MASTERS PROJECT

LeapMotion for Kids

Author:
Christopher DESCH

Supervisor:
Dr. Jerry FAILS

Submitted in partial fulfilment of the requirements for the degree of Masters of Computer Science

in the

Department of Computer Science
Montclair State University

May 2013

MONTCLAIR STATE UNIVERSITY

Abstract

Department of Computer Science

Masters of Computer Science

LeapMotion for Kids

by Christopher Desch

The LeapMotion Technology presents a new way of interacting with a computer system alternative to the customary keyboard and mouse interface. The goal of the project was to discover whether or not an alternative, "hands off" interface has the possibility of being as successful and reliable as the keyboard and mouse interface. The purpose of this project/paper (?) is to describe the development process of building an application for this new interface in order to highlight the capabilities of the new interface. The application was designed by KidsTeam through cooperative inquiry techniques for the LeapMotion.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Jerry Fails for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Children of the KidsTeam for introducing me to the topic as well for the support on the way. Also, I like to thank the participants in my survey, who have willingly shared their precious time during the process of interviewing. I would like to thank my loved ones, who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love. . . .

Contents

List of Figures

List of Tables

Abbreviations

API	**A**pplication **P**rogramming **I**nterface
HUD	**H**eads **U**p **D**isplay
SDK	**S**oftware **D**eveloper **K**it
RAD	**R**apid **A**pplication **D**evelopment
USB	**U**niversal **S**erial **B**us

Chapter 1

Overview

1.1 Introduction

For decades, the keyboard and mouse have been the standard interface mechanisms of input for computer systems. Their legacy is apparent in the applications that have been designed with the purpose of maximizing their efficiency and effectiveness. However, as technology advances, new mechanisms have been created in the hopes of making interactions with computer systems less rigid. While the keyboard and mouse interface are less physically demanding on the body in order to carry out commands, they do not allow for an expressive range of motion. The keyboard and mouse are only expressive in two dimensions as they are limited to up, down or right, left movements of the mouse across a computer screen. The keyboard is even less expressive than the mouse as the user must use key strokes to carry out commands, thus making the mouse slightly more fluid in motion than the keyboard but not by much. With the advent of tablets and smart devices, the envelope had been pushed in terms of what could be considered ”effective” means of input for a computer system. These devices allow for a user to carry out a command with a simple stroke of their finger directly on the screen of their computer system, thus making the command more expressive than the traditional keyboard and mouse interface. The devices with touchscreen capabilities have opened the door for opportunities to create a fresh interface. This new mechanism would have the ability to replace the use of a keyboard and mouse while offering an even greater, more expressive range of motion to the user in a more ”hands-off” capacity, the term ”hands-off” referring to the fact that the keyboard and mouse interface as well as the touchscreen interface both require the user to literally be touching a device of some sort in order to carry out a command. However, certain interface technologies were created with the intention of never having to place one’s hands on a device.

With these ideas in mind, the project set out with the simple goal of developing an application for children by children using an interface with these specifications. The interface chosen was LeapMotion.

1.2 LeapMotion

The LeapMotion is a small device slightly larger then a USB drive which sits in front of a monitor and captures motion in 3 cubic feet of space using a pair of cameras. The small cameras triangulate the positions of hands, fingers and tools in their relative space between the LeapMotion and the monitor relaying accurate position and velocity data in real time. The data can then be used to control application by driving the user interface of the system [3].

This type of interaction with the computer system is different from the traditional keyboard and mouse because it does not require any physical contact with objects connected to the system and has the ability to sense a much wider range of input. The LeapMotion is also expressive in three dimensions as opposed to two.

In this paper and throughout the project we refer to a finger or tool interacting with LeapMotion as a ”pointable.” ¡fix?¿

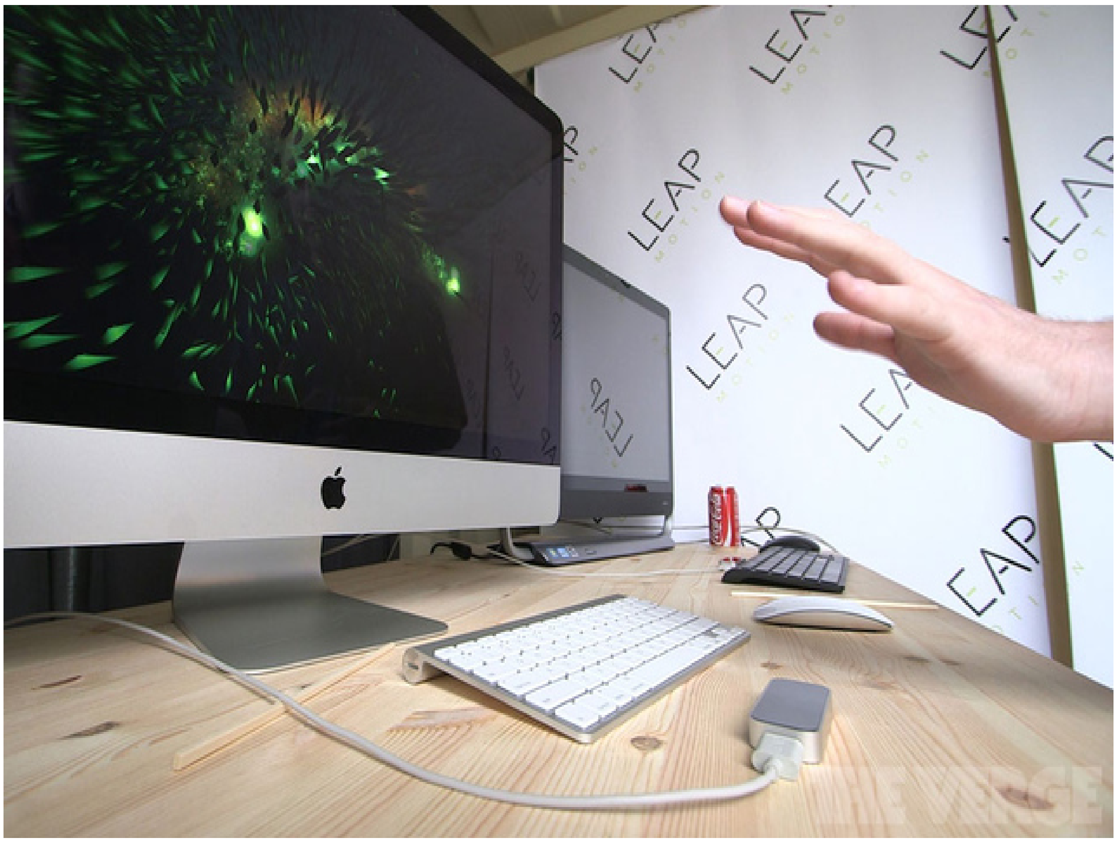

FIGURE 1.1: Interacting with the LeapMotion [1]

Chapter 2

Development Methodology

2.1 Development Methodology

This project differed from traditional software development projects in that it worked closely with children participating in the design process of the application. The project worked closely with a group called KidsTeam. The children affiliated with this particular group participated in the application's design process. Because of this collaboration, the project's development process was different from the traditional projects in the way the phases were performed and required a very flexible model of development.

2.1.1 KidsTeam

The KidsTeam is a group of eight children ages 6 to 11, male and female, overseen by Dr. Jerry Fails at Montclair State University. The group met twice a week for one hour sessions over the course of a semester. During that time, the KidsTeam worked on various projects which were facilitated by Dr. Fails along with several undergraduate and graduate students. The objective was to use the children's natural capacity for divergent thinking in order to study and identify new ways of collaborative learning and development as a means of designing a fun educational game to help elementary and middle school aged students learn and practice basic math skills. In order to achieve

FIGURE 2.1: Reviewing the sticky notes using a spatial graph on a white board for easy comparison and analysis.

an optimal outcome, the professor and students created a community of respect and rapport. The purpose was to create a safe space where the children felt free to explore and share their ideas. Therefore, it was imperative that the atmosphere remain relaxed. This was enforced by rules that framed particular behaviors such as encouraging the children to speak when they have a thought, idea, or question rather than raising one's hand. The only strict requirement of the children during the sessions was that they must respect everyone, including the adults, in KidsTeam.

KidsTeam creates a dialog when designing and developing an application between the developers and the users by building off each others ideas. This intergenerational design team collaborates by exchanging ideas and giving opportunities to enhance every aspect of the application.

2.1.2 Collaboration Techniques

Several techniques were implemented in order to aid the development process. These techniques were chosen for their means to foster a community of cooperative inquiry where the children felt encouraged to contribute design ideas freely. [4][5]

2.1.2.1 Sticky Notes

Each child was given a pad of Post-It notes[1].During activities which required comment, as the children played with the cubes, the children would write comments on the PostIt notes[2]. These comments were categorized into like, dislike, or design idea and then stuck

[1]Small square of paper with adhesive on reverse side.
[2]One comment per sticky note.

to the white board. A facilitator would then organize the notes based on the category and the comment content to resemble a spatial graph where similar comments are grouped closer together, while outlying comments are spread farther apart. Comments relating to a specific function or component are arranged into the same row while the category of the comment will determine the column. At the end of the session the observer debriefs with the children about the session by reviewing the comments arranged on the white board and having the children comment about the session overall. This time allows for the observer and the children to summarize the session, gives the children extra time to comment and provide more specific feedback, and permits the observer to further clarify a child's reasoning for making certain comments. The result is a frequency analysis as seen in Figure 2.1 which feedback can be turned into specifications for the next design cycle. [6][7]

2.1.2.2 Bags of Stuff

Each child is given a Bag of Stuff that contains a variety of arts and crafts supplies such as Popsicle sticks, felt, construction paper, markers, etc. The children are then given a design concept by the observer and asked to use these materials to construct that concept. As the children build, they must explain in their own words how their concept works while the observer takes notes. [6][7]

2.1.2.3 Storyboarding

The children are split into groups. The groups were chosen at random and did not remain intact from session to session. Each group then receives one large piece of construction paper. They will use the paper, sticky notes, and markers to draw their particular game concept sequence of events. Any actions or changes in a scene must be drawn in the order that they occur. The children must use arrows to delineate the progression of events. In order to make sure their ideas are conveyed clearly, the children are asked to be as descriptive as possible while facilitators take notes as the children work. [6][7].

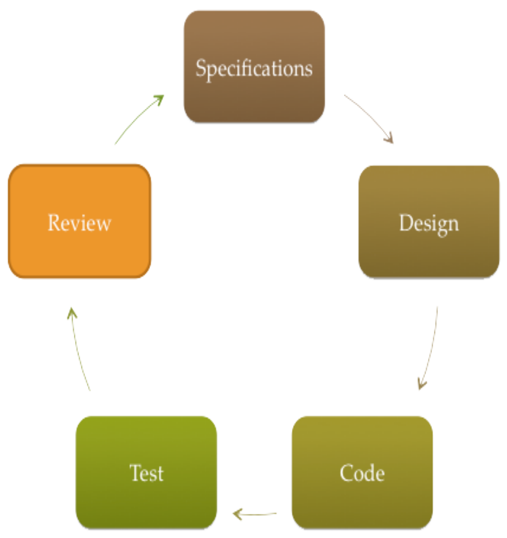

Figure 2.2: Rapid Application Development cycle

2.1.3 Development Model

Working with KidsTeam required a lot of flexibility in the development process and is why Rapid Application Development (RAD) model[3] for software development was chosen as the best fit model for this project. The RAD model uses rapid prototyping[4] and continuous iterative cycles which allowed for testing with KidsTeam on a weekly basis. Each session with the KidsTeam generated new requirements and fixes to be implemented within tight time constraints prior to the next session.

The model centered around five phases Specification, Design, Code, Test and Review as seen in Figure 2.2 constituting a full cycle. Changes in the specification were then implemented and reflected in the application prototypes ready for the next session. Each session with KidsTeam marked the end of current cycle and start of the next in the development process. [8][9]

The children were most heavily involved in the specifications, design, test and review phases in each of the nine sessions. Some parts of the design and testing were done independently of the children but mostly elaborated either on their designs and ideas or were performed to ensure the application was stable enough for testing with the children.

[3]Iterative or incremental development process resembling an evolutionary pattern.
[4]Rapid Prototyping is a quick mockup for testing

Chapter 3

Design and Development

3.1 Design and Development

The initial sessions centered around introduction of the LeapMotion exploring how it could be used with existing interfaces and designing new interfaces. In the later sessions the application and interface begin to take form and requirements will begin to solidify.

3.1.1 Session 1: Introduction and Brainstorm

This first session with the children started with brainstorming ways to use a computer system or control an application without a keyboard and mouse and only using their hands. Their ideas were made using the bags of stuff exercise. Afterward the children were allowed to play with a visualizer/footnoteThe visualizer 3d rendering of what the LeapMotion detects of the LeapMotion at the end of the session.

!!Flesh it out with the specific ideas that the kids came up with. Explain what that last sentence means.

3.1.2 Session 2: Testing Breakout and Designing a Paint Application

The children tested a simple BreakOut game which allowed them to interact with the LeapMotion with an application and get a feeling of how it works. With that experience the children were then tasked with designing a painting application that can draw, choose colors and brushes only using gestures to control the interface of their application.

Along with the interface the children came up with several gestures in controlling a variety of user interface controls but most of the gesture controls relied on techniques similar to the way a mouse would function in picking up, dragging and dropping an icons on the screen. Other designs required an action to be performed while pointing at the targeted user interface control indicating a beginning action or ending action[1] instead of waving of a hand, drawing a figure eight in the air or a slashing motion.

3.1.3 Gesture Design Challenges

The immediate challenge that faced gestures requiring beginning and ending triggers is how to interpret when each trigger takes place to perform the action. In the mouse and touch screen interfaces the typical design paradigm consists of a beginning action, intermediate action and ending action respectively representing the state in which the input is currently in.

Table 3.1: Apple Mac OS X and iOS Application Programming Interface for standard methods of input [2]

Action	OS X	iOS
Begin	MouseDown	TouchesBegan
Moved	MouseDragged	TouchesMoved
End	MouseUp	TouchesEnded
Alternate	–	TouchesCancelled

With the LeapMotion there was no standard way of detecting when each action is to be performed. With this challenge the children came up with the concept of using two

[1] begin and end actions

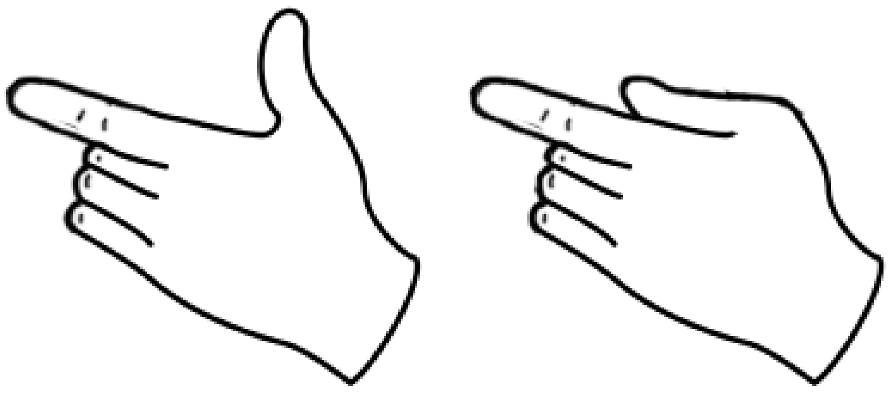
(a) Open Thumb (b) Closed Thumb

Figure 3.1: Hand trigging actions

fingers to indicate when an action would be performed. Using the thumb as the control mechanism to indicate the action state to be performed and the index finger to indicate the targeted interface control to act upon, the interface control could triggered based on its functionality. The gesture consisted of pulling the thumb flush to the hand while pointing at the user interface control to begin the action and releasing the thumb to indicate the end of the action.

With this simple system the a gesture could perform a BeginAction, MovedAction, and EndAction on user interface controls.

Use case:

- Icon Movement pulling the thumb flush to the hand would trigger the BeginAction "pickup" the icon and start dragging it on the screen. The icon could then be placed anywhere on the screen until releasing icon and "dropping" at that position by moving the thumb to a relaxed position no longer flush with the hand performing the EndAction.
- Selecting a color. The BeginAction would present a popover that would remain open while the user pointed at the desired color until the EndAction. The EndAction would select the color that was pointed at when triggered.

- Radio Buttons. A quick BeginAction and EndAction trigger while pointing at the radio button would change the state of the selected item.

3.1.3.1 Implementation Challenges

In concept this idea seemed to be an ideal and natural way of interacting with the system but faced several challenges. When the thumb became flush with the hand the LeapMotion could no longer detect it as a finger. This unique challenge was difficult because by the thumb could not be accounted for in all cases. An option might be to use the thumbs absence as a trigger although this is prone to many false positive[2] when the user's hands where on the boarder line of the LeapMotions visibility range or has lost track of the thumb due to occlusion[3]. Reversing the actions and performing the opposite gesture did not feel natural. Holding the thumb closely to the rest of the hand is not relaxed state and also did not mimic the act of grasping something or triggering an action as people are more commonly used to.

3.1.3.2 Unity Testing

Touching two or more fingers together rendered them not recognizable by the LeapMotion and thus would not allow pinch gesture to be used. Alternatives might be calculating when two fingers are close to each other although the LeapMotion cannot differentiate the fingers into which is the index, middle, ring, pinky, and thumb fingers. The application would have to assume any two fingers were performing the gesture and would not allow for the natural separation between fingers unless all fingers were kept folded tight into the hand until needed.

To observe this behavior a separate project was setup using the Unity 3D Engine to track the different pointables and their interactions in 3D space.[10]

It was clear after some lengthy prototyping and testing that some of the gestures developed by the children would not be possible and also raised some preliminary speculations

[2]False positive indicates a given condition is present when it is not.
[3]Occlusion occurs when the a surface is hidden by blocking the line of sight.

that the LeapMotion may only be supplemental in its role except when in application specific situations. Different methods of interacting with these types of controls would need to be developed or would rely on the keyboard and mouse for their functionality.

3.1.4 Session 3: Interface Controls

Common user interface controls were printed out and the children were tasked with brainstorming how they might use LeapMotion to control the widgets or design their own widget that can reproduce the functionality.

The user interface controls that we focused on included performing the following tasks were: buttons, drop downs, color palettes, check boxes, radio, buttons, sliders, toggles, steppers, wheels, trees and tabs.

The children did not come up with many new ways of interacting with the user interface controls or developing their own and through observation reinforced the suspicion that the may not be able to take a role as a dedicated device but a supplemental device.

3.1.5 Session 4: Testing a drawing application

Demonstration and hands on testing of a basic drawing application developed based on the specifications the children had designed in the previous. The children provided feedback with sticky notes.

3.1.5.1 HUD Requirement

One item that became apparent early on was the necessity for a Heads Up Display (HUD) which would track where the a pointable is pointing on the screen and also display and interaction with the pointable. Actions might include motion streaks or gesture animations to provide visual conformation of the action being performed.

3.1.6 Session 5: Designing Gestures

The children were given some gestures and asked to design a way of using the gestures with Microsoft Paint The simple gestures were turning, tapping, swiping, hand wave.

This session solidified Reinforced the idea that the LeapMotion cannot be a dedicated device but a supplemental device.

3.1.7 Session 6: Testing

The children were tasked with testing the painting application dubbed LeapPaint. While not testing they designed their own brushes, shapes and tools to be included in the next iteration of the tool. The overall consensus from this session was to implement better accuracy of the tool which required redesigning how the coordinate systems were translated from LeapMotion space into the application.

3.1.8 Session 7: Testing

The children were tasked with testing the painting application with the fixes based on the last session and providing feedback using the sticky notes technique.

3.1.9 Session 8: Testing

The children were tasked with testing the painting application

3.1.10 Session 9: Usability Comparison

We then compared the LeapPaint application with a three different methods of drawing with different interfaces. The children would attempt to draw the same picture of their choice on each system. Each child was given 10 minutes of time to attempt to reproduce the drawing.

Comparing a hand drawing using markers, Microsoft Paint using a mouse and keyboard, SimpleDraw on the iPad's touch interface and LeapPaint using the LeapMotion.

Chapter 4

Application Architecture

4.1 Application Architecture

¡Fix this ¿ Intro to Some Architecture stuff goes here

4.1.1 User Interface Layout

The interface layout of controls was based upon a combination of designs made by the children as seen in Figure 4.1 with the canvas to paint on centered and the user interface controls tucked to the sides of the canvas.

4.1.2 Pointable Tracking

Initially a tracking system was design to use the relative coordinates of a pointable in the LeapMotion coordinate space and translate them to coordinates within the application. Later with the an SDK update the LeapMotion could provide coordinates on the screen where a vector from the pointables tip would intersect. This required some screen calibration to be performed such that the LeapMotion would be able to track points on the screen.

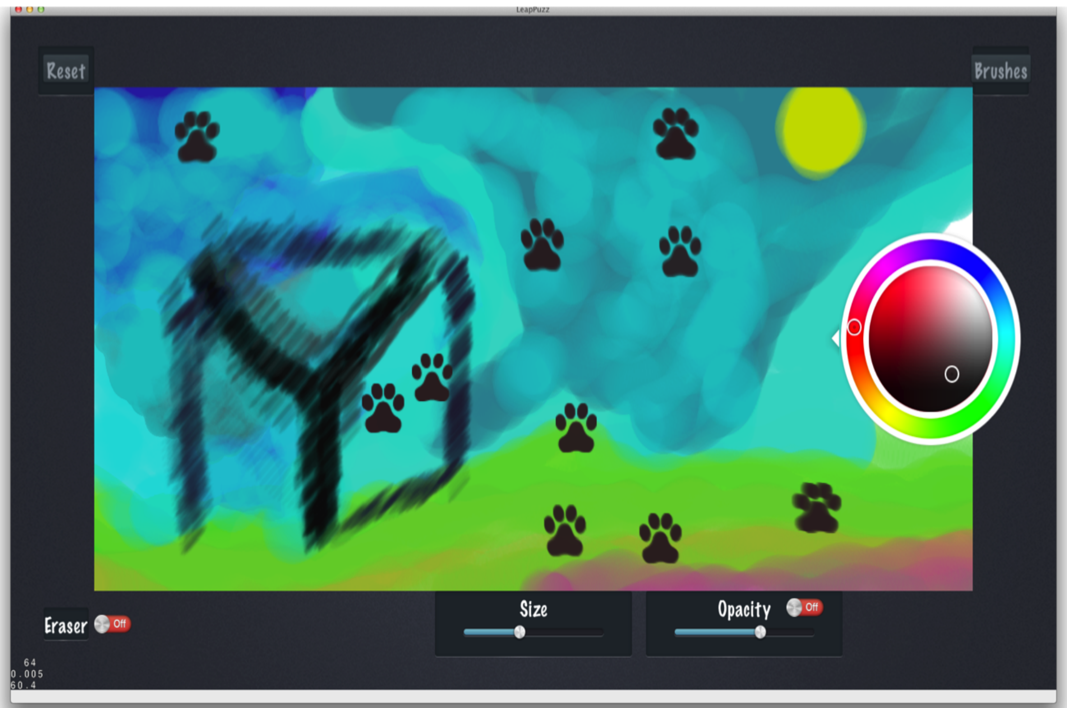

FIGURE 4.1: Screenshot of finished application with a drawing.

4.1.3 Application Layers

The application was broken into component layers to manage and modularize different functionalities as seen in Figure 4.2. This allowed for some components to become reusable for other applications by providing a common application programming interface (API).

- Background Layer manages the background images and for the application. This practice is fairly standard and
- Drawing Layer is where the image will be edited and render.
- Controls Layer provides the user interface controls that for different aspects of the application.
- HUD Layer displays feedback information for the position of the pointable in relation to the screen and any different actions that might be taking place.

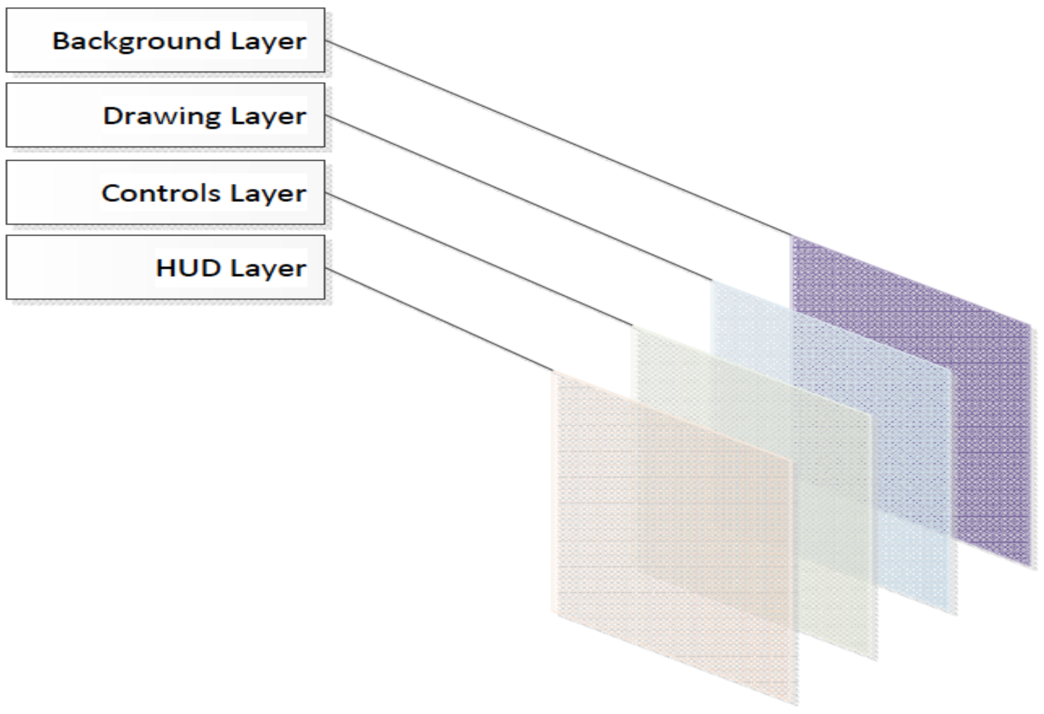

FIGURE 4.2: The ordered layers of visibility in the application.

4.1.4 External Libraries

This project was made with the help of several external libraries

- LeapMotion SDK is the provided API for using with the LeapMotion.[3]
- Cocos2d is a game engine framework that allows simple animation and control of sprites and layers. [11]
- CCControlExtensions are user interface control elements built for the Cocos2d Engine. [12]
- Graphics Recognition Toolkit was used initially for doing gesture recognition with customized pipeline until a LeapMotion SDK update provided the same functionality. [13]

4.1.5 Testing

The project uses unit testing on all the class functions to verify their input and outputs correspond to their intended function. Using the simple methodology of making the test fail and then making the test pass with a variety of inputs and outputs helped modularize class functionality into smaller and more reusable sections. The testing framework OCUnit tests at class and bundle levels to produce full code test coverage. [2]

Parts of the project unit testing could not cover were mostly involved at the interface level where the LeapMotion SDK provided data. Environmental factors effect the Leap-Motions performance when in a different lighting conditions. The different types and positions of lighting caused erratic effects that often had to be compensated for when noticed.

4.1.6 Documentation

Code documentation is done with in-line comments and then automatically generated with Doxygen. This was an essential tool due to the ever changing state of the project during each cycle that the project be able to automatically reflect changes in the design.

4.2 Experimental Features

Several experimental interface designs were added for the children to test with since there were often more than one design approach to a features in the application.

4.2.1 Drawing Modes

Two different drawing modes were tested with the children. The first was a mode where the pointable would begin drawing when crossing a boundary on the Z axis toward the screen. The cursor could then be moved about the screen without interacting with the drawing by pulling the pointable back and then pushing forward when ready to begin drawing. A ring around the cursor icon would indicate weather or not the cursor would begin drawing based on the depth of pointable. The ring would change from red when not in not drawing state to green when beginning to draw.

Further development might include a yellow color ring indicator when approaching the threshold to transition states.

Reviews. The children did not like this option as they had trouble becoming accustom to the threshold in which the application would begin drawing and stop drawing. They did like that they could draw and change colors at the same time by using their free hand with the mouse to create continuous rainbow effects with the brush.

The second drawing mode was to begin drawing when pressing the space bar on the keyboard and disregarding the depth of the pointable. This proved to be the favorite method of input for the children to indicate their actions.

4.2.2 Depth Opacity

Another option explored was using the Z axis to control the opacity of the brush. The intent was to provide a pressure sensitive brush which would mimic drawing implements in the real world where pressing harder on a paint brush or marker will draw a darker or thicker line.

The children did like this feature although did not take advantage of it very often.

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] David Pierce. A look inside leap motion, the 3d gesture control that's like kinect on steroids, June 26 2012. URL http://www.theverge.com/2012/6/26/3118592/leap-motion-gesture-controls.

[2] Apple Computer Inc. Mac OS and iOS API, 2013. URL http://developer.apple.com.

[3] Leap Motion Inc. Leapmotion, April 2013. URL http://www.leapMotion.com.

[4] Stacey D. Scott, Regan L. Mandryk, and Kori Inkpen. Understanding children's collaborative interactions in shared environments. *J. Comp. Assisted Learning*, 19(2): 220–228, 2003. URL http://dx.doi.org/10.1046/j.0266-4909.2003.00022.x.

[5] Vanessa Colella, Richard Borovoy, and Mitchel Resnick. Participatory simulations: Using computational objects to learn about dynamic systems. In *Proceedings of ACM CHI 98 Conference on Human Factors in Computing Systems*, 1998.

[6] Allison Druin. Cooperative inquiry: developing new technologies for children with children. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 592–599, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303166. URL http://doi.acm.org/10.1145/302979.303166.

[7] Allison Druin. The role of children in the design of new technology. *Behaviour and Information Technology*, 21:1–25, 2002.

[8] Shari Lawrence Pfleeger and Joanne M. Atlee. *Software Engineering: Theory and Practice (4th Edition)*. Prentice Hall, 2009. ISBN 0136061699. URL

http://www.amazon.com/Software-Engineering-Theory-Practice-Edition/dp/0136061699%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0136061699.

[9] Nayan B. Ruparelia. Software development lifecycle models. *SIGSOFT Softw. Eng. Notes*, 35(3):8–13, May 2010. ISSN 0163-5948. doi: 10.1145/1764810.1764814. URL http://doi.acm.org/10.1145/1764810.1764814.

[10] Unity Technologies. Unity SDK, 2013. URL http://unity3d.com/.

[11] Community Project. Cocos2d game engine framework. URL http://www.cocos2d-iphone.org/.

[12] Yannick Loriot. Cccontrolextension. URL https://github.com/YannickL/CCControlExtension.

[13] Nick Gillian. Gesture recognition toolkit. URL https://code.google.com/p/gesture-recognition-toolkit/.