional gestures. We developed a game which lets the user perform virtual object tasks in an entertaining way. We also ran a user study to evaluate the user experience when using our 3D user interface.

## 3. SYSTEM ARCHITECHTURE

A brief architecture for the system is as follows (Figure 3). A depth sensor (we used the Kinect in our system) is used to capture a user's gesture input (wave, push forward, etc.). The data is then delivered to a recognition system on a workstation that can detect the gesture types, and then translate the data into digital commands. These commands are sent to the game engine where corresponding activities are performed. Proper output devices connected to the workstation display the results.

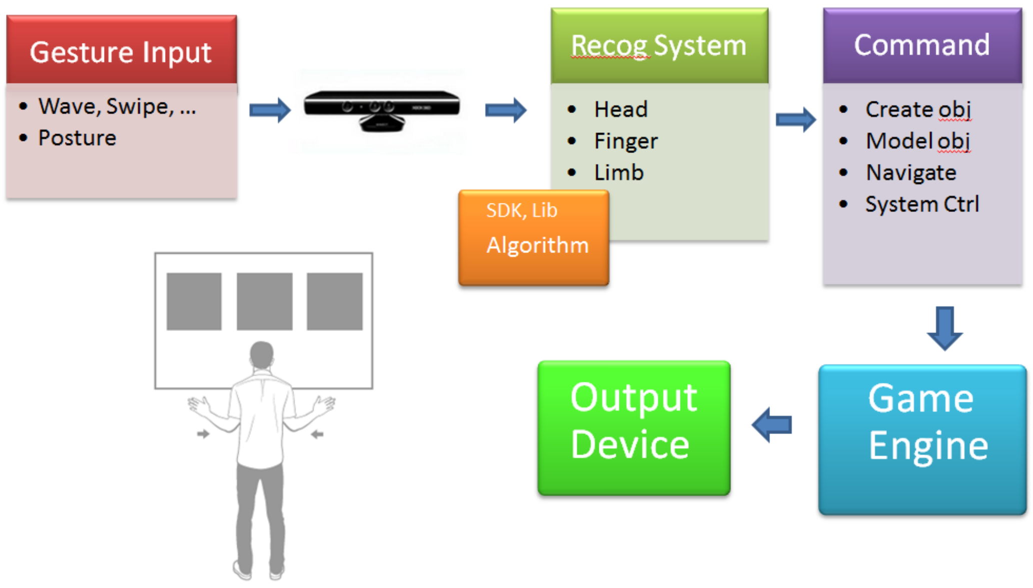


**Figure 3: Architecture of the Natural Interaction System**

### 3.1 3DUI DESIGN

In order to realize the proposed architecture, our method focused on solving the following three problems: (1) Finding a pose descriptor that concisely represents a human body motion; (2) Designing a proper gesture recognition system that robustly identifies gesture input commands in an interactive way; (3) Building an immersive gaming environment which makes full use of the gesture recognition system and allows the user to perform complex virtual tasks efficiently.

### 3.1.1 KINECT SENSOR AND SKELETAL MODEL

The Kinect (Figure 4) is innovative game controller technology introduced by Microsoft in November 2010. The Kinect depth sensor consists of a depth camera and an RGB camera which give an RGB image as well as depth at each pixel. Early in 2012, Microsoft shipped the commercial version of the SDK which contains the NUI application programming interface (API) and the Microsoft Kinect drivers to integrate the Kinect sensor within Microsoft Windows. The version we used is Kinect SDK 1.5 which was released in June 2012.

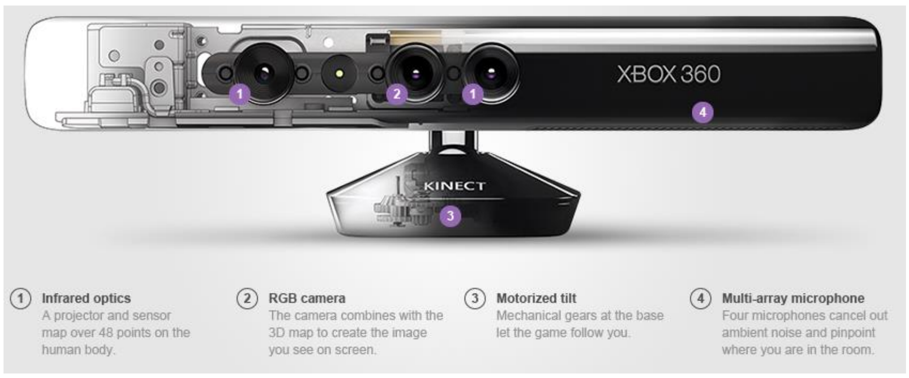

XBOX 360
KINECT
1
2
1
4
3
① Infrared optics
A projector and sensor map over 48 points on the human body.
② RGB camera
The camera combines with the 3D map to create the image you see on screen.
③ Motorized tilt
Mechanical gears at the base let the game follow you.
④ Multi-array microphone
Four microphones cancel out ambient noise and pinpoint where you are in the room.

**Figure 4: Microsoft XBox Kinect Sensor [18]**

Generally, the SDK provides two models of analyzing human motion: Image-based modeling and Depth-based modeling. Image-based modeling relies on the visual image of the human body. By analyzing the image in each frame, the human body’s posture is identified. This process usually requires various kinds of digital image processing knowledge. The depth-based model makes full use of the depth stream and eventually builds up a skeleton model of the human body in front of the sensor. Normally it requires significant processing work using image-based models. Although we can get accurate results to reconstruct a human body in detail after massive calculation, it is not efficient. For our project, those results are not necessary. In our gaming environment, we only need to care about the activities of the key points of the user, their head and hand gestures, for instance. Therefore, joint tracking becomes a great choice in our interaction application. The depth-based model, which consists of 20 joint positions, is sufficient and is employed to represent the human (Figure 5) [19].

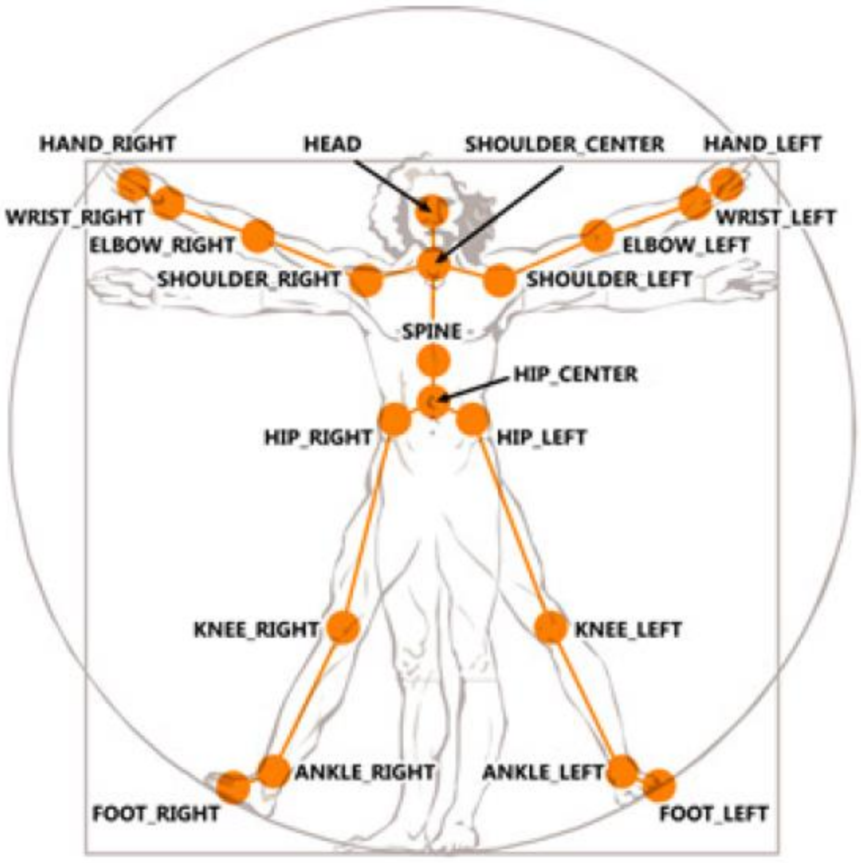


**Figure 5: Human Skeleton Model**

### 3.1.2 INTERACTIVE GESTURE RECOGNITION SYSTEM

With the development of inexpensive motion capture based input devices, such as Nintendo Wii Remote and the Kinect, video games have already come to the new generation. However, people still cannot enjoy a virtual reality experience due to technical limitations. For instance, the usage of these modern game consoles, in terms of game control, is still simplistic. In order to achieve virtual reality interaction, we need to extend the existing 3D user interface techniques. We need a system which can enable the user to have more choices to perform much more complex actions.

Time is another issue in this system. The gaming world is a time-sensitive environment. Any time delay would significantly decrease the user’s immersive feeling. Therefore, being interactive is one of the requirements of our project. We can't accept any noticeable latency to guarantee the natural interaction between the user and the system. We will have a more detailed discussion of the implementation of our interactive gesture recognition in Chapter 4.

### 3.1.3 APPLICATION IMPLEMENTATION

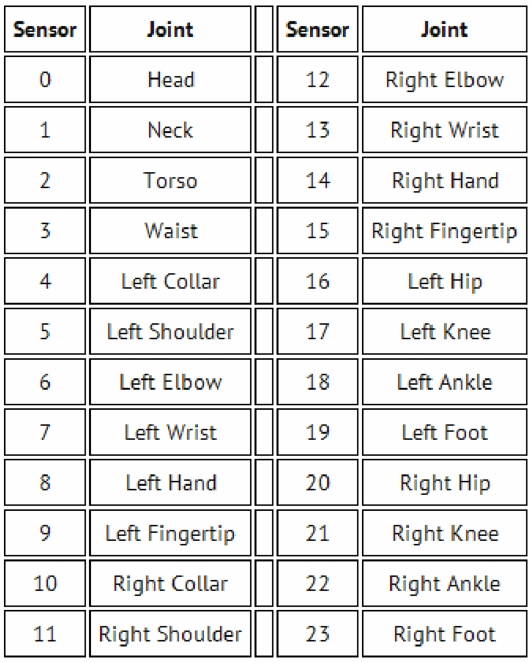
We used the Unity Game Engine as the development platform and built a 3D virtual environment. Unity supports flexibility for virtual object editing. Most importantly, developers can directly use c# in Unity. This feature significantly enhanced the compatibility when importing the recognition system to our game development environment. We developed a virtual object modeling system in Unity. During the game play, the user's gesture is translated into game commands and applied to the target object.

## 3.2 CONNECTING EVERYTHING TOGETHER

Our work also involves many other components, which need to talk to each other. Starting from capturing raw data from the Kinect sensor, we need to consider the communication between the following pairs: Kinect sensor and gesture recognition system; gesture recognition system and game engine.

In order to make it easier to use the gesture recognition system in the game engine, we used c# to develop our gesture recognition system and compile the codes into libraries. We then put these libraries into the Unity environment, so that we can call defined functions during game play.

Communication between the Kinect and the gesture recognition system is more complicated. A system called VRPN [20] is introduced here. VRPN, which stands for Virtual-Reality Peripheral Network, is a set of classes within a library and a set of servers that are designed to implement a network-transparent interface between application programs and the set of physical devices (trackers, etc.) used in a VR system. VRPN provides connections between the application and all of the devices. Therefore, VRPN can be treated as a middle-layer which provides some sort of standard.



| Sensor | Joint | Sensor | Joint |
|---|---|---|---|
| 0 | Head | 12 | Right Elbow |
| 1 | Neck | 13 | Right Wrist |
| 2 | Torso | 14 | Right Hand |
| 3 | Waist | 15 | Right Fingertip |
| 4 | Left Collar | 16 | Left Hip |
| 5 | Left Shoulder | 17 | Left Knee |
| 6 | Left Elbow | 18 | Left Ankle |
| 7 | Left Wrist | 19 | Left Foot |
| 8 | Left Hand | 20 | Right Hip |
| 9 | Left Fingertip | 21 | Right Knee |
| 10 | Right Collar | 22 | Right Ankle |
| 11 | Right Shoulder | 23 | Right Foot |

**Figure 6: FAAST's Joint's Sensor**

For Kinect devices, USC's FAAST (Flexible Action and Articulated Skeleton Toolkit) [21] includes a VRPN server to stream user skeletons over a network, allowing VR applications to read the skeletal joints as trackers using any VRPN client. A total of 24 skeleton joint transformations are streamed as sensors (Figure 6). It supports Microsoft Kinect for Window SDK for Windows *64 (64-bit), which fulfills the OS requirement in our work. At last, we brought in another a middle-ware called UIVA (Unity Indie VRPN Adapter), which is designed to adapt VRPN to the Indie version of the Unity game engine [22]. We used UIVA in our project to exchange data between FAAST's VRPN server and our recognition system. At this point, we have generated a more detailed system architecture overview (Figure 7).

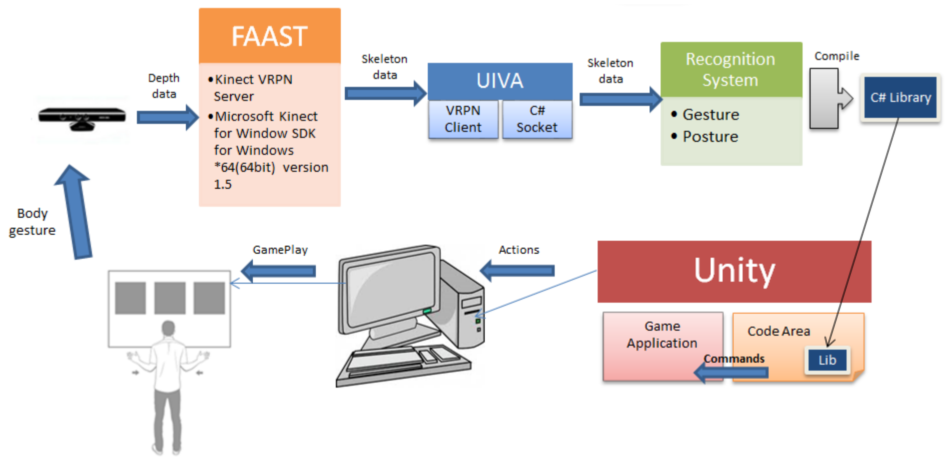


**Figure 7: A More Detailed Architecture of the System**

# 4. METHODOLOGY

## 4.1 TM-SIFT GESTURE RECOGNITION SYSTEM

Human body gesture recognition from video has been studied for many years. While the recognition rate of many good algorithms can reach nearly 100%, they tend to be complex and computationally demanding. Therefore they are not well suited for interactive applications. In addition, due to the various limitations of the devices and algorithms, most of development remains in the experimental stage. In our project—modeling interactive virtual objects for gaming applications—we faced specific challenges when designing our recognition system. Compared to other applications, object modeling requires lower latency in operation. Even slight latency will result in increased inconsistency between the virtual and real contexts. Therefore, we developed a low-cost but satisfactory interactive gesture recognition system that fulfills our goals.

Scale-Invariant Feature Transform (SIFT) is an algorithm in computer vision to detect and describe local features in images [23]. SIFT detects many interest points (key points) in a 2D image and descriptors of these points are used to match static objects (Figure 8). SIFT only works on vectors that are supposed to be distinctive and invariant to any scaling, rotation, or translation. A core step in SIFT is to calculate a gradient magnitude and orientation near the key points and create a gradient histogram. This Histogram of Gradient (HOG) is used to generate the feature which represents the area information near a key point (Figure 9) [23].

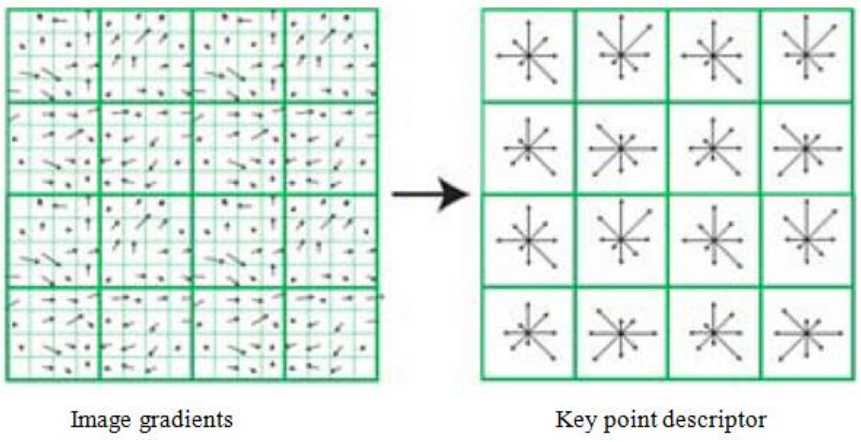


**Figure 8: Image Gradients and Key Point Descriptor**

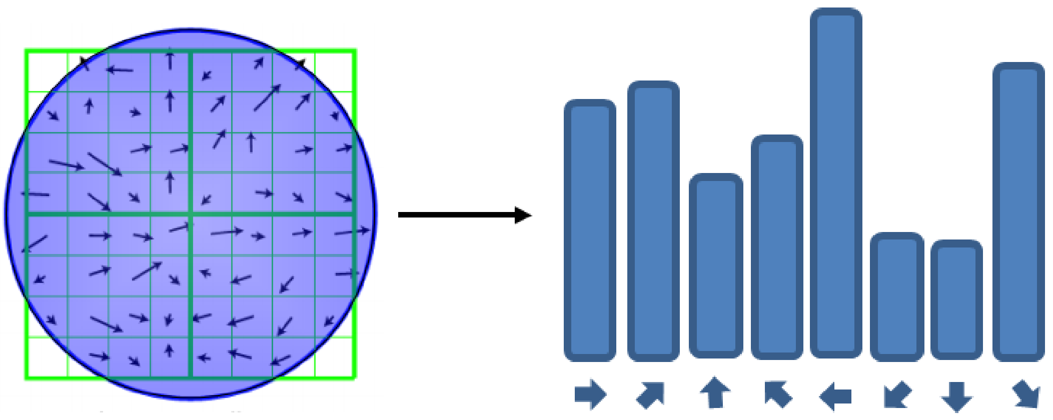


**Figure 9: SIFT and Histogram of gradient (HOG)**

SIFT is designed to detect distinctive interest points in a still image. For motion detection, Histogram of Optical Flow (HOF) [24], similar to HOG, is used to extract motion features from video streams. The idea is to use the velocity vector of the key point between adjacent frames (temporal differences) to describe its motion state. However, if the time used to finish the motion varies, these motions would be treated as different. It is similar to other template-based methods, which may require the same speed to perform a gesture [25].

Inspired by HOF, we developed a new method using a similar idea but bringing in new recognition criteria: Trajectory Matching. We call this method TBTM-SIFT (Template Based Trajectory Matching Scale-Invariant Feature Transform) recognition. Under our new criteria of recognition, the speed of finishing the motion or the velocity during the motion does not matter. The user would have no time pressure on performing activities. Moreover, the user could accelerate his gestures once he gets used to the system.

## 4.2 ALGORITHM DESIGN

An advantage of skeleton model is that we could use joint points to represent corresponding parts of human body. This method largely decreases the complexity of calculation. We create 20 ArrayList objects to store the last n positions of each joint of the human body. Once we want to know the behavior of the certain joint, we only need to deal with the ArrayList for the corresponding joint. For each slot in the array, we record the position and time information. Therefore, the movement of each joint is divided into n-1 segments. Each segment contains two points, $p_t$ $(x_t, y_t, z_t)$ and $p_{t-1}(x_{t-1},$

$y_{t-1}$, $z_{t-1}$), and is represented by a vector $v_t$ ($x_t$ - $x_{t-1}$, $y_t$ - $y_{t-1}$, $z_t$ - $z_{t-1}$). We define m vectors as standard direction vectors (Figure 10). In our work, we defined 8 standard direction vectors plus two depth direction vectors (forward and backward), as shown in Figure 11.

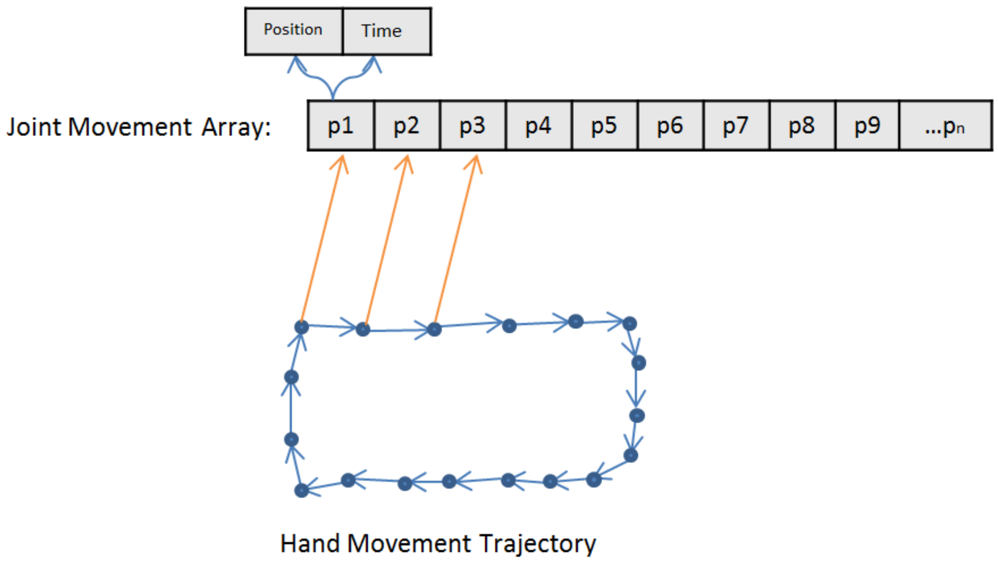


**Figure 10: Data structure to record joint motion information**

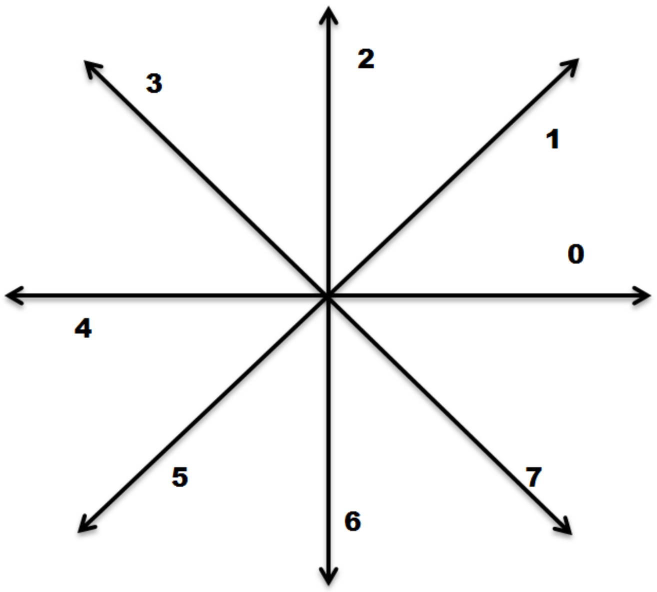


**Figure 11: Standard Direction Vectors**

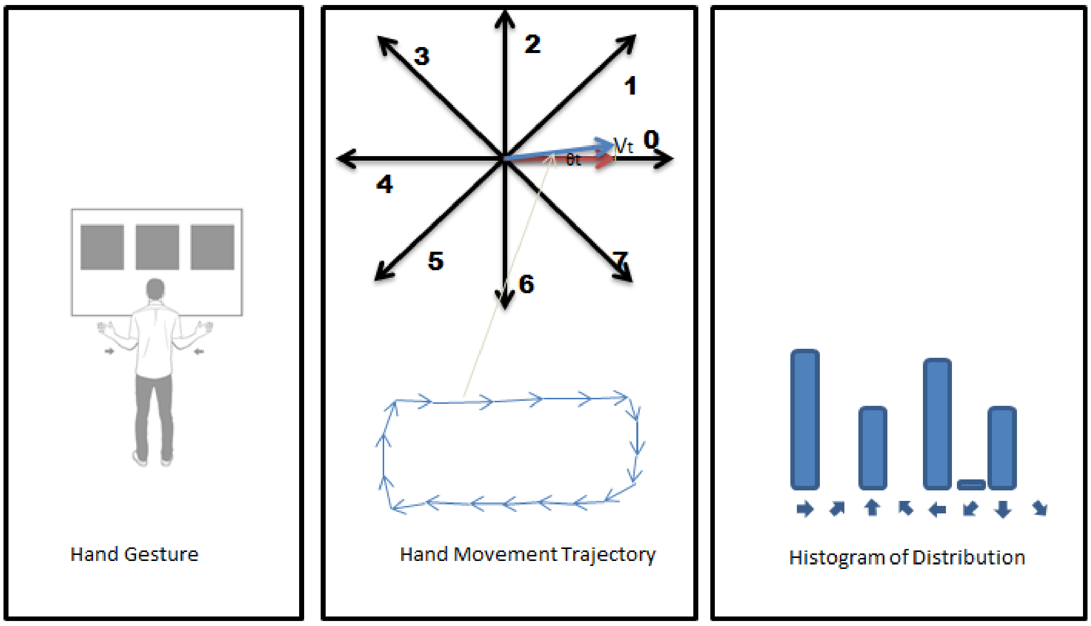


**Figure 12: Extract Motion Trajectory Features to Form Histogram of Distribution (HOD)**

Then we find the closest direction vector of each segment vector $v_t$ and calculate the projection along that direction vector. All the projections are accumulated as energy to form a histogram which indicates the distribution of these standard directions. We call this histogram as Histogram of Distribution (HOD). At this point, joint motion description is replaced by HOD, and is used for all the following processing (Figure 12).

## 4.3 DATA PRE-PROCESSING

Noise interference is unavoidable, and data from the Kinect is not yet completely reliable. A "shake" phenomenon may happen when limbs are close to the body. Also, because of the characteristics of the depth-based data, errors in recognition are more likely to happen when the user's hands are right in front his body. These influences need to be minimized, and our method addresses this problem. Before a position is entered into the array, we detect if it moves "too much" based on its previous position. If the displacement exceeds the threshold, we would treat it as bad data and simply throw it away. When the next data comes, the threshold will be extended because the time span has been increased. The same evaluation would be performed to make sure that all the data stored is valid. It needs to be noted that occasional bad data would hardly affect the result, because within a short time interval, the new displacement vector $v_t'$ $(x_t - x_{t-2}, y_t - y_{t-2}, z_t - z_{t-2})$ would indicate the same and correct energy contribution for HOD. While “too much” is bad, “too little” is also treated as not good. Since our criteria is trajectory matching, time cost should not be taken into account. If the user’s motion pauses for a while at some point, the array should not be updated, since the “new” data would contribute nothing to trajectory matching. We simply throw them away. However, we still need to update the time data for the previous slot so that when the next useful data comes, it would be recorded correctly under our first rule.

## 4.4 VALID DATA COLLECTION

A challenge in motion recognition is how to define valid motion segments. In some situations, it is relatively easy to solve this problem. Cited as an example, Samsung’s new smart phone Galaxy 4 [26] allows the user to control the cell phone with a wave of his hand over the sensor. Whether the user’s hand is in or out of the view of the sensor is an important signal that is used to detect gesture division [26]. However, in our environment, the user will always be in the Kinect camera’s view area. That means we need to rely on some methods to detect a valid motion segment. An easy solution is to ask the user to perform a specific gesture to tell the system when to start and end the gesture input [27, 28]. However, this may add complexity to the method. More importantly, extra time is added to the whole interaction progress, which will decrease the user experience. Although such extra signal may only need 0.5 ~ 1 second, under frequent interaction, such cost is not acceptable.

A very important advantage of our method is that we need no such start and end points to collect valid motion data for recognition. Under the criteria of “trajectory matching”, our method continuously detects if the total length of movement reaches the threshold. Once “enough” movement is collected, a recognition method would be applied to the captured trajectory. If a valid gesture is detected, the array would be cleaned for the coming data. Otherwise, the array will simply update its data.

## 4.5 RECOGNITION PROGRESS

Once enough movement is captured, the corresponding HOD array is calculated. We then interrogate the HOD to analyze the motion. For instance, if the energy in a single direction exceeds 80%, we would treat it as a single direction movement. If the energy falling into the direction of “right”, “down”, “left” and “up” is around 25% and the total deviation is less than 20%, we would treat it as a “square”.

However, there is a problem. Consider the situation where the user is gesturing a standard "square" verses doing a half size "square" twice. The result would be the same. In order to decrease the rate of error recognition, we extracted another feature to increase the robustness of the algorithm. We found that the big difference between the two trajectories is how many sharp changes (more than 60 degrees) of the move direction occur, as happens around the corners of the square. In this example, there are four sharp changes in the standard square and eight in the half size square. We use this value to distinguish different trajectories.

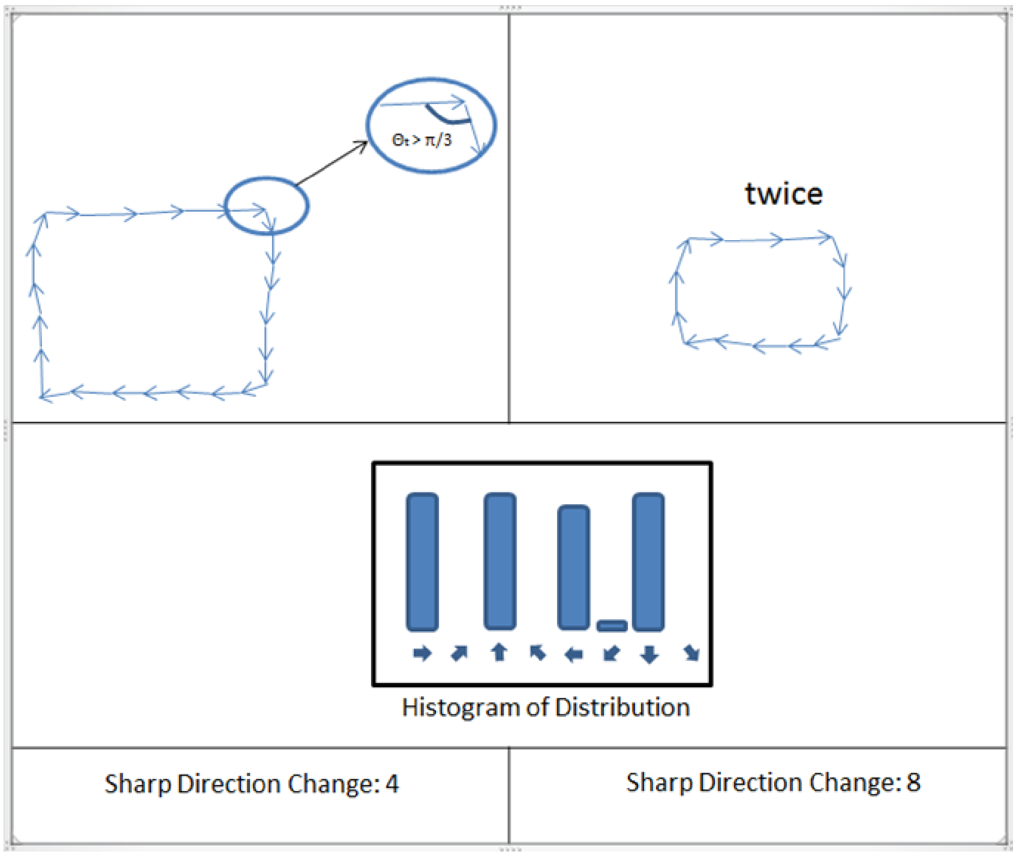

θt > π/3
twice
Histogram of Distribution
Sharp Direction Change: 4
Sharp Direction Change: 8

**Figure 13: Add "Sharp Direction Change Times" as another Feature**

In real applications, we also need to consider the fact that users are all different. It is nearly impossible to define the perfect standard trajectory for everyone. Sometimes even though the user thinks the trajectory is very clear, his real performance could be different. In order to increase the recognition rate, we decrease the threshold for trajectory matching, but ask the user to use both hands to complete a motion. For instance, for "square" detection, the user needs to use both hands to draw left and right halves of the square. (Figure 14) Our system will detect if both the user's hands are doing the corresponding motion at the same time. This approach which requires using both hands should help decrease the chance of misunderstanding due to unintentional gestures.

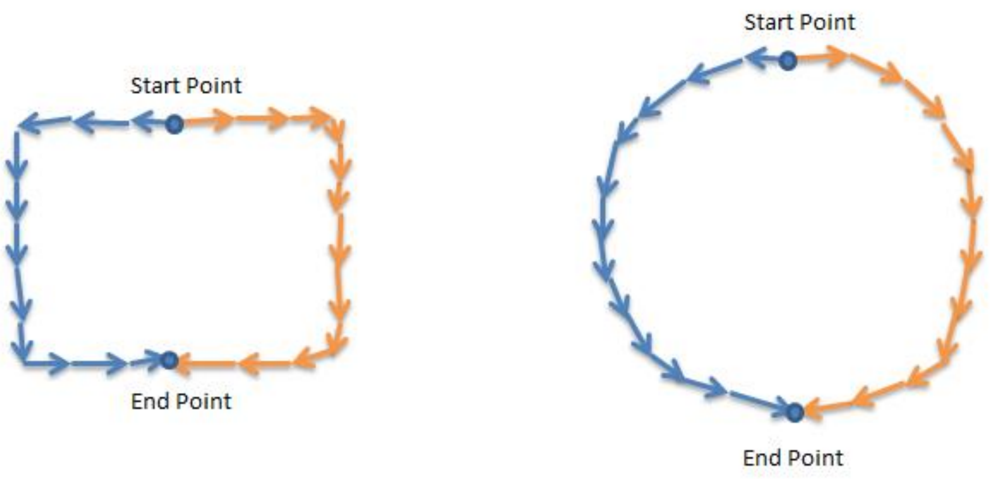


**Figure 14: Both Hands "Drawing" Shapes**

## 4.6 OTHER GESTURE AND POSTURE RECOGNITION

In order to make our system more expressive, we needed to define sufficient gesture/posture input in preparation for complicated system operations in applications. In many cases, we only need the latest value in the array for recognition. For instance, DetectFeetSplit() is a function to detect if the user is standing with his feet apart. In this function, we only need the top value from the array of “left foot” and “right foot”. A complete set of currently supported actions are shown in Table 1. We will elaborate how to use these actions to perform virtual object modeling tasks in the next chapter.

**Table 1: Supported Functions of the Algorithm**

| Function | Description |
|---|---|
| **DetectSquare()** | **Return the similarity value to “square” trajectory** |
| **DetectCircle()** | **Return the similarity value to “circle” trajectory** |
| **DetectTriangle()** | **Return the similarity value to “triangle” trajectory** |
| **DetectCover()** | **Return the similarity value to “cover” trajectory** |

| | |
|---|---|
| **DetectWave()** | **Return the similarity value to “wave” trajectory** |
| **HeadLeaning()** | **Return the direction of head leaning** |
| **ShoulderLeaning()** | **Return the direction of shoulder leaning** |
| **HandClose()** | **Detect if hands are close together** |
| **LimbCross()** | **Detect if arms are doing a “crossing” posture** |
| **IsShaking()** | **Detect if a joint is shaking within a small area** |
| **IsFootSplit()** | **Detect if feet are split apart,** <br> **left and right, or back and forth** |

# 5. GAME DESIGN

All of our efforts are made to make sure that we could develop a user-friendly user interface dedicated to solving virtual object modeling tasks at the application level. With our core gesture system established, whether we could build up an elegant and efficient UI would determine the success of our work. In this chapter, we will discuss the development of our application—the 3D puzzle game which involves virtual object modeling. We will highlight the design of the user interface.

## 5.1 GAME OVERVIEW

While *Wake up the Box 4* [1] gave us our initial inspiration, another inspiration of our work, a short sci-fi file *World Builder* (Figure 15) [29], told us what the interaction might look like. The concept of *World Builder* is about building up a virtual world within the virtual world itself. The protagonist simply uses his body as an input device and starts by creating and customizing basic virtual objects, such as cube buildings. He later adds details to each box and makes them look like buildings. He also has the ability to make windows, doors, and even flowers.  When he finishes, those objects no longer look like computer models but physical objects. Even though it is only a concept video, it points out an interesting direction for 3DUI development. While obviously most of the technology in this video is not available in our world, some basic functionality deserves scientific research interest.

**Figure 15: World builder by Bruce Banit**

*Memory Hacker* is the game we developed. It is a 3D puzzle game and the general idea is to let the user place virtual objects to solve physics-based puzzles. In each level, a wisp will walk from a start point to a destination point, following a fixed, prescripted route (which varies on different levels). We treat this behavior as “walking to a destination based on memories” (Figure 16). However, our poor wisp lost part of his memories and will fall into the hole or will be blocked by obstacles due to confusion in his memory. The player, as a memory hacker, needs to “repair” this route in his memory to help him walk to the destination. Normally, the way of repairing this route is to cover the missing gap with an appropriate object. As the difficulty level increases, the player needs to adjust the shape of the objects accurately to guide the wisp to the destination.

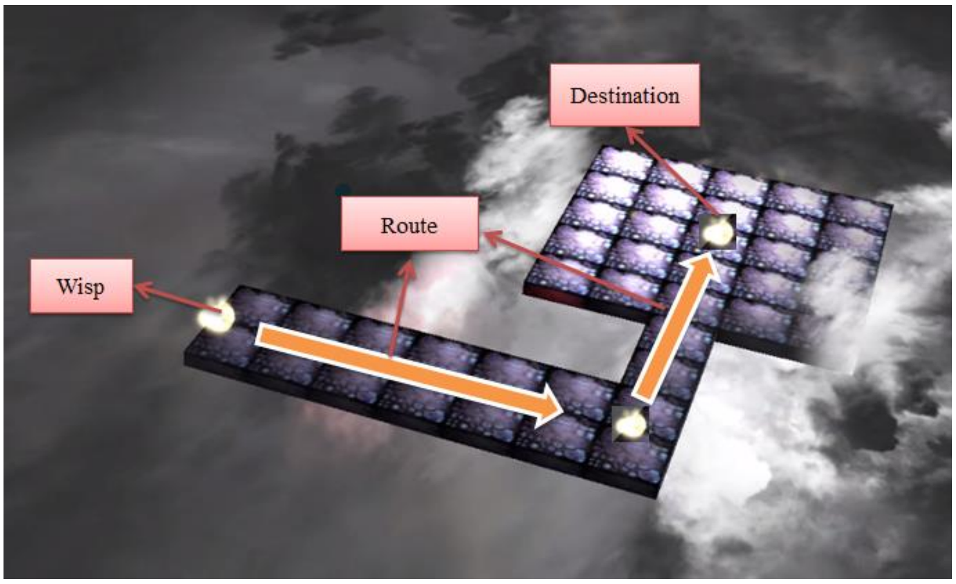


**Figure 16: Basic Game Logic**

## 5.2 GAMEPLAY

The basic game flow is shown in Figure 17. When the game starts, the user can choose from “NEW GAME”, “CREDITS” and “EXIT”. Once the user selects the “NEW GAME” option, he will be brought to the level selection panel. When the user finally chooses a level to play, he enters the game level. No matter the user fails to succeeds in the puzzle, he will also be led back to “LEVEL SELECTION” panel. If the user successfully solves a puzzle and the time is shorter than previous tries, the new record will be updated and shown in the “LEVEL SELECTION” panel.

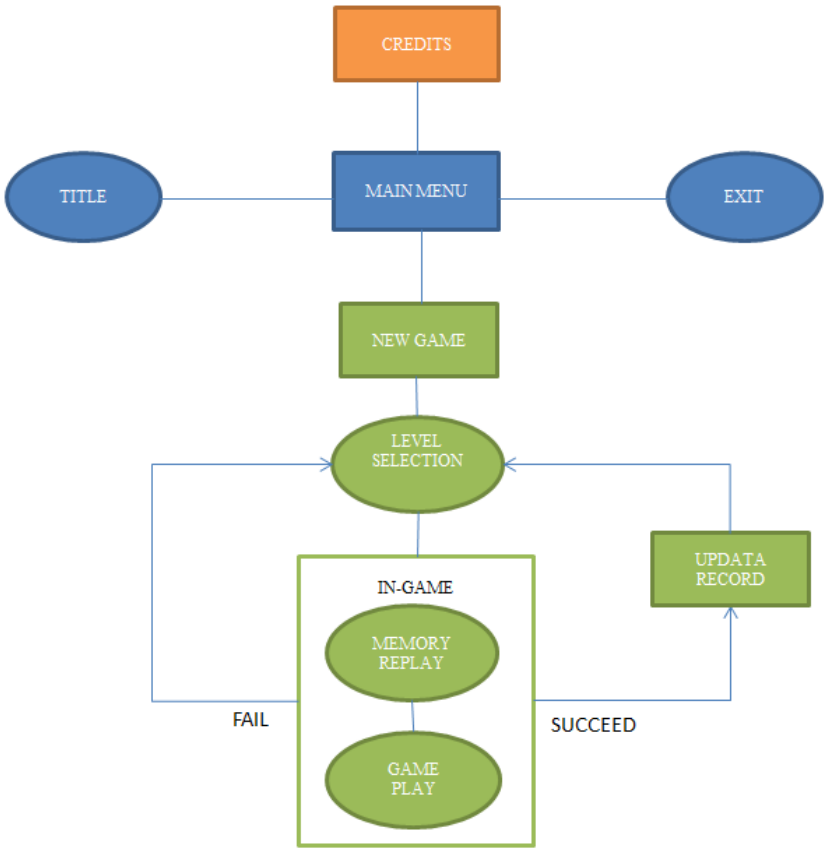


**Figure 17: Game Flow**

### 5.2.1 FIRST MINUTE

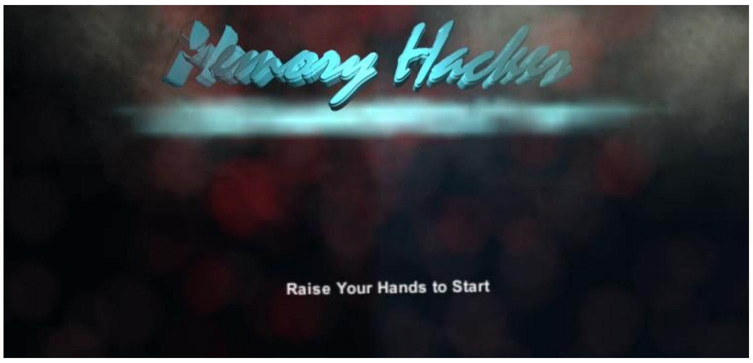


**Figure 18: Title Screen**

Once the game is launched, a title screen is presented to the player. After a short animation, the name of the game “Memory Hacker” will show up on the top of the screen. At the same time, a warning message “raise your hands to start” will appear at the bottom section. (Figure 18) The player needs to raise both of his hands to active the game menu. Once player selects “New Game”, the player will be led to the level selection panel. A total of six boxes representing six different levels will show up on the panel. (Figure 20) On each level box, the player can read the best time record. When the player confirms a selection of the level to play, he enters the level immediately. At first, a memory replay will be presented to the player. The player can learn the “route in the memory” of the level. The player can do nothing at this time but just watch. Once the demo ends, the player enters the playable level. (Figure 21) The description of the current level will appear in a window at the left of the screen. A clock indicating how much time is left to finish the level will be located in the left-top corner of the screen. At the right side, there is a function bar. The player needs to use his gesture to select the first function-- “Creation”. Then the player performs the correct gesture to create a cube. After this, the player selects the second function-- “Translation”, and gets ready to adjust the position of the cube…

## 5.3 GAME ACTIONS

### 5.3.1 SYSTEM CONTROL ACTIONS

In Kinect-based gaming environments, the human body is the exclusive controller, which replaces traditional input devices like the mouse and keyboard. This means the user would be able to control the game flow without using other devices. Our game includes the system control actions listed in Table 2.

**Table 2: System Control Actions**

| **Action** | **Description** |
|---|---|
| Menu Selection | Choose an option in the menu (e.g., New Game) |
| Level Selection | Choose a level to play in level selection panel |
| Move Control | Navigation in the virtual 3D game environment |
| Modeling Function Selection | Choose one of the modeling functions from the Modeling Function Bar |
| Play Move | Finish adding objects, and test if the solution can solve the current puzzle |

### 5.3.2 VIRTUAL OBJECT MODELING ACTIONS

Virtual object modeling actions are the core actions in the gameplay of our game. In the beta version of our game, we provide the following modeling actions for the player shown in Table 3.

**Table 3: Object Modeling Actions**

| **Action** | **Description** |
|---|---|
| Object Creation (Cube) | Create a basic cube model |
| Object Creation (Sphere) | Create a basic sphere model |
| Translation | Translate the current model |
| Scaling (Size) | Scale the model proportionally along three axes |
| Scaling (Shape) | Scale the model along three axes individually |
| Rotation | Rotate the model |
| Delete | Delete the current model |
| Drop | Change the model into a "real" object in the virtual environment, dropping it to the surface |

## 5.4 INTERFACES

### 5.4.1 HUDS

In the "Game Title" panel and the "Level Selection" panel, the user needs to wave his hands to control the curser (the red magic fire in Figure 19 & Figure 20). The left-hand fire is used to select one of the available choices on the screen. Once the user selects an option, a corresponding animation effect would occur.  If the user wants to confirm the current choice, he needs to use his right hand to perform a "Confirm" action in the "Confirm Area" (Figure 20).

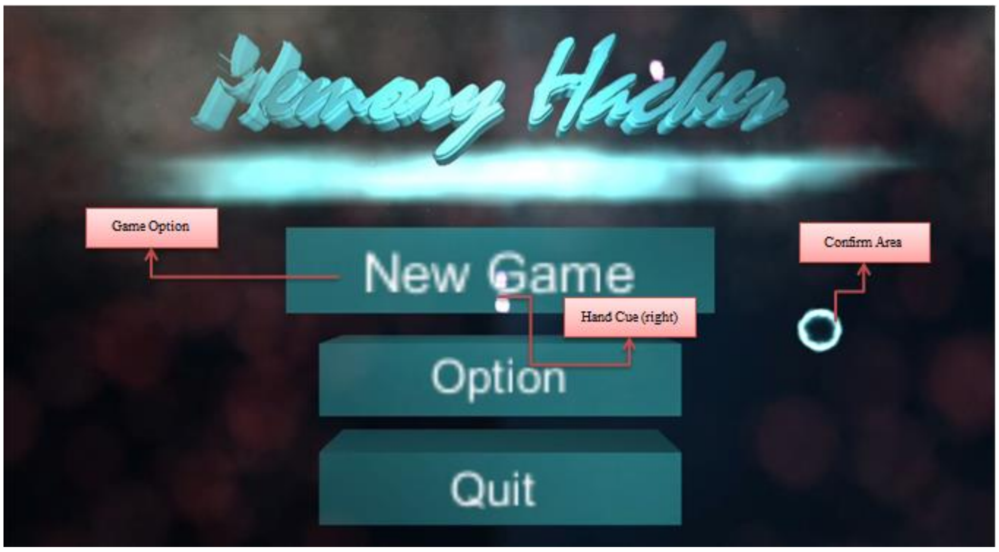


**Figure 19: Game Title Panel**

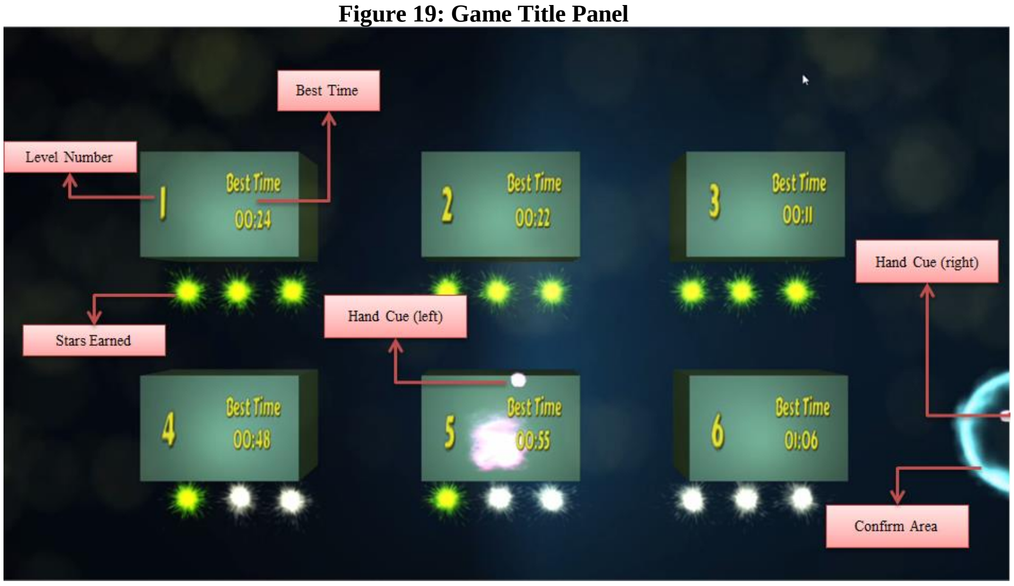


**Figure 20: Level Selection Panel**

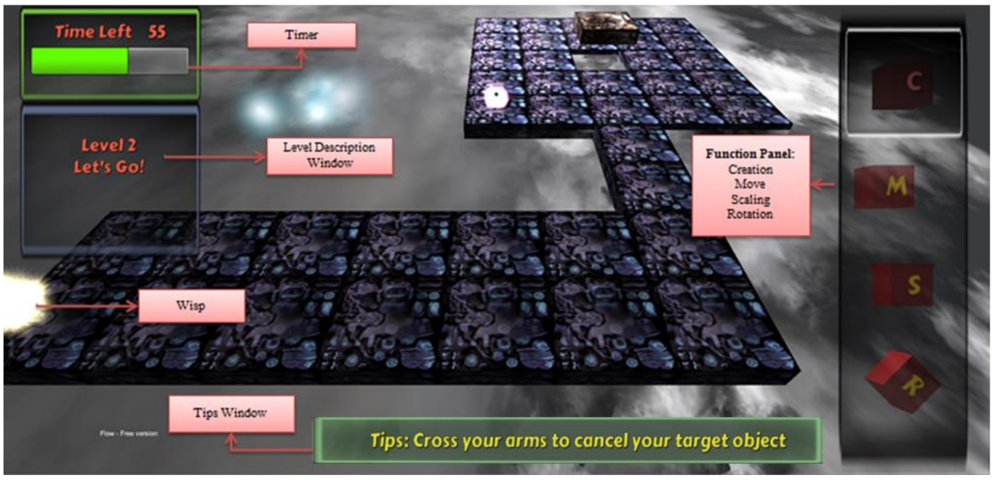


**Figure 21: Game Interface**

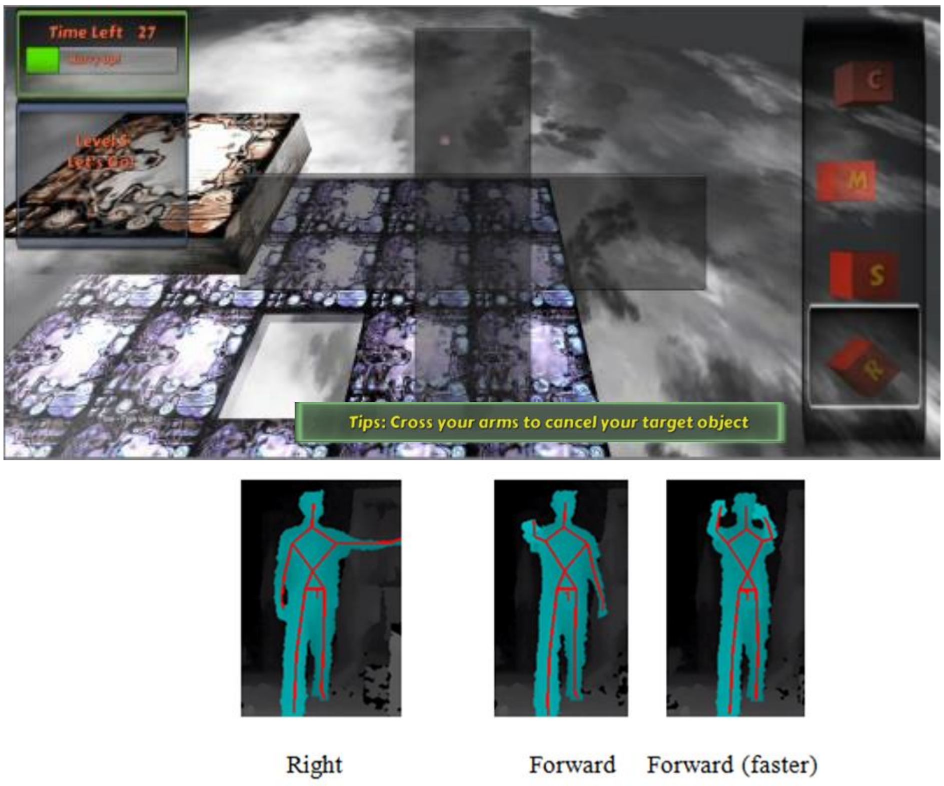
Once the user enters the “Game” interface, he can find a timer and a description window at left side. A “Tip” window shows up at the bottom which gives user guides and tips during the gameplay. At the right side, there is a function bar which is used to switch between different modeling functions (Figure 21).

### 5.4.2 CONTROL WITH TBTM-SIFT RECOGNITION SYSTEM

**Table 4: Modeling with TBTM-SIFT Recognition System**

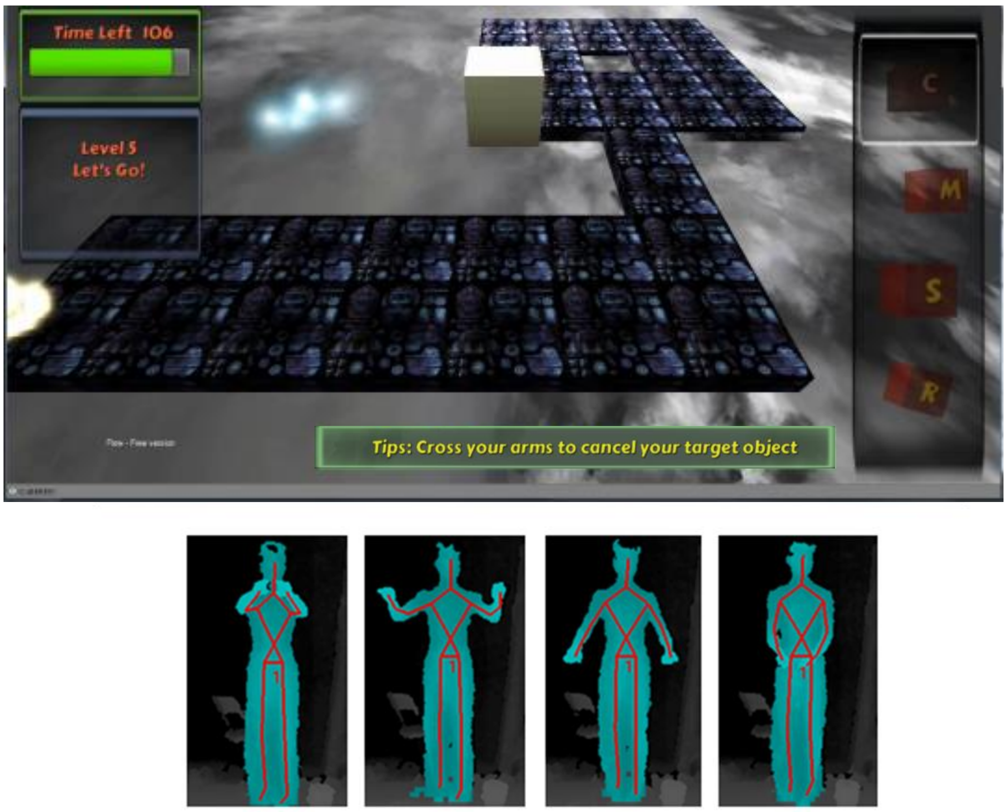
| Action | Solution |
|---|---|
| **Menu Selection** | **Left hand controls the curser<br>Right hand shakes in the “confirm area” to perform a “click”** |
| **Level Selection** | **Left hand controls the curser<br>Right hand shakes in the “confirm area” to perform a “click”** |
| **Move Control** | **Separate feet left and right to trigger “move” state<br>Lift right arm to point the move direction** |
| **Modeling Function Selection** | **Separate feet forward and back to trigger “function selection” state<br>Lift right arm to select the function<br>Close feet to trigger “function selection” to confirm the selection** |
| **Play Move** | **Under “function selection” state, lift both arms up** |
| **Object Creation (Cube)** | **Select “Creation” mode<br>Hands draw left and right half of a “square”** |
| **Object Creation (Sphere)** | **Select “Creation” mode<br>Hands draw left and right half of a “circle”** |
| **Translation** | **Select “Translation” mode<br>Clap hands to trigger “translate” state<br>Left hand controls the movement of the model<br>Both hands stay still for 2 sec to leave “translate” state** |
| **Scaling (Size and Shape)** | **Select “Scaling” mode<br>Turn shoulders to left and right to change mode between “Size Scaling” and “Shape Scaling”<br>Clap hands to trigger “Scaling” state<br>Use both hands to scale the object<br>Both hands stay still for 2 sec to leave “translate” state** |
| **Rotation** | **Select “Rotation” mode<br>Clap hands to trigger “Rotation” state<br>Use both hands to rotate the object<br>Both hands stay still for 2 sec to leave “translate” state** |
| **Delete** | **Cross arms to delete the current model** |
| **Drop** | **Separate feet forward and back to trigger “function selection” state<br>Push arms forward to confirm all the modeling operations and drop the object into the game environment** |

A general description of how to use our TBTM-SIFT recognition system to perform different object modeling tasks is shown in Table 4. It is noteworthy that, in our UI system, we use a “modal selection system” to let the user select between different modeling functions. The difference between “modal selection” and “non-modal selection” is that, in non-modal selection, the user has no visual UI. The user relies on another set of gestures to tell the system what function he is going to perform. Undoubtedly, without a visual UI, the user will be less aware of the fact that he is using an artificial interaction system. This would be a great help on increasing the “presence” feeling for the user in the virtual environment. However, the cost is that the user would have to face a much more complicated interaction system. As the number of available functions increases, the user would need to learn many more gestures, which at the same time need to be distinguished from each other. Eventually, the interaction manner will become awkward. That would run counter to our desire. On the other hand, using a modal selection scheme would simplify the interaction and help improve system efficiency, but it is unavoidable that user needs to frequently notice the existence of the artificial system. It is a trade-off between the user experience and system efficiency, which also partly affects the user experience. In view of this theory, we designed our UI system by adding a modal selection scheme merely for the user to navigate to the categories of modeling functions, but giving the user complete freedom to perform all the modeling activities purely using body gestures. Figure 22- Figure 25 show some important modeling tasks performed using our user interface.



Tips: Cross your arms to cancel your target object

**Figure 22: Camera Move**



Tips: Cross your arms to cancel your target object

**Figure 23: Object Creation (Cube)**

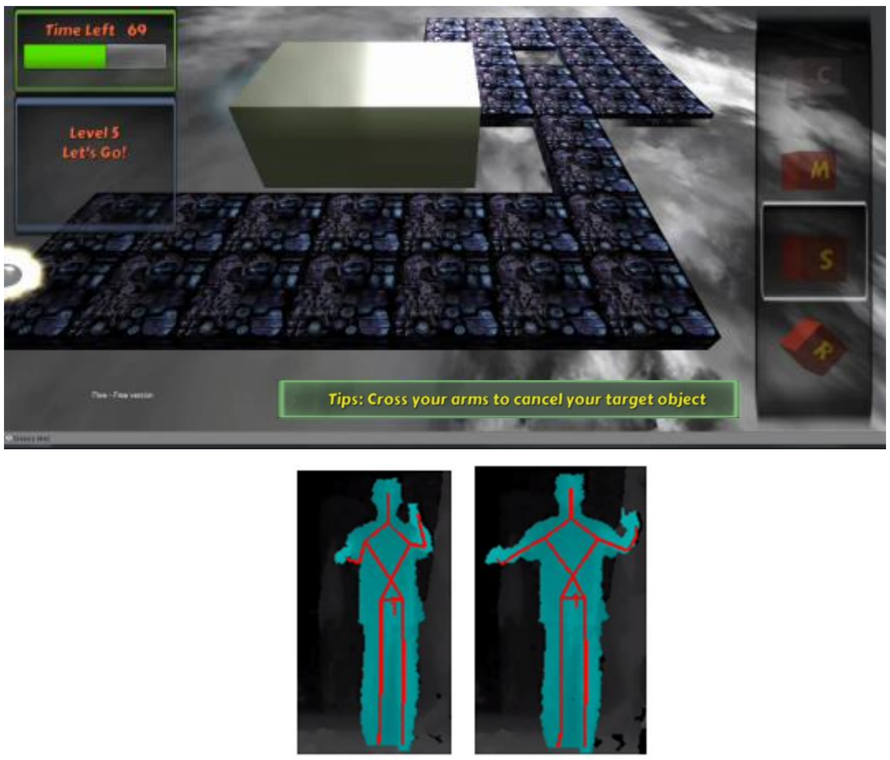


Figure 24: Scaling (Shape)

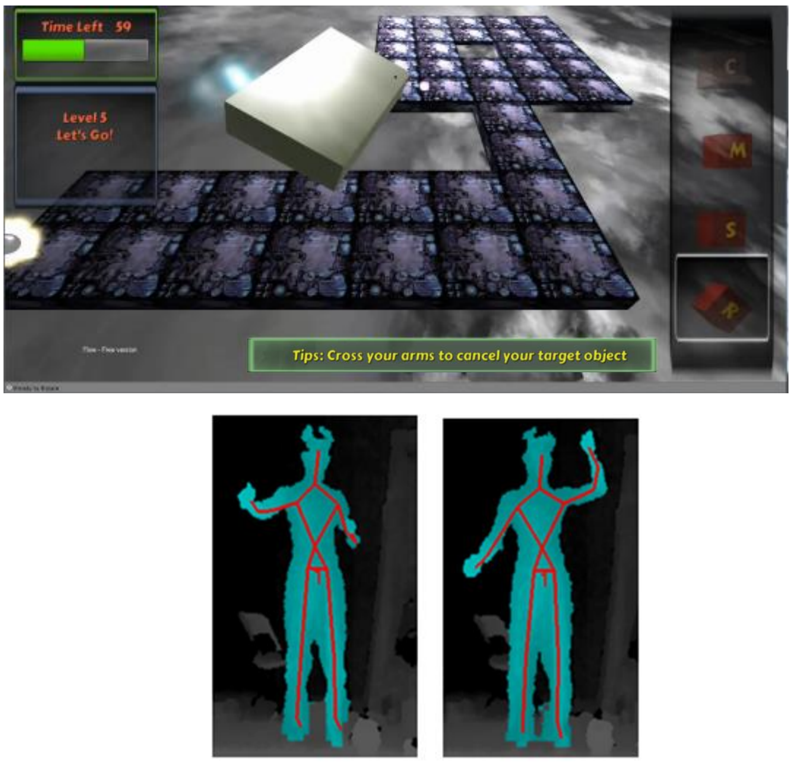


Figure 25: Rotation

# 6. EVALUATION

Since the purpose of the project is to create an effective system which supports a Natural User Interface in a real gaming context, our user study is designed to answer two questions: (1) what are the most natural ways to interact with the target game environment, and (2) what are the most effective ways to complete the tasks given in the game. In consideration of the fact that participants may not be able to provide a flawless solution in limited time, or that participants' solutions may not be practical, we defined a set of gestures tailored to our system performance. We asked the user whether they accept our complete solution.

## 6.1 STUDY DESIGN

We prepared two different studies for the participants. The first study was designed in a survey style to gather ideas from the participants. We first introduced the research question and described the gameplay to the participants. Then we asked them to give their ideal gestures for performing each action that may happen in the game. We also discussed with our subjects the extension of other possibilities of the gameplay just to help stand in a higher level when providing a design plan. The second study is an experiment designed to evaluate the whole system and the gameplay. In this experiment, participants used a scheme we designed to perform the game task. They were then asked about the experience and give comparison between their solutions and our solutions.

We hypothesized that participants would have better understanding of the research question, and would either have new ideas or revise their previous solutions to be more practical.

## 6.2 SUBJECT DEMOGRAPHICS

A total of nineteen people participated in the study (thirteen male, six female), with a mean age of 22.42. Among them, 16 participants have played video games which have a 3D environment. When participants were asked their experience playing motion based consoles, sixteen participants indicated that they have played Wii Remote, Xbox Kinect, PS Move Controller and other motion sensing input devices before. 50% of these sixteen participants have played Kinect games. Most of the participants were recruited from a general psychology course, and were offered a research credit for participating.

## 6.3 EQUIPMENT

We ran experiments in the tech suite of the school library. Such a tech suite is a private workspace offering an opportunity to perform our user study. Most of these tech suits are more than eight square meters and are equipped with the following standard technology: (1) 50” plasma display, (2) Dedicated networked PC, (3) Wireless coverage for the entire room, (4) Ability to connect two additional laptops using ports in the pop-up receptacles on the desk and (5) Whiteboard.

The Kinect sensor is the core equipment used in the experiment and no equipment is required to be worn. The experiment was run on a 3rd Generation Intel Core i7 2.30GHz PC running Windows 7 with a total of 6GB of RAM and an NVIDIA GTX 660M graphics card.

## 6.4 DETAILED PROCEDURE

The study contained two sub studies. Each one took approximately 30 minutes to complete. Participants were initially given an opportunity to read the informed consent form and ask questions about the study. After participants signed the informed consent form, they were required to complete a demographics survey. As part of this questionnaire, they were asked their experience of playing video games, especially those which have 3D context. They were also invited to share their game experience using motion-based controllers. For those who had rich experiences, we discussed with them the advantages and disadvantages of different devices. We focused on how they feel when using these devices. If they happened to have an idea to improve the usage of these devices, we would write them down.

After completing the pre-questionnaires, the subjects were introduced to the whole project. More detailed descriptions were given if the participant was confused or curious. Those questions which were not quite related to the user study were explained briefly and promised to be given more deep explanation after the user study. In order to help the participants understand the research question, we prepared one flash game to explain the inspiration and a video to demonstrate the concept result.

For those who were not familiar with Kinect games, we spent some more time on explaining the mechanism to them. Then, we made clear to participants why we were doing the user study, and what they could do to help us gather useful information.

When the participants were ready, we introduced the two sub studies and began with the first study. We first introduced detailed gameplay and then pointed out the possible behaviors, from level selection to level completion. We then presented them a paper that listed all the important behaviors that may happen in the game and asked them to describe their solutions. We also asked the participants the following question:

Q1: How would you let the system know what commands mentioned you are giving?

Which one do you prefer?

a. Non-modal selection: Make the gesture to execute the action. (do a gesture to tell the system what task you are performing)
b. Modal selection: Select the mode first, and then make the gesture to execute the action. (Use a gesture to switch between these modes.)

It is important to note that we did not give tips or anything that may affect the user's solution. We only explained to them when we found they misunderstood a certain question. We also pointed out to the participant if we found his solution impractical. Normally, they would have a much better idea once they met with such a problem. When participants confirmed their solutions, first study ends. It is very likely that their final solutions still had flaws. However, most parts would be feasible.

We then presented a video which had been recorded in advance to demonstrate our design. If there was a big difference between the solutions, we would summarize the two solutions and explain, if any, the potential problems to them.  We then setup the experiment and invite them to use our system. When everything was ready, we asked the participants to try each behavior we just discussed. When participants got used to these behaviors, they were free the play the game and attempted to solve the puzzle. We also gathered some game bugs during their play. We usually left 10 minutes for the participants to write down any feedback of our system. We asked them to rate their feelings on each behavior solution and recorded them to form a 1-7 scale (1 = terrible, 4 = normal, 7 = excellent). It should be noted that we encouraged participants to give low score if they feel awkward on any solution even if it worked. Besides the Likert scales, dedicated to be used to evaluate existing behavior solutions, we also asked them if they liked our UI style. We provided two more questions in this section to help them evaluate our system:

Q2: When giving gesture/posture input, which of the following ways do you feel more natural and comfortable?

a. Continuous gesture recognition system

b. Prompted gesture recognition. (Giving commands after specially appointed signals in dialog style)

a. Standing at a fixed location

b. Standing at arbitrary location

Finally, we asked them to write other comments and discussed them to gather qualitative feedback. In particular, we always asked participants to point out the worst design part of the whole system.

## 6.5 RESULTS AND DISCUSSION

### 6.5.1 PART1: SOLUTION GATHERING

Based on the demographics data, subjects were divided into four groups: (1) Female who has used motion tracking based devices; (2) Female who has not used motion tracking based devices; (3) Male who has used motion tracking based devices; (4) Male who has used motion tracking based devices. When organizing their solutions, we found common ideas and recorded them into Table 5.
(Note: Sometimes, the entire solution for a single action may vary, but part of the solution was very similar. We also recorded these ideas and treat them as valuable information.)

**Table 5: Most Selected Solutions or Ideas from the Participants**

| Action | Most Selected Solutions or Ideas |
|---|---|
| Item Selection (System Control) | Use one hand to select the item, and then dwell over for a while to confirm the selection (5 participants)<br>Put both hands on the item to select the item (4 participants) |
| Move Control | Turn and Lean body to movement (4) |
| Creation (Cube) | Hands split horizontally (6) |

| | |
|---|---|
| | Draw square with two hands (4) |
| Creation (Sphere) | Hands split vertically (5)<br>Draw circle with two hands (5) |
| Translation | Both hands hold the object to move, separate hands to stop (6)<br>Use one hand to move, use the other hand to stop (3) |
| Scaling (Shape) | Use both hands to scale along different axes (5)<br>Click virtual arrow to scale (4) |
| Scaling (Size) | Use both hands to scale (5)<br>Click virtual arrow to scale (4) |
| Rotation | Use both hands do rotation (6)<br>One hand swipe to rotate (3) |

It should be noted that only a few participants gave a complete set of solutions for all the actions. Also, some of the gathered solutions had big defect and were therefore not feasible. Sometimes, even the participant himself was not satisfied with his own design ideas.  On this basis, the solutions recorded in the above table can be treated as highly centralized results.

For "Item Selection", five participants (three male and two female; four out of the five have used motion-tracking devices before) preferred to "Use one hand to select the item, and then hang over for a while to confirm the selection". This is a very simple solution and it works. Our solution "Use one hand to select, use the other hand to shake to select" aimed to avoid misoperation by applying complex gestures was not quite accepted by our participants. While most participants agreed that our solution made sense and worked, they preferred other ways. As a result, we concluded our first solution as a bad design.

For "Move Control", four participants (four male; all have used motion-tracking devices before) suggested using head or torso as a joystick. That was one of our solutions during experimental stage. But we late found that after several attempts, the user would begin feeling dizzy or nauseous. We therefore would not consider this idea. Apart from this one, there's no other common idea. So we decided to keep our original solution.

For "Creation (Cube)" and "Creation (Sphere)", five participants (four male and one female; all have used motion-tracking based devices before) came to the solution "Hands split horizontally and vertically". While it was indeed a concise solution, it was not an ideal one. We told our participants in the second part of the user study that we would add more basic models later, and then they realized the problem. Every one of the five agreed with the idea that we should use those gestures which made more sense and were easier to remember. Our project aimed to solve large-scale object modeling tasks in the future, and we decided to keep our original solution.

For “Translation”, six participants (five male and one female; four out of the six have used motion-tracking devices before) chose to “hold” the object to move and “release” hands to stop. This could be considered as the most natural manner to move the objects. However, the flaw in this method is that it is difficult to define the critical point between “hold” and “release”. Suppose we define that the object will always stay in the center of the user’s two hands the user is moving it, it is possible that the object would continue moving while the user’s two hands are “releasing” at slightly different speed. While we believed that using two hands are more natural to move an object than using only one hand (our original solution), we decided to use a compromised solution: putting hands really close together to move the object, and separating hands away to stop moving.

For “Scaling”, the most chosen method was similar to ours. The difference was that our solution allowed the user to scale along three axes at the same time. Eighteen out of nineteen participants thought that our solution was better than “scaling along one axes at one time”, and all the nineteen participants agreed that it was good to have two different scaling modes (size scaling and shape scaling). We therefore decided to continue using this solution.

For “Rotation”, six participants (four male and two female; five out of the six have used motion-tracking devices before) thought that the most natural way is to use two hands to rotate as if “holding” a real object in front of the chest. Our method did exactly the same way and was therefore kept.

For question Q1 (Action execution style between mode selection and non-modal selection), thirteen out of nineteen participants (nine male and four female; twelve out of the thirteen have used motion-tracking devices before) chose modal-selection. In these thirteen participants, three stated that modal-selection was very suitable for our application. However, they would prefer non-modal selection if the disadvantages of which are overcome in the future. There was another participant who suggested that if the commands are not too many in an application, he preferred to use non-modal selection. But for our application, he believed that “modal selection” was more suitable. Based on all the views from our participants, we concluded our design as a proper solution.

### 6.5.2 PART 2: SYSTEM EVALUTION

Although, some of our solutions were not exactly the same as the participant’s ones, it didn’t affect the fact that our solutions were highly accepted. Actually, most of the participants stated that our solutions were better than their solutions (Figure 26). As the statics suggest, the average 7-point degree of comfort rating of each section is much higher than 4 (feels OK). “Object Creation” (Cube & Sphere) and “Rotation” received the best comments. “Item selection” (system control) and “Camera control” (movement) had the relative lowest score.

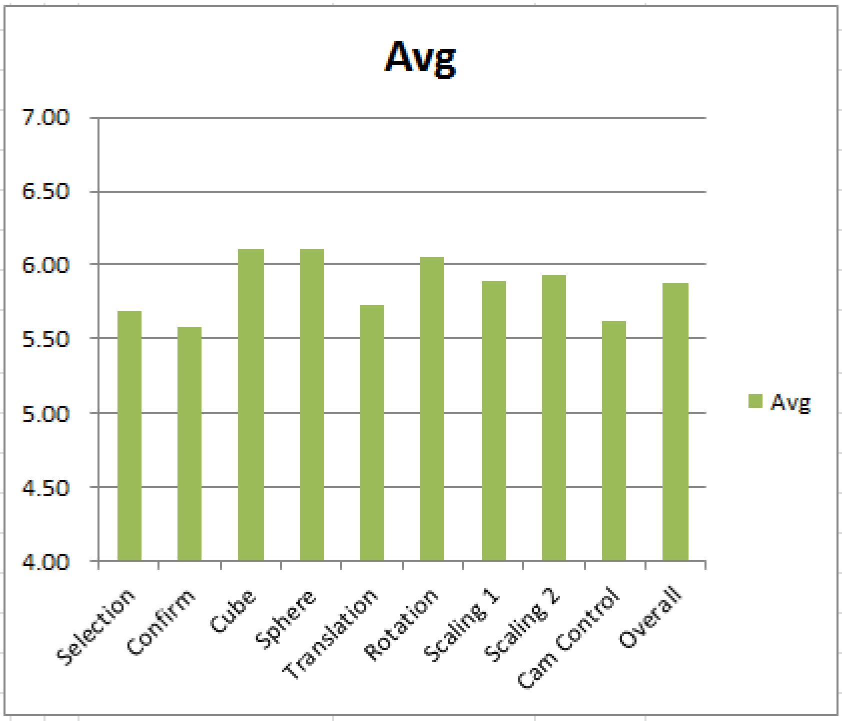


**Figure 26: Rating Scores of the System**

We discussed a lot with every participant on every detail of our system. We found that even though some of our design solutions received same score, the reasons were not quite the same. Participants gave high scores to “Rotation” and “Scaling” actions because participants thought our idea was much better than theirs and solved the problem. At the same time, for “Object Creation” (Cube & Sphere), our solution was

more complex than simply separating hands apart. But participants also gave us high score because they had good user experience on them.
For the question Q2 we presented in this part, twelve participants (seven male and five female; ten out of the twelve have used motion-tracking devices before) preferred continuous gesture recognition, which was one of the features of our gesture recognition system. In these nineteen participants, only half (six male and four female; nine out of the ten have used motion-tracking devices before) of which preferred "Standing at arbitrary position", which was another feature of our system, when giving commands. After further discussion, we found that what participants really meant was that they believed "Standing at fixed position" would help the system understand the gesture. This didn't mean that "Standing at arbitrary position" was useless. Quite the contrary, this proved that users cared about the recognition rate when performing gestures in front of the camera, and our system gave them great help.

## 6.6 DISCUSSION

The user studies we designed have achieved the expected result. In our first user study we showed that people did have common ideas on natural interaction for our application. These common ideas were what we were looking for and were treated as a standard to evaluate our work. The results showed that most of our design met with the users' ideas. At the same time, we also found those designs which were different from the common ideas. We treated them as not ideal solutions even though they worked effectively. We would redesign the methods and make them as close as possible to people's common thought.
Our second study was designed to evaluate the user's feeling when using our system to perform virtual object modeling tasks, and the results showed that people felt good and comfortable. The second study was conducted immediately after the first study. We wanted the participant to give a direct comparison between his solutions and our solutions. In spite of the fact that our solutions were well accepted, participants pointed out some problems. Similar to what we did in the first study, common views were considered as more convincing comments. Many participants stated that our solution for "Item Selection" and "Confirmation" was complex. We therefore decided to change the solution to the most suggested methods. Another problem participants stated was that our solution for navigation in the virtual environment was not very

natural, although it worked perfectly. However, participants could not give a feasible solution. Therefore, we decided to keep our solution while tagging it “not ideal”. Generally, we received high scores for the overall performance of our system. The average score was 5.88 in the 7-point degree of comfort rating.

We also received other valuable comments. These comments covered different respects of our work. Many participants suggested adding tutorials in the first few levels to help the user learn how to use the system. Some participants suggested that we should strengthen the awareness of position in the virtual environments to help increase the accuracy of the modeling work. For instance, we could add a mini map to tell the user’s position in another view aspect. Some participants wanted us to add more basic models so that our system could be more powerful. We thought all of these comments were of great value and plan to add these features in the next version of our system.

# 7. CONCLUSION AND FUTURE WORK

In this project, we developed an immersive 3D user interface for a virtual object modeling application. We used an Xbox Kinect sensor to set up the working environment, which required no aid from other body-worn devices. We expanded the usability of the Kinect sensor by designing a human body gesture & posture recognition system. The recognition system has many advantages compared to other systems: (1) It does not require huge time and space resources and can work in an interactive way; (2) It does not require the user to perform the gesture at a specific speed; (3) The user does not have to do extra “start” & “end” gestures to help the system select the meaningful motion division; (4) It can recognize gestures in three dimensions; (5) It is compiled into a c# library, and could be imported to any application that is run in a c# environment; (6) The target device does not need to be the Kinect sensor, as any motion-tracking based device which can tell the position information would be able to make full use of our algorithm.

We developed an object modeling based puzzle game and gave the player high freedom for performing virtual object modeling activities. When we finished the beta version of the game, we conducted a user study. In this user study we found out the users’ common ideas on natural interaction between human and computer. We also gathered other useful ideas on developing such an application. In the second part of the study, we let the participants use each function we designed in the application and asked them to describe their experiences. We received many useful comments. For instance, many participants thought that our solutions for “Item Selection” and “Navigation” were not quite natural and needed further modification. In spite of these comments, the overall results were promising. We got an average score of “5.88” in a 7-point degree of comfort rating. Thus, we concluded that people felt good and comfortable when using our system to perform virtual object modeling tasks.

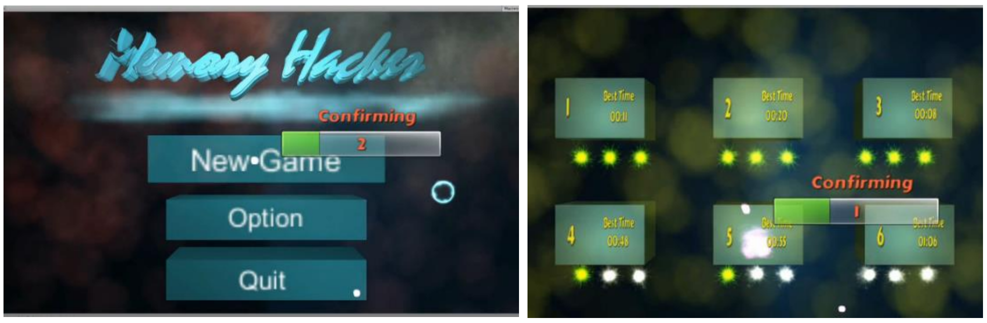
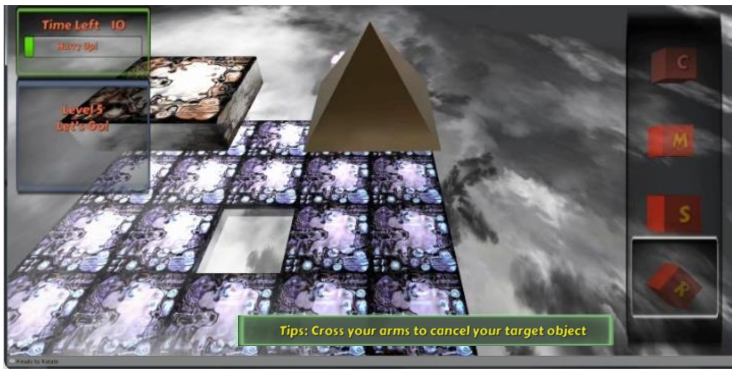


**Figure 27: Use "dwell time" as the selection method**



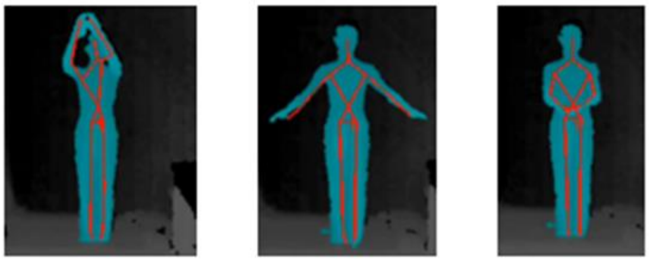


**Figure 28: Object Creation (Pyramid)**

However, we noticed that some of our solutions were not the same as the most ideal ones from the ideas we gathered from the participants. Although most of the participants stated that our solutions worked effectively, we thought about changing “not ideal” solutions since they were not conducted in the most natural manner. For those functions that got a better idea from the participants, we simply replaced them in our system. In the latest version of our application, we changed the gesture input for “Item Selection” to the most ideal solution gathered from the user study (Figure 27). For those functions that had no better alternative (e.g., “Navigation”), we will explore the research in that filed, and try to come up with a better solution in the future. We added one more primitive model—“Pyramid” (Figure 28) to give our users more choices when solving the puzzles. It is also important to improve our gesture recognition system to support more complex gesture input. In the next step, we plan to extract more features of the joint motion in the algorithm and combine them with body postures. In the next game version, we will add more features based on the participants' comments on gameplay. Finally, since we treated our application as the first stage of the big concept project—*world builder*—we would put in more effort to create more possibilities and eventually realize the real "virtual world builder".

## 8. REFERENCES

[1] Wake up the Box 4 http://www.notdoppler.com/wakeupthebox4.php

[2] N. Dalal and B. Trigges.: Histograms of oriented gradients for human detection. CVPR, 1 (2005) 886-893

[3] N. Dalal, B. Triggs, C. Schmid.: Human detection using oriented histograms of flow and appearance, in: European Conference on Computer Vision, Graz, Austria, May 7–13, 2006

[4] S. Ikemura, H. Fujiyoshi.: Real-Time Human Detection using Relational Depth Similarity Features. ACCV 2010, Lecture Notes in Computer Science, 2011, Volume 6495/2011, 25-38

[5] DG. Lowe.: Object Recognition from Local Scale-Invariant Features. Proceedings of the International Conference on Computer Vision. 2 (1999). pp.?1150–1157

[6] I. Laptev, On space-time interest points, International Journal of Computer Vision, vol. 64, no. 2, pp. 107–123, 2005

[7] M. Chen, and A. Hauptmann, Mosift: Recognizing human actions in surveillance

[8] Michalis Raptis, Darko Kirovski, Hugues Hoppe, “Real-Time Classification of Dance Gestures from Skeleton Animation”, Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2011

[9] Real-time Dynamic Gesture Recognition System based on Depth Perception for Robot Navigation

[10] Kinted based dynamic hand gesture recognition algorithm research

[11] Human Gesture Recognition through a Kinect Sensor

[12] Clark, James H. Designing Surfaces in 3-D. Communications of the ACM, 19(8):454-460, August 1976

[13] Jeff Butterworth, Andrew Davidson, Stephen Hench, and T. Marc Olano. 3DM: A three dimensional modeler using a head-mounted display. In Proceedings of the symposium on Interactive 3D Graphics, 1992

[14] Masatoshi Matsumiya, Haruo Takemura and Naokazu Yokoya. An immersive modeling system for 3D free-form design using implicit surfaces. In Proceedings of the ACM symposium on Virtual reality software and technology, 2000

[15] Falko Kuester, Mark A. Duchaineau, The Designers Workbench: Towards Real-time Immersive Modeling. The International Society for Optical Engineering, SPIE, 2000

[16] Daniel Leithinger, David Lakatos, Anthony Devincenzi, Matthew Blackshaw, Hiroshi Ishii. Direct and gestural interaction with relief: a 2.5D shape display, In Proceedings of the 24th annual ACM symposium on User interface software and technology, 2011

[17] "Reconfigured Self as Basis for Humanistic Intelligence", Steve Mann, USENIX-98, New Orleans June 15–19, 1998, Published in: ATEC '98 Proceedings of the annual conference on USENIX Annual Technical Conference USENIX Association Berkeley, CA, USA ©1998

[18] http://www.microsoft-careers.com/content/rebrand/hardware/hardware-story-kinect/

[19] http://msdn.microsoft.com/en-us/magazine/jj159883.aspx

[20] R. M. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. VRPN: a device-independent, network-transparent VR peripheral system. In ACM Virtual Reality Software & Technology, pages 55–61, 2001.

[21] E. Suma, B. Lange, A. Rizzo, D. Krum, and M. Bolas, "FAAST: The flexible action and articulated skeleton toolkit," in IEEE Virtual Reality Conference, march 2011, pp. 247 –248.

[22] Unity indie VRPN adapter (UIVA) http://web.cs.wpi.edu/~gogo/hive/UIVA/

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004

[24] B. Horn and B. Schunk, "Determining optical flow," Artifical Intelligence, vol. 17, pp. 185–204, 1981.

[25] Kinect for Windows SDK. http://social.msdn.microsoft.com/Forums/en-US/category/kinectsdk

[26] Samsung Galaxy S4 User Manual HTTP://ALLABOUTGALAXYS4.COM/GALAXY-S4-USER-MANUAL/

[27] Youwen Wang, "Kinect Based Dynamic Hand Gesture Recognition Algorithm Research," Intelligent Human-Machine System and Cybernetics (IHMSC), 2012 4th International Conference, Vol. 1, pp. 274 – 279, 2012

[28] Dan Xu, “Real-time Dynamic Gesture Recognition System based on Depth Perception for Robot Navigation,” Robotics and Biomimetics (ROBIO), 2012 IEEE international Conference, pp. 689 – 694, 2012

[29] World Builder (concept video)
http://www.youtube.com/watch?v=VzFpg271sm8