HUMAN COMPUTER INTERACTION SYSTEM DESIGN WITH VIBROTACTILE FEEDBACK

By

Pu Dong

Senior Thesis in Electrical Engineering
University of Illinois at Urbana-Champaign

Advisor: Minh. Do

May 2014
Abstract

Human computer interaction (HCI) studies a human being and a computer at the same time. Within an HCI system, a graphical user interface (GUI) as software and input and output (I/O) hardware must be matched. Common sensory channels that can be used to provide feedback to the user include visual, auditory and haptic feedback. An HCI system with haptic feedback is promising and worth studying. In this thesis, we presented how such a system was built from scratch. Three layers were discussed in details which included software, application programming application (API) and hardware. Then, two case studies were provided to illustrate the creation procedure. One was a 2D Tower of London game and the other was a 3D Drag-and-Drop game. They both showed how to use APIs to create software applications. And in the first case study, integration of vibrotactile feedback was documented. This thesis serves as documentation of an HCI system with haptic feedback. It is beneficial to the further development of the system.

Subject Keywords: human computer interaction; haptic feedback; system design
Acknowledgments

I would like to express my gratitude to my advisor, Professor Minh Do, for his continuous support in the project. Professor Minh Do is the first ECE professor I met at University of Illinois and I have always been inspired by his thoughts. I would also like to thank David Jun for spending the time in helping me step by step to build a system in the past year. He generously shared his knowledge and offered me a lot of help.
## Contents

1. Introduction .................................................................................................................. 1
  1.1 Background .............................................................................................................. 1
  1.2 Motivation ................................................................................................................ 1
  1.3 Contribution ............................................................................................................. 1

2. Literature Review ....................................................................................................... 2
  2.1 A Wearable AR System ............................................................................................ 2
  2.2 A Kinesthetic Pen for Virtual Interaction ............................................................... 3
  2.3 ImmersiveTouch™ ................................................................................................. 3

3. System Design Overview ............................................................................................ 5
  3.1 An Example ............................................................................................................. 5
  3.2 Software Application ............................................................................................... 6
  3.3 APIs ......................................................................................................................... 6
  3.4 Hardware ................................................................................................................. 7

4. Case Study: 2D Tower of London ............................................................................. 9
  4.1 Hierarchy of the Game ............................................................................................. 10
  4.2 Main Application Window ...................................................................................... 10
  4.3 Control Panel ......................................................................................................... 12
  4.4 Game Entity ........................................................................................................... 14
  4.5 Leap Motion Hand Tracking System ..................................................................... 18
  4.6 Vibrotactile Feedback ......................................................................................... 18

5. Case Study: 3D Drag-and-Drop Game .................................................................. 20
  5.1 Motivation .............................................................................................................. 20
  5.2 Hierarchy of the Game ............................................................................................ 21
  5.3 AR Module ............................................................................................................... 21
  5.4 Leap Motion Module ............................................................................................... 22
  5.5 Problems ............................................................................................................... 22

6. Future Research ........................................................................................................... 24

7. Conclusion .................................................................................................................... 25

References ....................................................................................................................... 26
1. Introduction

Human computer interaction (HCI) studies the way people interact with computers and designs human and computer activities. Studies on human computer interaction have been focusing on enriching the immersive experience of users by using multiple sensory channels which are based on the senses of sight, hearing and touch [1]. The incorporation of haptic feedback inside an HCI system is promising and worthwhile.

1.1 Background

Naturally, people use multiple sensory channels to interact with the world. Feedbacks in HCI systems have been accepted already. Some guidelines exist to give developers assistance to provide users with feedback. Although vision and hearing are still two dominant ways to produce it, haptic feedback is an additional way to achieve closed loop control in the HCI system [2]. Vibrotactile is the form we take for haptic display because it is inexpensive and easy to prototype [3].

1.2 Motivation

To study haptic feedback in HCI, a system is needed which supports both the basic form of interaction between human and machine with software application and generation of haptic sensations with hardware circuit [4, 5]. In addition, to enrich the usability of the system, a platform is necessary on which different application software can be created. An HCI system with haptic feedback can be analyzed in multiple layers which, for the purpose of development, mainly are software application layer, API layer and hardware circuit layer. For one thing, the documentation of each layer on which future modification, improvement and continuing development of the system can rely is necessary. For another thing, haptic technologies have existed for a while but it has not been applied widely inside HCI systems. Therefore, it is quite meaningful to document the incorporation of the haptic technology.

1.3 Contribution

My contribution to the platform was mainly on software application. It included setting up the software development environment, searching for and learning proper APIs and creating one 2D HCI game as well as one 3D scene for HCI research purpose using augmented reality (AR). The circuit module which controlled and generated the sense of vibration was contributed by David Jun. I also participated in design of the behavior of vibrotactile actuators and how it was linked to the front software application.
2. Literature Review

Recently, there has been much research on introducing haptic feedback to an HCI system such as a wearable AR system with haptic feedback by Kazuya Murakami et al. [5], a kinesthetic pen for interaction in virtual environment by Sho KAMURO et al. [4] and ImmersiveTouch™ by Cristian Luciano et al. [6].

2.1 A Wearable AR System

The wearable augmented reality system takes force feedback as the form of haptic feedback. Besides studying the effectiveness of haptic feedback in an AR system, this system design mainly focuses on the wearability of the system so as to provide the user with unconstrained space of movement. This system mainly has one wearable haptic device named HapticGear, one head mounted display (HMD), one laptop, one controller for Haptic Gear and two cameras. One camera is for taking the images from the head and the other camera is for tracking HapticGear.

The HapticGear is a haptic display of forces. A control pen is used by the user to pick and move virtual objects. Picking an object is replaced by clicking the button on the pen. Virtual gears are generated and used for experiments. Figure 1 shows the result. Although the simple task does not lead to significant difference in time, the error rate obviously decreases with haptic feedback.

![Figure 1. Time required to finish the test and errors [5]](image-url)
2.2 A Kinesthetic Pen for Virtual Interaction

This research focuses on designing a user-friendly haptic display without mechanical linkages that can be used in ordinary AR system. Figure 2 shows the designed kinesthetic pen. A downsized application is created by the haptic display, an AR scene created by ARToolKit and an infrared camera which is used to capture the position of the haptic pen’s tip. The interactive scene from the user’s view is shown in Figure 2. By getting the tip’s position, the value of the force that needs to be fed back can be calculated.

![Image of haptic AR system](image)

**Figure 2. Haptic AR system [4]**

2.3 ImmersiveTouch™

Many AR applications or systems move with the user. Examples are AR applications on cellphones and the previous two. However, ImmersiveTouch™ presents an integrated solution of immersive touching experience in an augmented reality scene.

It constrains the user to a workplace with a certain area, basically a table. Figure 3 displays the workplace. In the hardware part, it also takes force feedback. Instead of using ordinary computer graphics where user can only see the virtual object in the monitor, ImmersiveTouch™ designs its own rear-projection-based virtual reality (VR) devices. It can create a virtual object by projecting an image on the screen which is between the user’s eyes and his hands. Also it has the ability of head and hand tracking. On the software side, it integrates several libraries to create its own API.
Figure 3. Workplace of ImmersiveTouch™ [6]
3. System Design Overview

To study HCI and haptic feedback, the system is a platform where games and psychological experiments can be built. A game or task is created in the computer and it is displayed. The user plays the game or completes the task with assistive haptic feedback. Quantitative measurements of movement time and error rate can be done by the system to evaluate the user’s task performance. Furthermore, the effectiveness of the vibrotactile feedback device can be studied. The platform consists three layers which are software application, API and hardware circuit.

3.1 An Example

For example, the Tower of London test is a famous test in neuropsychology to evaluate a person’s planning ability. Figure 4 shows how the Tower of London test looks in a real scene. A virtual Tower of London test is achievable using the platform which displays the virtual test objects on the computer monitor. Instead of using mouse to assist object movement, a hand motion detector is incorporated so that the testee can move the virtual objects by the real hand’s movement. At the same time, the testee receives the feeling of vibration if he touches the virtual objects.

![Figure 4. Tower of London Test](image)

Similar studies which can be done on the platform include studying the haptic effects using vibrotactile motors with varied amplitude, wave shape and duration.
3.2 Software Application

This is the first layer accessible to the end-user directly. In general, software is a collection of instructions and data saved in the computer. Specifically, in our case, we design and implement an HCI application. It supports both basic interactive functionality such as input and output and designed specific task which needs to be completed by the user. A friendly GUI is important in this application in order to achieve better interaction. For software development, an integrated development environment (IDE) with a built-in code editor, building tools and a debugger is absolutely necessary.

3.3 APIs

To support all the functionalities in the first the application layer, libraries and game engines can be quite helpful. Application programming interface (API) defines communication rules among software components in the application. The following APIs might be useful for the development.

3.3.1 JUCE

JUCE is a C++ library for developing GUI as well as dealing with text editing, graphics and sound. One of the excellent features is that a developer does not need to include many third party libraries to achieve functionalities. Tools for graphics, audio as well as animation have already been included. That is why it is alone capable of creating highly customized user-interface. JUCE project can be customized, created and managed using its own integrated development environment (IDE).

3.3.2 Box2D

Box2D is a game engine in C++ which simulates movements of two-dimensional bodies such as balls and squares. Box2D is a powerful tool if the game or specific task on the computer is in two-dimensional space. Gravity, friction and restitution are available in the engine. Collisions can be detected and simulated among the bodies. A famous game which is built upon this 2D game engine is Angry Birds.

3.3.3 Unity3D

Unity is another game engine with an IDE built inside. It is a powerful rendering engine to create interactive contents in a three-dimensional environment. A strong developer community exists for Unity which provides numerous asset packages in the Unity Asset Store. As a development environment, it is capable of incorporating third party libraries in the form of asset.
3.3.4 Leap Motion API

The Leap Motional Controller is a device which detects and tracks hand and finger movement. Its API is designed to support multiple languages such as C++, C# and Java. The Leap sees objects in its own view which is a pyramid upside down above the device. The API provides developers with access to the location of the hands, fingers and several gestures.

3.4 Hardware

The hardware part includes a computer with Universal Asynchronous Receiver/Transmitter (UART) ports and a monitor, a Leap Motion hand tracking device, a microcontroller, a driver for linear resonant actuator (LRA) vibrator motors and the LRA motors. The connections between the hardware components are presented in Figure 5 which is originally provided by David Jun.

![Diagram of hardware modules connection]

Figure 5. Hardware modules connection

3.4.1 Computer

A computer provides the basic platform for IDEs for software development. The UART ports on the computer build the communication channel between the computer and the microcontroller. The monitor is for the display of the HCI application.
3.4.2 Leap Motion Hand Tracking Device

This hand tracking device can detect two hands with all ten fingers and track their movement in an upside-down pyramid space above it with high precision. It is used as an input method in the HCI application.

3.4.3 Microcontroller

A microcontroller receives the instructions from the computer and outputs pulse width modulation (PWM) signals to the drivers of the motors.

3.4.4 Driver for Motors

A driver is a chip used for driving LRA motors. It receives PWM signals from the microcontroller and output’s physical layer signals.

3.4.5 LRA Vibrator Motors

The motors receive the input AC signals and vibrate.
4. Case Study: 2D Tower of London

HCI technologies help games to create better user experiences. In turn, games are test benches for research of HCI. In this case study, a virtual Tower of London game is built using JUCE API, Box2D game engine and vibro-motor’s API. Figure 6 shows the game interface. And Figure 7 shows the complete system. The application includes several modules to support full functionality such as main application window, buttons to support user input by mouse, full game features of Tower of London, a timer, Leap Motion hand tracking system and a vibrotactile feedback module.

![Game Interface](image)

**Figure 6. Game interface**

![System Display](image)

**Figure 7. System display**
4.1 Hierarchy of the Game

Before the explanation of each module in detail, the hierarchy of the game is presented below in terms of classes.

The first layer is the main application window realized by “MainAppWindow”. The second layer is the layer where all the contents in the main application window live. It is implemented by “Box2DDemo”. The third layers contains contents such as buttons, the timer and the control and display of game entity. They are achieved each by “juce::TextButtons”, a string and “Box2DRenderComponent”. The fourth layer is the game entity by “SimpleBrick”. Figure 8 displays this hierarchy.

![Hierarchy Diagram]

Figure 8. Hierarchy of the 2D Tower of London in the software part

4.2 Main Application Window

The main application window is a dialogue window with quit, minimize and maximize buttons. It supports the user to interact with the application via different input methods such as mouse and keyboard.

4.2.1 Multiple Levels of Inheritance

Figure 9 shows the multiple levels of inheritance in the JUCE API.
Figure 9. Multiple level of inheritance in JUCE

“TopLevelWindow” is used to create the major part of an application which is usually on top of the desktop. A “ResizableWindow” is a “TopLevelWindow” with additional functionality of dragging and resizing. Then finally, a “DocumentWindow” is used widely by developers to create a window with a title bar and multiple buttons for maximizing, minimizing and closing, similar to most of the windows we see in other applications.

4.2.2 Component in JUCE Library

“Component” is the base class for all JUCE user interface objects. It is inherited by almost all the other classes in the JUCE library which include classes for buttons, sliders and images. The main application window is like an empty canvas. If other objects need to be created inside the main application window, they should also inherit from “Component” so as to become child components of the main window. Two major pure virtual functions need to be overridden which are “Component::paint()” and “Component::resized()”. They are responsible for the display of the component.

4.2.3 Contents in Main App Window

In the game interface shown in Figure 5.1, we can easily notice that contents inside the main app window include a control panel and the display of game entity. The contents of a “DocumentWindow” get registered as the child component of it by the function “ResizableWindow::setContentOwned.”
However, these contents are achieved inside a created class named “Box2DDemo” which becomes the child component of main app window directly.

4.3 Control Panel

In the control panel, there are two major interactive components which are two buttons and a stopwatch.

4.3.1 Buttons

Buttons provide users with a basic way of interaction with the computer. In the 2D game Tower of London, there are two buttons in the control panel. One is “Start” button and the other is “New Game” button. When the user presses the “Start” button, the timer starts and the text on the button changes from “Start” to “Press Me When You’re Done”. When he moves the disks to a desired pattern and finishes the game. He presses the button again to stop the timer. If the user clicks the “New Game” button, the timer is reset and the locations of the disks are reset randomly.

JUCE provides developers with a variety of choices of buttons by different classes. They include “ArrowButton”, “ImageButton”, “TextButton”, etc. All of them inherit the base class “Button” which is also a “Component” in JUCE. In our case, “Start” and “NewGame” buttons are plain buttons with texts so we choose “TextButton” to create them. How a “Start” button is implemented is illustrated in Figure 10.

![Figure 10. Start button implementation](image-url)
Both two buttons are registered as child components of “Box2DDemo” and they are initialized its constructor. A button is a simple text box if its parent component is not registered as the listener of this button when it is clicked. Therefore, “Box2DDemo”, the class we create to hold all the contents, inherits not only from “Component” but also “ButtonListener”.

A pure virtual function in “ButtonListener” needs to be rewritten in “Box2DDemo”. Inside the function “Box2DDemo::buttonClicked()”, how other components will react to button clicks is written. The two buttons, the timer as well as the main game entity are all registered as the child components of “Box2DDemo”. Therefore, they can be easily accessed since their pointers are declared as private members in “Box2DDemo”.

4.3.2 StopWatch

In our game, the user needs a stopwatch in the control panel to tell him how much time has passed while he is playing the game. Also, a stopwatch is helpful for data collection if further experiments are needed. Technically, the stopwatch is implemented by a string displayed on the control panel. To keep updating the string as time increases, we need a system timer which provides repeated a callback function for refreshing. How the stopwatch is implemented is shown in Figure 11.

![Stopwatch Diagram](image)

**Figure 11. Stopwatch implementation**
In the JUCE API, there is not a class for graphic timer for our purpose of display and it has to be done in an indirect way. As is mentioned above, the stopwatch is created in “Box2DDemo” using a string which is quite simple for display. Two values are read to find the time elapse. The first is the time when the user clicks the start button. It is obtained by “Time::getMillisecondCounterHiRes()” which returns the number of milliseconds since the system’s startup. The second value is acquired by the same method. The difference lies in that it is called when the finish button is clicked. The time user takes to finish the game is the difference of those two values.

A system timer provides nothing but a repeated callback function. JUCE provides three types of timer which are “HighResolutionTimer”, “MultiTimer” as well as “Timer”. We choose the normal “Timer” because we need neither high precision nor callbacks at different frequencies. “Box2DDemo” is registered as “Timer” and it inherits a pure virtual function which is “Timer::timerCallback()”. It is rewritten to update the stopwatch string at the frequency of 60 Hz. Another its functionality is to update game entity that will be discussed later.

4.4 Game Entity

This module implements the game Tower of London with full functionality with the help of 2D game engine Box2D.

4.4.1 Game – Tower of London

Three poles are placed separately in the game scene. Three disks are stacked together and they are centered in one pole when they are initialized. Disks can be movable either by mouse or by user’s hand movement captured by Leap Motion device. Only the center area of the disk allows penetration when it is moved and placed on a pole. Collisions between disks and poles, disks and boundaries as well as collisions among disks exist. There are two game rules in our case. The first is that only one disk can be moved at a time. The second is that each move consists of taking the upper disk from one of the stacks and placing it on top of another stack (a disk can only be moved if it is the uppermost disk on a stack).

4.4.2 A 2D World in Box2D

A 2D world in Box2D is a virtual world with physics. Bodies are the basic objects in this virtual scene. Like an object in the real world, a body in a Box2D world holds many features such as mass, linear and angular velocity and position. All the game objects in the world of our game are defined in the class “SimpleBrick”.

14
Three types of bodies are available when a body is created which are static, dynamic and kinematic. In our case the first two types of bodies are used. A static body has does not move under simulation and behaves as if it has infinite mass [7]. A dynamic body can move under the influences of forces and collide with other bodies with finite body mass. There are seven bodies in total in the game world with one boundary and three poles being static and three disks being dynamic.

Box2D's collision detection and resolution system consists of three pieces: an incremental sweep and prune broadphase, a continuous collision detection unit, and a stable linear-time contact solver. These algorithms allow efficient simulations of fast bodies and large stacks without missing collisions or causing instabilities [7]. However, the collision does not happen among bodies. Instead, collision is one way that fixtures interact with each other.

A body is a skeleton if no fixture is attached to it. Fixtures are used to define the body's shape, size, density as well as regulate how a body reacts to collisions. One body may contain multiple fixtures. The boundary contains four fixtures which are left, top, right and bottom walls (b2EdgeShape). They construct an area where disks are restricted when being moved. For the poles, each of them only has one fixture with the shape of a rectangle (b2PolygonShape). As for the three disks, each is constructed by three consecutive rectangulars (b2PolygonShape) which are placed on the left, middle and right. The middle one allows penetration by being tagged as a sensor (a sensor allows two fixtures' collision with no response) so that the user could place the disk on the poles. Figure 12 shows how bodies in the world are constructed by fixtures.
The gravity inside this world can be defined arbitrarily big in an arbitrary direction by a 2D vector. The three disks’ movement is influenced by the gravity.

4.4.3 Move the Disks

In one of the previous chapters, we have presented the hierarchy of the game. "Box2DRenderComponent" instantiated a game world in Box2D of the class "SimpleBrick". The rules of Tower of London are also defined inside "Box2DRenderComponent". And it is the child component of "Box2DDemo".

There are two factors that determine the movability of a disk. The first one is whether there is other disk(s) above the one the user wants to move. Mouse joints are to put the objects inside Box2D under the force applied by the mouse. The second one is if a valid joint between the disk and mouse is created. We will later talk about how to substitute the mouse with Leap Motion.

'Box2DRenderComponent' can get access to locations of the objects in the game world by keeping a pointer of the world in its private member list. By dereferencing it, each disk can be accessed with ease. Through iterating the possible disks that contact the current one, we can get the relative position of
each disk pair (the current one and the contacting one). If the current disk is below any of the contacting disks, it is not moveable. If not, it is allowed to be moved. Similarly, when the user is selecting a disk either by mouse or by leap, this control interface is able to detect if a joint is created on this disk and mouse.

After the movability of the disk is verified, a mouse joint for dragging is created between mouse and the disk. As ‘Box2DRendererComponent’ is a subclass, it inherits mouse event functions from its parent class, ‘Component’. Three function callbacks are used when we implement the mouse drag which are “mouseDown”, “mouseDrag” and “mouseUp”. Conceptually, if a mouse-click-down event is made and detected, a mouse joint will be created after the movability of the disk is checked. Then it is programmed to detect the mouse-drag event which always happens after the mouse-click-down event. While the user is dragging, the object’s position will be updated repeatedly if a valid mouse joint of the object exists. Finally, when the mouse’s left button is released, the existing mouse joint will be deleted.

The process of detecting and dragging is presented in Figure 13.

Figure 13. Moving a disk in the game
4.4.4 Monitor Display and Game World

In the game world created by “SimpleBrick” in Box2D, it uses MKS (meters, kilograms, and seconds) units and radians for angles. However, our game is displayed on the monitor in terms of pixels, which causes a lot of trouble. For example, assume a display’s resolution is set as 1280 pixels by 800 pixels. The dimension of the game world we created is 32 meters by 32 meters. The point of mouse click and mouse move is detected in “Box2DRenderComponent” in the unit of pixels. However, the dragging movement happens in the game world in the unit of meters. Therefore, a transformation is needed. In our case, we use a built-in transformation in JUCE, Affine transformation.

4.5 Leap Motion Hand Tracking System

Before the description of the hardware part, we would like to credit Gabriel S. Hruskovec Gonzalez’s work in his ECE420 project. He integrates the Leap Motion and the vibrotactile, which provides us with great help.

The Leap Motion hand tracking module replaces the mouse while the player is playing the game. He moves his hand over the Leap Motion device to control his virtual hands to move the disks in the game world on the monitor.

4.5.1 How Leap Works

Leap Motion uses two CCD cameras and two infrared LEDs to acquire depth data. It uses USB2.0 to streams the data with maximum 290 FPs.

4.5.2 Hand Motion to Substitute Mouse Movement

With the Leap Motion API, the computer gets all possible information of the user’s hands and fingers by each frame. The Leap has a coordinate system with X, Y and Z axe. The hand’s position can be represented relative to the Leap. It is mentioned above that Leap Motion API provides easy access to some of the built-in gesture’s detection. A special gesture in our case is used to replace mouse clicks. And X and Y axe are used for movement in the 2D plane.

4.6 Vibrotactile Feedback

The vibrotactile feedback module provides the player with the feeling of vibration when he touches and moves the objects in the game world.
4.6.1 Arduino Microcontroller

Arduino UNO is used for receiving instructions from the computer and output PWM to the driver board, simply a PWM generator. We use UART port on the computer to send serial data to the Arduino where PWM function is generated by appropriate built-in functions.

4.6.2 Texas Instruments 2603 PWM Driver

The DRV2603 is a drive specifically designed by Texas Instruments to drive two types of vibration motors which are Linear Resonance Actuator (LRA) and Eccentric Rotating Mass (ERM) [8]. Here we choose its LRA mode. PWM signals input will output voltages with appropriate frequencies which can drive the LRA motors.

4.6.3 LRA Motors

Linear resonance actuator (LAR) motors are an alternative to ERM motors with better haptic performance characteristics and they are more efficient when working at resonance frequency [8].
5. Case Study : 3D Drag-and-Drop Game

In this case study, a three-dimensional AR scene is created. The user can see his own hands in the monitor to manipulate a virtual object and at the same time receives the sense of vibration from the motors attached to the fingers. This case study is developed in the Unity game engine incorporated with Leap Motion API as well as Vuforia API which is for augmented reality.

5.1 Motivation

As soon as the 2D Tower of London game is finished, a problem occurs that the user’s interaction with computer is not natural enough. When a human being uses his hand to proceed, touch and manipulate a real object in the real world, his hand and the object can be watched at the same time by his eyes. Based on the visual feedback, the movement of the hand is adjusted all the time and it is quite precise and natural.

However, the configuration of the 2D Tower of London has restrictions. The consistent visual feedback is not guaranteed in that case. This is because the real hand and the virtual hand locations do not overlap no matter how smart the mapping between the two is. The leap motion is put on the table above which the user moves his hand. Meanwhile, he has to see where his virtual hand is on the screen. Since the Leap Motion Controller only supports a certain area of hand detection and tracking, it is highly possible that the user’s hand moves out of the boundary of that area if not enough attention is paid to his hand. Figure 14 shows the problem.

![Figure 14. Distraction when the user plays the 2D TOL](image-url)
5.2 Hierarchy of the Game

There are two main modules in this application both of which live in the Unity game engine. The first is the AR module. It is used for the generation of virtual objects and deals with its position relative to the marker. The second is the Leap Motion hand tracking and movement module. It is used for hand tracking, display of the virtual hand as well as moving the virtual object. Both of them can be downloaded as packages inside Unity and can be imported into one project and get merged.

In terms of the hardware, it consists of one laptop, one USB camera, leap motion as well as a printed out image as the marker.

5.3 AR Module

This module is supported by Qualcomm Vuforia which is a software platform for both Android, iOS and Windows for the application to identify images which have been uploaded to Vuforia target manager online. Once the image is recognized, this AR module helps with the creation of AR effects [9].

5.3.1 A Sample AR Scene

A target or a marker is usually a real object such as an image which provides a coordinate system where the augmented virtual object is generated and placed. As the computer detects and recognizes the target, it will augment the reality with virtual contents. In our case we take an image as the marker. The image is loaded already in the computer. Once the computer sees the image, it will generate a virtual cube in the middle of the image. We use a USB webcam and use the computer screen as monitor.

5.3.2 Create an AR Scene

Two cores of AR are object recognition and placement of the virtual objects in the right coordinate system. Fortunately, they already have been done by Vuforia. In the Unity IDE, after creating a new project and importing the Vuforia package, we replace the original camera in the scene with the “ARCamera”. Then create an “ImageTarget” which is loaded with an image used for recognition. Finally put a virtual cube on the “ImageTarget”.

A point worth noticing is that an image needs to be uploaded to the cloud first and downloaded as a data package. Then load it to the “ImageTarget”.

21
5.4 Leap Motion Module

The module builds a bridge between the Leap Motion device and the game itself. It gets the position of the hand above the device and maps it to the location in the game scene. It deals with the rendering of the virtual hands. Finally, it regulates how the hand grabs and moves the virtual objects.

5.4.1 Scripts and Objects in Unity

This module creates an object in the scene named “LeapController” with two scripts attached to it. “The behavior of “GameObjects” is controlled by the “Component” that are attached to them. Although Unity’s built-in Components can be very versatile, you will soon find you need to go beyond what they can provide to implement your own gameplay features. Unity allows you to create your own Components using scripts.” [10] Scripts allow you to trigger game events, modify Component properties over time and respond to user input in any way you like. Scripts are defines the functionality of the object and how the object interacts with other objects in the game scene. One object can have multiple scripts.

5.4.2 Virtual Hands Display

The virtual hands display is implemented in the classes named “LeapUnityBridge” and “LeapUnityHandController”. In the “LeapUnityBridge”, it calls to keep updating on the input of Leap Motion in the function “Update()”. Also, based on the Leap’s input, “LeapUnityHandController” helps to represent the hand in the unity scene. Unity supports two languages for scripts, C sharp and JavaScript. All the classes mentioned above are written in C sharp.

5.4.3 Drag-and-Drop

Drag-and-Drop is implemented by the classes named “LeapFingerCollisionDispatcher” as well as “LeapUnitySelectionController”. “LeapFingerCollisionDispatcher” is a relative simple class that is attached to the fingertip objects. It gives feedback to “LeapUnitySelectionController” if collisions happen between any fingertip and any touchable game objects. Then “LeapUnitySelectionController” implements the rules for selecting, highlighting and moving of the objects.

5.5 Problems

In our case, the AR part works perfectly. The main problem is the virtual hand does not overlap the real hand, which is shown in Figure 15. Therefore, when the user is seeing his hand grabbing the virtual
object behind the monitor. He still needs to pay attention to his virtual hand because it is the virtual hand that actually moves the virtual object.

Another problem is that the game scene displayed on the monitor in real time is not fluent. The user’s real hand movement is quicker than the virtual one. The user has to move slowly.

Figure 15. Two hands’ mismatch in 3D Drag-and-Drop
6. Future Research

Future research will proceed in both software and hardware.

In the software part, efforts will be given in the better integration of Leap Motion into the augmented reality scene. Matching the virtual hand with the real hand requires deep understanding of the multiple coordinate systems in the project. Once the problem is fixed, we could develop the single drag-and-drop game to another Tower of London game in three-dimensional space.

In the hardware part, for simplicity, the current one choose to use an Arduino microcontroller combined with motor drivers to stimulate the vibrotactile actuators. Professor Minh Do’s tactile group has already built a FPGA based interface between the computer and the vibration motors which has the capacity to drive multiple motors at the same time. Efforts should be given to moving from the current circuit to the new one.
7. Conclusion

The incorporation of haptic technology as well as hand motion tracking is quite promising in the area of human computer interaction. Although no formal experiment is done using the 2D game we created, the initial demo is quite inspiring. The introduction of the haptic feedback as the third sensory channel besides visual and hearing provides us with another direction of research. The documentation of the platform from software layer to middle API layer to hardware layer describes one HCI system with haptic feedback thoroughly, facilitating the system's development.
References


