3D Canvas Using Kinect And 3D Printer

By

Fahd Arshad (2010-NUST-BE-SE-232)

Fauzan Raza (2010-NUST-BE-SE-249)

Hassan Mehmood (2011-NUST-BE-SE-218)

Project documentation submitted in partial fulfillment of the requirements for the degree of ***Bachelors in Software Engineering (BESE)***

Department of Computing

NUST School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Islamabad, Pakistan

(2015)

CERTIFICATE

It is certified that the contents and form of this report entitled **"3D Canvas Using Kinect And 3D Printer"** submitted by **Fahd Arshad (2010-NUST-BE-SE-232**), **Fauzan Raza (2010-NUST-BE-SE-249**) and **Hassan Mehmood (2011-NUST-BE-SE-218**) have been found satisfactory for the requirement of the degree.

Advisor: _____

(Mr. Shamyl Bin Mansoor)

Co-Advisor: _____

(Mr. Muhammad Muddassir Malik)

DEDICATION

To Allah the Almighty

&

To our Parents and Faculty

ACKNOWLEDGEMENTS

We are deeply thankful to my Advisor and Co-Advisor, Mr. Shamyl Bin Mansoor, and Mr. Muhammad Muddassir Malik for helping us throughout the course in accomplishing my final project. Their guidance, support and motivation enabled us in achieving the objectives of the project.

We are also thankful to Mr. Stephane Ginier for his help, in providing valuable information regarding his system and communicating with us through emails. His timely and valuable feedback was the setting stone for our projects success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With Hephaestus, we provide users with a 3D canvas to manipulate objects in. The user can use gestures to interact with the objects on the screen. No contact is required to make the shapes. The motion of the user's hands are detected by Microsoft's Kinect and this is translated into input for the brushes.

Hephaestus offers an array of brushes totaling 10. These include 'Inflate', 'Drag', 'Smoothen', 'Flatten', 'Paint' and 'Crease' to name some. Moreover, users can also select from 11 'skins' or textures. '3D' files, formats like '.ply', '.obj' or '.stl', may also be imported and altered.

Models or sculptures created with Hephaestus can be exported in '.ply', '.obj' or '.stl', format so that users can get their models printed with a 3D printer. The sculptures that are created can be brought from the digital world into the 'real world' and users can actually showcase or use these models.

Hephaestus can be used by people of all ages and having differing familiarity with technology due to its intuitiveness. Although, it is intended to be used as an entertainment product it can nonetheless, be used by professionals as a technical tool.The key objectives of this project are to deliver a system that can be used for sculpting, reads gestures as input and is able to save 3D printable files.

CHAPTER 1

INTRODUCTION

1.1 Importance

For our Final Year Project we wanted to focus on creating a product. Throughout our degree it had been emphasized that we should attempt to create value. Another consideration of ours was that we were greatly impressed by Microsoft's second offering of their Kinect, the v2. We wanted to explore this amazing device as all three of us believed it to be part of the future of technology. We hence, decided to create a product that was based around Microsoft Kinect.

Hephaestus was the kind of project that would make for an extremely interesting product due to the facts that it would be usable by a large number of people and that it would connect the physical world with the digital world. We envisioned people using our project in their homes and creating 'art' that was special to and had value to them. But that is not all, the users would actually be able to then touch and hold what they had created by printing whatever object they see on the screen. This was an all important part which actually took the project full spectrum, from the digital to the physical.

Once again, the main objectives of our project are to deliver a system that can be used for sculpting, reads gestures as input and is able to save 3D printable files.

1.2 Scope

The scope of the project includes implementing Kinect functionality and interfacing it with the rest of the application that has all the functions for sculpting. Broadly the project was divided into two sub-projects. One was developing with the Kinect and the second was implementing a customized sculpting application. Lastly, we created an interface design that would maximize usability when using Kinect.

Son of Zeus and Hera, husband of Aphrodite, Hephaestus is the name of the Greek god of sculptors, artisans, smiths and craftsmen. We decided to name our

project Hephaestus and it seemed apt because of the God-like powers it bestows upon the users. Hephaestus (our project) allows the user to sculpt digitally by using gestures.

Hephaestus is a web-based application which combines the power of Kinect with the wonder of JavaScript. Hephaestus runs in the browser and users just need an internet connection and a Kinect which has to be attached to a computer to use the application. The application has been developed in C# as well as JavaScript.

1.3 Report Organization

Throughout the report we shall expound upon the details of our project pertaining to areas such as literature review, methodologies, design, architecture and technical progression. For the consideration of the reader the report shall also provide detailed UML diagrams and screen-shots explaining particular aspects of the project.

CHAPTER 2

LITERATURE REVIEW

2.1 Kinect Overview

2.1.1 Project Natal

Kinect, a motion sensing input device developed by Microsoft is one of its kind web-cam style peripheral. It allows its users to interact with their consoles and computers without the need for using hands. This product was launched on June 1st, 2009 at E3 keynote for Xbox. The project was named as "Project Natal" which was launched for the first time. (8bitjoystick, 2009)

"In addition to premiering 10 exclusive new games, revolutionizing the way we watch TV, and making it easier than ever to connect to friends, Xbox also welcomed visionary filmmaker Steven Spielberg to introduce "Project Natal" and controller-free gaming." (8bitjoystick, 2009) (Grant, 2010)

When Project Natal was unveiled for the first time to public it was presented as a revolutionary new way to play, no controller would be required. The launch was focused on mainly attracting Xbox users towards a new breed of gaming. The gaming where you could move your hands, use gestures to interact with games was the focal point of the launch. "See a ball? Kick it, hit it, trap it or catch it. If you know how to move your hands, shake your hips or speak, you and your friends can jump into the fun. The only experience needed is life experience."

Project Natal was the first sensor which combined RGB camera, depth sensor, multi-array microphone and custom processor running all in one. It focused on responding to commands, directions and change in emotions in voice. Some of the core features included full body play, personalized play and off the couch play. All of these features focus on attracting its users to a more fun experience of gaming.

2.1.2 Kinect v1

On June 13, 2010 Microsoft announced the system “Project Natal” would officially be called Kinect which is something on the lines of connect. This Kinect was launched for a new and improved Kinect 360 which was built only to serve Kinect sensor as an input. This launch included a Microsoft developed “Kinect adventures” which was shipped with the new x-box. (Grant, 2010)

Figure 1: Kinect V1

On June 16, 2011, the official release for non-commercial use SDK was launched. Later that year the first commercial SDK was launched for companies to start development. By the beginning of 2012, apps made for Kinect were developed for over 300 companies which covered almost 25 countries. The SDK which was launched earlier that year included windows 7 version compatible drivers. The developers were able to build application with C++, C# or Visual Basic using Visual Studio 2010. Some of the features it included was the raw sensor stream, skeletal tracking, advanced audio capabilities and a details documentation available for Kinect.

2.1.3 Kinect V1.5

In May, 2012 Kinect for windows version 1.5 was released. Some of the features added to its already existing features were the introduction of Kinect Studio. An application that allows developers to record, playback and debug clips for users interacting with application. It also included new added languages that were not part of the earlier version. (Warren, 2012)

The version also included the "10 joint" skeletal system that allowed users to track head, neck and arms of a user. This was possible irrespective of the users' position standing or sitting.

2.1.4 Kinect V 1.6, 1.7 and 1.8

These version of Kinect was launched in gradual increments. The last of its increments was released on September 17, 2013. This was marked as the last SDK version of Kinect which was launched by Microsoft. This last release for 1.8 was September 17, 2013.

2.1.5 Kinect V2

The second version of the sensor was released in September 2014. The second generation model was distributed with all retail Xbox one bundles. Some of the improved features included motion tracking and voice recognition functionality from it earlier version. The field of view was made wider and the addition of high definition camera was launched. The ability to track up to six bodies at one time was also made possible. On July 15, 2014 a windows version for the new Kinect, "Kinect for Windows v2" was launched on 15, July 2014. The SDK 2.0 not only allowed users to develop for Kinect, it allowed Kinect enabled apps for windows 8 and 8.1. By the end of the year an adapter to make sure already existing Kinect for Xbox could be used for windows by introducing the adapter to convert and connect to PC using USB 3.0.

The latest Kinect was greeted with a lot of positive reviews. En-gadget praised Kinect for its improved motion tracking and login recognition (Peckham, 2013). Time Magazine described the device as "creepy, but equally sci-fi-future cool"

Figure 2: Kinect v2

The Kinect V2 introduced 6 different data sources which were improved from its earlier version. These included Infrared, Color, Depth, Body Index, Body and Audio. Infrared can be used for single frame or for long exposures with 2 bytes per pixel frame data. The Color frame have RGBA, BGRA and YUY2 formats supported in its frames. The buffer is in the form of array of pixels. The bytes vary between different file formats. Depth caters to 2 bytes per pixel whereas Body Index 1 byte per pixel.

The Kinect V2 came to the market with new and improved skeleton tracking from the previous version. In this release there were more joints were added to the system. In total Kinect V2 reads up-to 32 body segment from the human body. States were also improved in the latest versions. With open, closed and lasso as states for hands. Activities like eyes closed, mouth opened, looking away can now be detected. The level of user engagement and facial expressions can be recognized with the new Kinect.

Unknown	0	The state of the hand in unknown.
NotTracked	1	Hand state is not tracked.
Open	2	The hand is open.
Closed	3	The hand is closed.
Lasso	4	The hand is in the lasso state.

Figure 3: Kinect V2 Hand States (Microsoft, 2014)

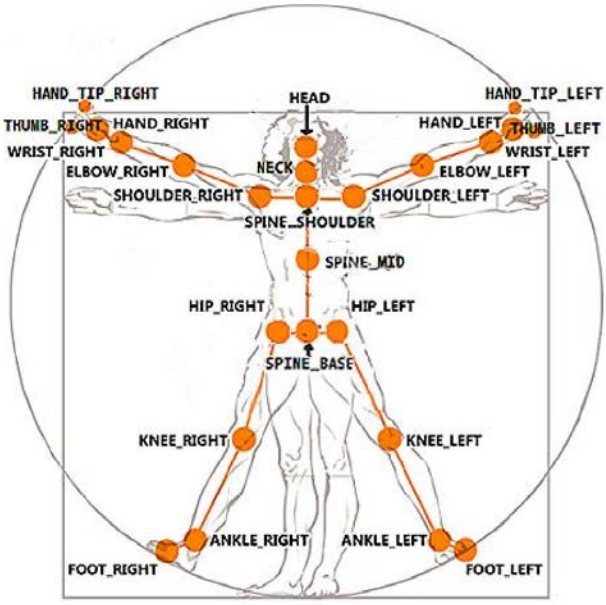


Figure 4: Skeleton Body Tracking (Microsoft, Tracking Users with Kinect Skeletal Tracking, 2012)

The main magic behind Kinect SDK is the machine learning which takes place in its core. The fact that 32 body skeletal segments can mean many possible body configurations. It also becomes a challenge when you have 30 frames per second. This was made possible by making use of Vision Algorithm. This was presented as a paper in CVPR 2011. This paper talks about Real Time Human Pose Recognition in Parts from a single depth image.

We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. Our large and highly varied training dataset allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes. The system runs at 200 frames per second on consumer hardware. Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton

nearest neighbor matching. (Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, 2011)

The Vision Algorithm allowed Microsoft to accurately predict 3d positions of body joints. Object recognition approach was take to make sure Kinect recognized the object in its view. Different body parts mapped to estimation of poses into simpler per pixel classification problem. There were large and highly varying data sets were used to estimate body parts irrespective of the pose, shape and clothing. (Jungong Han , 2013)

The sensor data model was Kinect v2 was changed immensely compared to Kinect v2. Two new models were added to the existing one, sources and readers. This meant that each type of input can be represented as a source. This can allow developers to open multiple readers for the same source.

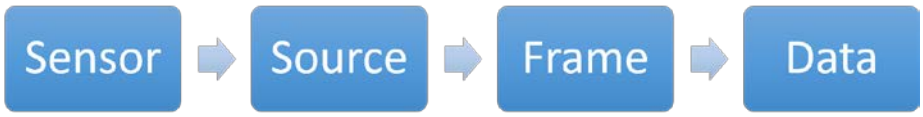


Figure 5: Kinect V2 Additions

2.2 Digital Sculpting

Digital Sculpting, which is also known as 3D Sculpting or Sculpt Modeling, is the use of software that offers tools to push, pull, smooth, grab, pinch and manipulate digital objects. This process revolves around the assumption that the object being manipulated is a real-life substance such as clay or other textures. Various application have different geometry for model manipulations which can provide benefits and limitations. Common practice of Digital Sculpting is to use mesh-based geometry. This geometry allows the manipulation of polygons using push and pull techniques. (Dylla, 2013)

Figure 6: Digital Sculpting Example

Over the years digital sculpting has become increasingly popular, it can still be taken as a relatively new method of sculpting compared to traditional sculpting techniques. This sculpting can often help sculptors create models which are otherwise difficult or impossible to create using traditional 3D modeling. This allows its users to create photorealistic and hyper realistic results.

2.3 3D Sculpting Applications

2.3.1 Z-Brush

This application was one of its kind back in 1999 when it was released in Los Angeles as a 2.5D painting app. It employed proprietary “pixol” that enabled artists to put detail into objects and paint them. The main difference between Zbrush and more traditional modeling packages is the fact that it is much more similar to sculpting. Zbrush allows the creating of high quality models which are used in movies, games and animations. It is known for its ability to help sculptors provide high to medium details to models. It was developed by Pixologic Inc founded by Ofer Alon. This applications comes with features that enable sculpting of models and meshes. Some of the application features include 3D brushes, Polypaint, Illustration, Transpose and GoZ. Initially the application provides 30 default brushes to be used by sculptors. (Pixologic, 2015)

Figure 7: Z Brush Screenshot

This can be regarded as one of the major applications that cemented digital sculpting as a viable method for modeling. ZBrush is a very powerful application, but it is often regarded as difficult to control because of its unconventional interface. The learning curve involved with this application is steep due to its workflow.

2.3.2 MudBox

Mudbox is a proprietary computer based 3D sculpting and painting tool which is being developed by Autodesk. This application can be seen as the biggest competitor of Zbrush. The developers of this application were part of the 3d team which were part of the Lord of the rings trilogy at Weta Digital. In 2007 Mudbox was released as a completed project and was acquired by Autodesk later the year. The primary application of mudbox is high resolution digital sculpting, texture painting and as a design tool. Some of its features include creating movable cameras that can be bookmarked, models manipulated with various tools, allows importing and exporting models in different file formats.

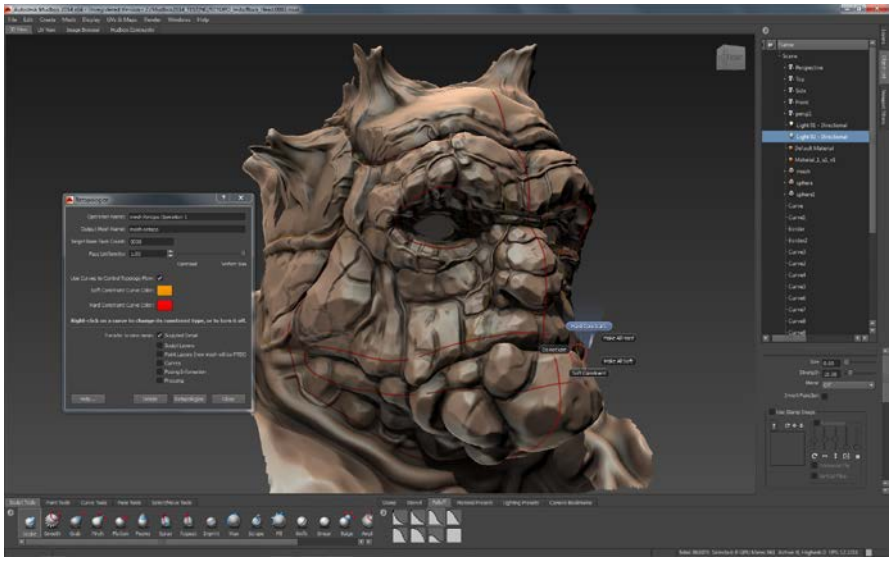


Figure 8: Mud Box Screenshot

MudBox takes a much more traditional approach to sculpting looking and working much like any other 3D application. The option to customize its interface is fairly limited which is something that can be difficult for someone to work with. However this application comes with a price of 425 £. (Autodesk, 2015)

2.3.3 Cinema 4D

"Turn your models into digital clay" (Maxon, n.d.)Is the moto for Cinema 4D, which introduced its sculpting module only recently. This was originally a 3D modeling and rendering application developed by Maxon Computer in Germany. In 2012 it added sculpting tools to its long list of features. Movies likes Iron Man, Tron and Spider Man 3 have used Cinema 4D. Cinema 4D comes with multi resolution sculpting, with advanced symmetry options which come along a range of brushes and stencils. The fact that it is a complete studio once you have sculpted your model you can render it with situ and smooth out irregular points.

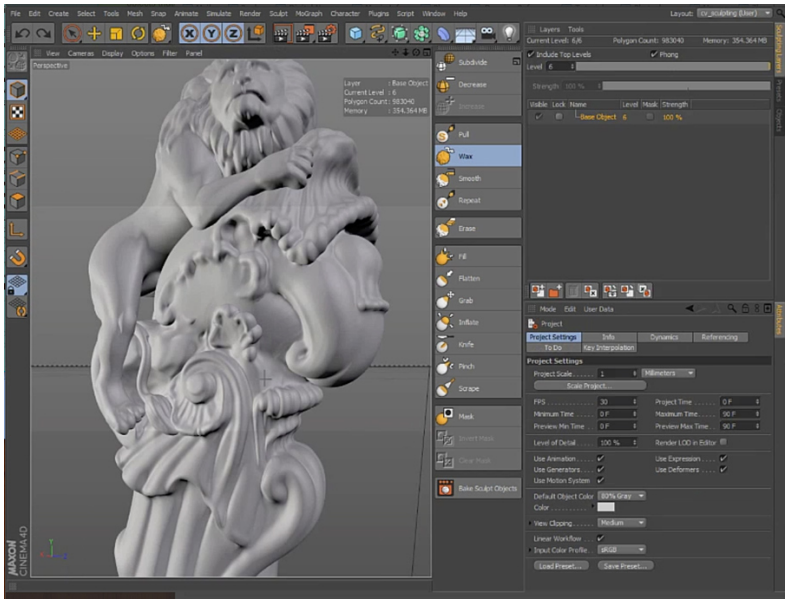


Figure 9: Cinema 4D Screenshot

Even-Though Cinema 4D can be regarded as a complete application it still lacks a painting application for sculptures. Although textures can be added to the model but painting not yet introduced. The high price also is a major disadvantage for people who wish to use the application for a small scale project.

2.3.4 Blender

Blender is a very famous software in the field. The application is mainly famous due to its open source nature. It gained sculpting tools in 2007, the system is similar to ZBrush and Mudbox which allows the user to change the geometry. The details of the mesh can be fine-tuned using the up resolution. Blender allows its users to freely sculpt and model objects without having to worry about the meshes that lie beneath the object. This clever system allows doodling away, adding or removing geometry at will. (Blender, 2014)

Figure 10: Blender Screenshot

Blender is a complete graphics application which has a very detailed sculpting panel for its users. Being an open source, free application users who are short on budget can work with these advances features. However for a novice user, the sculpting system is fairly complex. We tried using blender but it had a very steep learning curve with a great deal of meshes that only slowed down the sculpting process. The open source code is mainly in python and C++ but because of its vast details as a complete graphics application the code is very complex to understand and manipulate at will.

2.3.5 Sculptris

Sculptris is a virtual sculpting software program, with main focus on the concept of modeling clay. Sculptris can be seen as an alternative to Blender. It a very small application that was developed as part of a hobby code project by Tomas Petterson. This application proved so impressive that it was later acquired by Pixologic in 2010, with Tomas joining the programming team. The users can push, pull, pinch and twist the virtual clay. 3D meshes can be imported into the program for further detailing. (Pixologic, Sculptris, 2014)

Figure 11: Sculptris Screenshot

This application is very small sized with almost no lags when using it to sculpt models. The procedure it followed was connecting Zbrush via the GoZ Bridge. Unfortunately Sculptris is no longer being developed and is expected to be pulled out of service soon.

CHAPTER 3

FUNCTIONALITY AND DESIGN

Hephaestus provides basic functionality of sculpting. With commonly used tools you can almost sculpt anything you want. From basic shapes to complex structures our design is simple and can be used to sculpt, view and edit 3d-models. Basic functionality of Hephaestus includes brushes, intensity controller, radius controller, and some other tools which allow to sculpt easily. Some of the features are given below along with their detailed description how each was implemented and can be used accordingly. (Fahd, Hassan, Fauzan, 2015)

3.1 Sculpting Tools

3.1.1 BRUSH

Brush is the basic starting tool for sculpting which allows to add material on the object. Basically when you click on 3d-model using hand gestures it places some patches which can be dragged further, to sculpt on it.

Brush provide user with artistic touch to objects by providing the above mentioned functionality. Using kinect and hand gestures makes it more intuitive and highly effective using hand gestures which adds more fun using this tool. With the negative effect on, you can carve the object the same way as it appears on the picture above.

The functionality of this brush was obtained by selecting the brush value which was 0 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(0)

Figure 12: Brush Effect

3.1.2 INFLATE

Inflate allows to add materials on the object which is different than brush in such a way that it create bulges on the object. It is useful in achieving cylindrical effects. Inflate can be used to create pipes, draw bulges or increase the size on some part of 3d-model.

Inflate helps in creating round and cylindrical surfaces which can be seen in above picture. Enabling negative allows to achieve similar effects but in opposite direction, which in this case would be carving cylindrical shaping in the inner direction.

In order to implement this brush we have debugged code of this brush. We gained knowledge about its implementation and its functionality. By using different type of functions which were in the open source application we managed to utilize those in our application. Since the application is open source, we were able to identify the objective of each component. It was really hard to begin with, but as we moved forward we were able to find this brush and its usage.

The functionality of this brush was obtained by selecting the brush value which was 1 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(1);

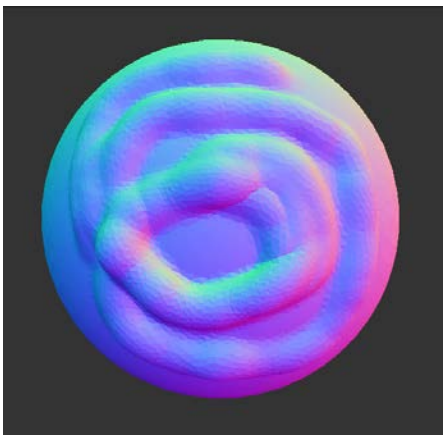


Figure 13: Inflate Effect

3.1.3 ROTATE

Rotate is pretty simple tool which rotates a certain portion on the model. It allows small tweaking on the complex part of the model and can be use to enhance the effects creating a realistic image. Right now we have this brush which only rotates in clockwise anti-direction so for an artistic this has to be used in the similar fashion.

With the help of Kinect and gesture you can easily achieve this effect. Effects of Rotate brush can be seen in the figure below

Rotation is necessary when you are sculpting. In order to get the idea about rotation we looked for the rotation brush in the open source app. The functions were very hard to understand at first because there wasn't any documentation with this open source app. we have to understand the logical part of rotation brush along with all its implementation. Now we were able to fully implement this brush. The code for this brush is mainly given in sculpt.js file.

The functionality of this brush was obtained by selecting the brush value which was 2 as mentioned in the sculpt.js

```
sculptgl.gui_.ctrlSculpt_.setValue(2);
```

Figure 14: Rotate Effect

3.1.4 SMOOTH

Smooth brush is very simple as it smoothens rough edges from the 3d-model. Most of time when you are creating models you need thing tool to help create model which are fine and smooth. This brush is mostly used in conjunction with inflate tool. Using Inflate and smooth allows to create model quickly. Below are the effects which can be achieved through this brush.

Smooth brush is an important brush. For this purpose we understood the logic and code behind it which was mentioned in sculpt.js file. The code is self-explanatory so it was easy to use it like other brushes. The function used was

The functionality of this brush was obtained by selecting the brush value which was 3 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(3);

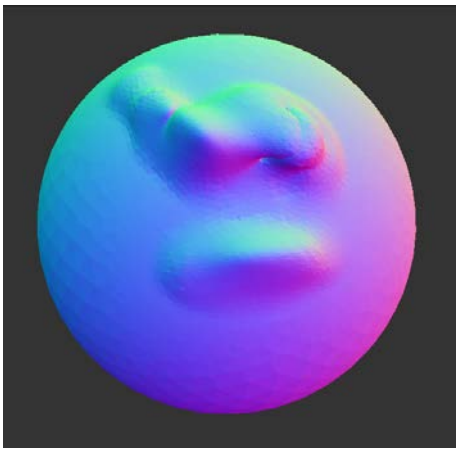


Figure 15: Smooth Effect

3.1.5 FLATTEN

Flatten brush is use to flat the edges. When you are sculpting sometimes you need to flat the surfaces of your 3d-model for that purpose you can use this brush. The difference between flatten and smooth is that flatten brush remove the material on your model whereas smooth brush simply smooth it without losing any material.

Flatten brush helps in creating flat surfaces. Using Kinect hand gestures you can easily flat surfaces. Effects of Flatten brush can be seen below

The functionality of this brush was obtained by selecting the brush value which was 4 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(4);

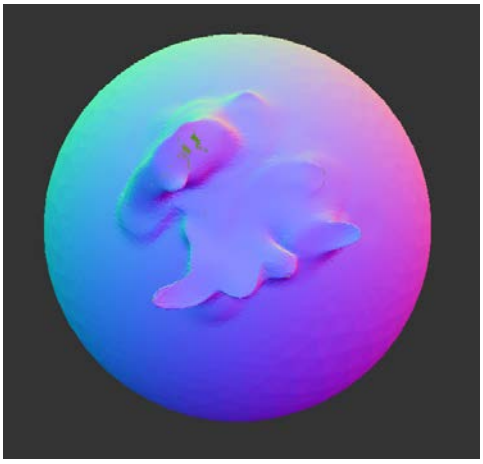


Figure 16: Flatten Effect

3.1.6 PINCH

Pinch brush is use to give the final touches to the 3d-model. This tool is very important as small tweaking are mandatory to make the objects look realistic. It provides a pinch like behavior in a circle which can be dragged further to increase its effects. A pinch effect is shown in the picture below.

The functionality of this brush was obtained by selecting the brush value which was 4 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(4);

Figure 17: Pinch Effect

3.1.7 CREASE

Crease is just like a line or pencil brush which allows to create a line structure over your 3d-model. This tool can be used to highlight the boundaries of your model which provide depth like look in your model. Moreover you can create paintings on 3d-model as it works just like a pencil. With the negative mode crease creates the opposite effects.

The functionality of this brush was obtained by selecting the brush value which was 6 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(6);

With Kinect you can easily create line arts in a similar fashion as on any other paint software. Effects of crease brush can be seen below.

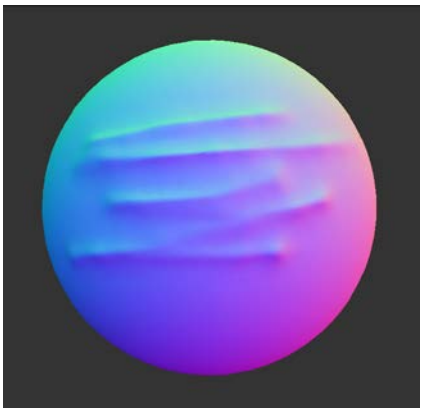


Figure 18: Crease Effect

3.1.8 DRAG

Drag brush is unique brush which helps to point things out from your 3d-model. This brush helps in creating complex textures like hairs or beard or other small pointing things. With very small radius you can create hairs and with very large radius you can create oval shapes. Kinect provide a real sculpting feeling when using this brush. Effects of drag brush can be seen below.

The functionality of this brush was obtained by selecting the brush value which was 7 as mentioned in the sculpt.js

```
sculptgl.gui_.ctrlSculpt_.setValue(7);
```

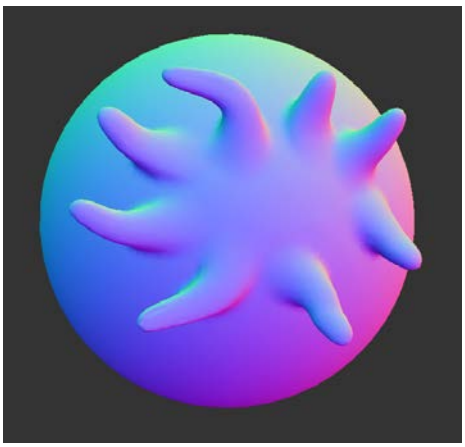


Figure 19: Drag Effect

3.1.9 PAINT

As the name suggests, this brush is use to paint your 3d-model. For painting you need to select the color from the color palette which gives values in the form of RGB. Once you select your paint color you can then easily paint. Effects of paint is shown below.

The functionality of this brush was obtained by selecting the brush value which was 8 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(8);

Figure 20: PaintEffect

3.1.10 SCALE

Scale brush scales things up and down. With Scale brush you can increase or decrease the size of particular structure in a circular region. This region can be increased using the radius. Scales helps in creating prominent structures, parts etc. This tool can be very helpful on small and hidden places in your model. Effects of scale is shown below.

The functionality of this brush was obtained by selecting the brush value which was 9 as mentioned in the sculpt.js

sculptgl.gui_.ctrlSculpt_.setValue(9);

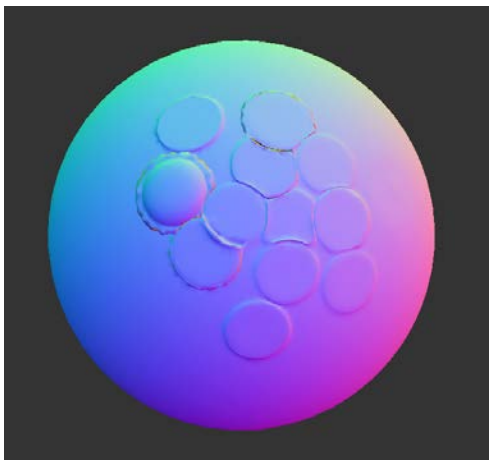
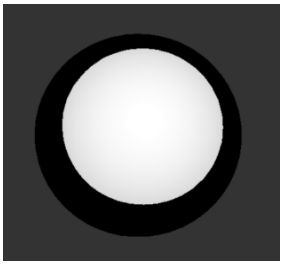


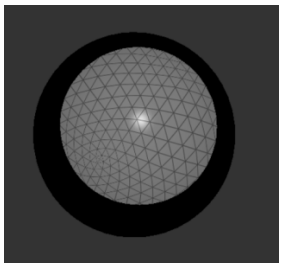
Figure 21: Scale Effect

3.2 Type of Skin Options

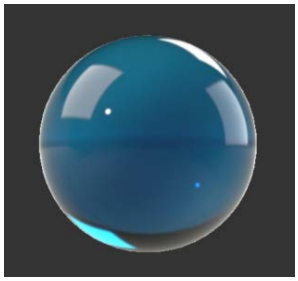
3.2.1 Phong



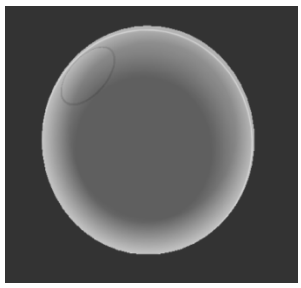
3.2.2 Wireframe



3.2.3 Drink



3.2.4 Transparent



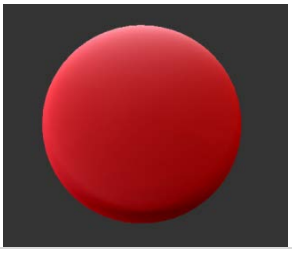
3.2.5 Normal Shader

3.2.6 Clay

3.2.7 Skin

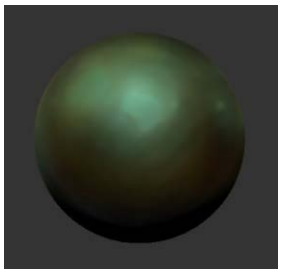
3.2.8 Chavant

3.2.9 Red Velvet



3.2.10 Orange

3.2.11 Bronze



3.3 Input controls

There were 8 different types of input controls which affects the input from the brushes. These controls were the core options which were available in most of sculpting software like Zbrush or blender. These controls holds the major part of sculpting. These are as follows

3.3.1 Intensity controller

This controller enhances the effects of any brush which is currently selected. For each clicks on your 3d-model the effects are greater which helps in creating models quickly. The controller can be used in the following way

```
document.getElementById("myint").addEventListener('change', function  (event) {
var slider = event.target;
document.getElementById("range").innerHTML=slider.value;
sculptgl.sculpt_.intensity_ = slider.value;
//  alert(sculptgl.sculpt_.intensity_);
});
```

The original source code of this controller can be viewed in sculpt.js

3.3.2 Radius controller

Radius controls the size of the brush allowing effects on the bigger scale. You can change the radius from the setting menu. The minimum value for radius can be 5 and maximum value can be 200. If the radius the increased beyond limit the effects distort the model. Radius controller can be used as follows.

```
document.getElementById("myrad").addEventListener('change', function  (event) {
    var sliderRad = event.target;
    document.getElementById("range2").innerHTML=sliderRad.value;
    sculptgl.picking_.rDisplay_ = sliderRad.value;
  //  alert(sculptgl.sculpt_.intensity_);
 });
```

The original source code of this controller can be viewed in picking.js

3.3.3 Negative controller

Negative controller is just like axis controller. You can enable and disable it from the setting menu. It allows to sculpt in opposite direction. For example, with negative inflate you can sculpt in the inner direction of your 3d model like carving. Negative controller can be used as follows.

```
document.getElementById("s2").addEventListener('click', function(event)
    {
     //$('.ferromenu-controller').removeClass("open");
    // $.fn.ferroMenu.toggleMenu("#nav");
            if(isN)
            {
                $("#s2").css("background-color", "green");
                sculptgl.sculpt_.negative_=true;
                isN=false;
                 $('.settingsmenu').removeClass("hidden").addClass("show");
                 $('.content').toggleClass("transparent");
            }   else
            {
                $("#s2").css("background-color", "red");
                sculptgl.sculpt_.negative_=false;
                isN=true;
                 $('.settingsmenu').removeClass("hidden").addClass("show");
                 $('.content').toggleClass("transparent");
        }
    });
```

The original source code of this controller can be viewed in sculpt.js

3.3.4 Continuity controller

Continuity controller enable sculpting to be continuous. It saves the time of clicking and helps creating smooth models. Continuous controller can be used as follows.

```
document.getElementById("s3").addEventListener('click', function(event)
    {
      //$('.ferromenu-controller').removeClass("open");
     // $.fn.ferroMenu.toggleMenu("#nav");
    //sculptgl.gui_.saveFileAsOBJ();
                if(isC)
                {
                    $("#s3").css("background-color", "green");
                    sculptgl.continuous_=true;
                    isC=false;
                     $('.settingsmenu').removeClass("hidden").addClass("show");
                     $('.content').toggleClass("transparent");
                }
                else
                {
                    $("#s3").css("background-color", "red");
                    sculptgl.continuous_=false;
                    isC=true;
                     $('.settingsmenu').removeClass("hidden").addClass("show");
                     $('.content').toggleClass("transparent");
    }
    });
```

The original source code of this controller can be viewed in sculptgl.js

3.3.5 Symmetry controller

Symmetry controller allows the mirror effects on your 3d-model. Just like symmetry in geometry, you can use this feature to create eyes, ears and other effects without worrying about the distances. . Symmetry controller can be used as follows.

```
document.getElementById("s1").addEventListener('click', function(event)
    {
```

```
        if(isS)
        {
            $("#s1").css("background-color", "green");
            sculptgl.symmetry_=true;
            isS=false;
             $('.settingsmenu').removeClass("hidden").addClass("show");
             $('.content').toggleClass("transparent");
        }
        else
        {
            $("#s1").css("background-color", "red");
            sculptgl.symmetry_=false;
            isS=true;
             $('.settingsmenu').removeClass("hidden").addClass("show");
             $('.content').toggleClass("transparent");       }
});
```

The original source code of this controller can be viewed in sculptgl.js

3.3.6 Undo controller

Undo controllers' helps to step back. The functionality is very limited because when you are sculpting, there are a lot operations involved which includes canvas redrawing. As you are sculpting the canvas is drawn multiple times. For this purpose, states of the model can't be stored for each canvas snapshot hence there is limited functionality of undo. Undo controller can be used as follows.

```
document.getElementById("myundo").addEventListener('click', function  (event) {
    //alert("undo");
    sculptgl.onUndo();
    $.fn.ferroMenu.toggleMenu("#nav");
});
```

3.3.7 Redo controller

Redo controller works just like undo controller with limited functionality. Redo controller can be used as follows.

```
document.getElementById("myredo").addEventListener('click', function  (event) {
    //alert("undo");
    sculptgl.onRedo();
    $.fn.ferroMenu.toggleMenu("#nav");
});
```

3.3.8 Reset controller

Reset controller resets your model. Model are loaded again. You can use reset controller as follows.

```
document.getElementById("myreset").addEventListener('click', function  (event) {
    //alert("undo");
    sculptgl.resetSphere();
    $.fn.ferroMenu.toggleMenu("#nav");
  //  $('#backgroundopen').trigger('click');
});
```

3.4 Other controls

Other controls includes zoom in, zoom out, import and export. These are the controls which facilities the sculpting process. These features used as follows.

3.4.1 Export as STL

```
document.getElementById("stl").addEventListener('click', function(event)
    {
      //$('.ferromenu-controller').removeClass("open");
     // $.fn.ferroMenu.toggleMenu("#nav");
    sculptgl.gui_.saveFileAsSTL();
     $('.content').toggleClass("transparent");

      $('.filemenu').removeClass("hidden").addClass("show");
     // alert("brush");

    });
```

3.4.2 Export as PLY

```
document.getElementById("ply").addEventListener('click', function(event)
    {
      //$('.ferromenu-controller').removeClass("open");
     // $.fn.ferroMenu.toggleMenu("#nav");
     sculptgl.gui_.saveFileAsPLY();
     $('.content').toggleClass("transparent");
      $('.filemenu').removeClass("hidden").addClass("show");
    });
```

3.4.3 Export as OBJ

```
document.getElementById("obj").addEventListener('click', function(event)
    {
      //$('.ferromenu-controller').removeClass("open");
     // $.fn.ferroMenu.toggleMenu("#nav");
    sculptgl.gui_.saveFileAsOBJ();
```

```
        $('.content').toggleClass("transparent");
        $('.filemenu').removeClass("hidden").addClass("show");
      });
```

3.4.4 Zoom in

```
document.getElementById("myzoombutton").addEventListener('mousedown', function (event) {
      //alert("undo");
      sculptgl.camera_.moveZ_ = 1;
       if (sculptgl.cameraTimer_ === -1)
        {
          sculptgl.cameraTimer_ = setInterval(function ()
          {
            sculptgl.camera_.updateTranslation();
            sculptgl.render();
          }, 20);
        }
});
document.getElementById("myzoombutton").addEventListener('mouseup', function (event) {
      //alert("undo");
      sculptgl.camera_.moveZ_ = 0;
       if (this.cameraTimer_ !== -1 && this.camera_.moveX_ === 0 && this.camera_.moveZ_ ===
0)
        {
          clearInterval(this.cameraTimer_);
          this.cameraTimer_ = -1;
        }
});
```

3.4.5 Zoom out

```
document.getElementById("myzoomoutbutton").addEventListener('mousedown', function (event)
{
      //alert("undo");
      sculptgl.camera_.moveZ_ = -1;
       if (sculptgl.cameraTimer_ === -1)
```

```
        {
          sculptgl.cameraTimer_ = setInterval(function ()
          {
            sculptgl.camera_.updateTranslation();
            sculptgl.render();
          }, 20);
        }


});
document.getElementById("myzoomoutbutton").addEventListener('mouseup', function  (event) {
       //alert("undo");
       sculptgl.camera_.moveZ_ = 0;
        if (this.cameraTimer_ !== -1 && this.camera_.moveX_ === 0 && this.camera_.moveZ_ ===
0)
        {
          clearInterval(this.cameraTimer_);
          this.cameraTimer_ = -1;
        }


});
```

3.5 Design and architecture

3.5.1 WEB APPLICATION DESIGN

3.5.1.1 Development Language

As discussed earlier, we are using an open source web application so most of the development is in java script. The main functions which we have implemented can be viewed in index.html. (Ginier, 2014)

3.5.1.2 Development

The application consists of different java script files. The biggest challenge was to find the core of application and find the links to various objects used in js files. The design of sculptgl can be summarized in the following 6 java script files

- **Gui.js**

Gui.js file contains the elements related to the GUI part of sculpgl application. It uses dat.gui which is java script file for easy installation of menus to any web application. dat.gui can be viewed here (https://code.google.com/p/dat-gui/). Related examples of dat.gui can be viewed here (http://workshop.chromeexperiments.com/examples/gui/#1--Basic-Usage)

Gui.js contains the main sculptgl object. This object is manipulated based on the user interaction with dat.gui. For our web application, we changed the GUI so that it can be made Kinect friendly. The code for new GUI can be viewed in index.html. Some CSS files were added to change the layout.

- **States.js**

States controls the undo and redo functionality. It maintains the states of canvas whenever it is redrawn. As canvas is drawn multiple times so undo/redo functionality is limited.

- **Sculpt.js**

Sculpt.js contains the main functionality of brushes. The logic of different types of brushes is mentioned in this file.

- **Picking.js**

Picking.js contains the logic behind the radius and intensity controller.

- **Camera.js**

Camera.js contains the zoom in / zoom out functionality.

- **Sculptgl.js**

It is the main sculpting js which controls all other js.

3.5.1.3 Architecture Overview

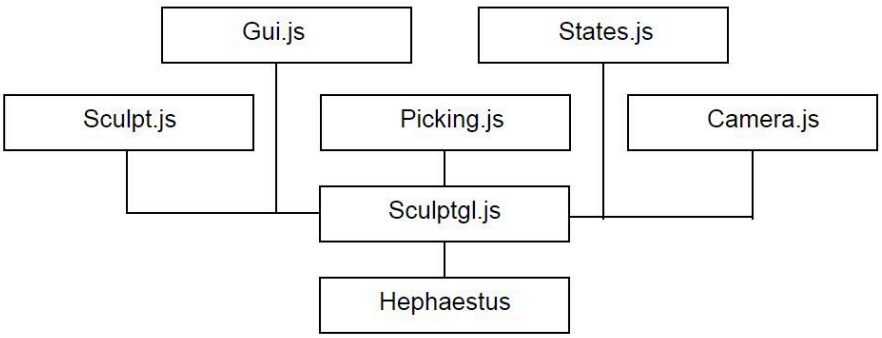


Figure 22: Hephaestus Architecture View

Main web application consists of major modules as shown in the figure. Sculptgl.js holds the 5 major modules

- Camera.js
  Contains the functionality of zoom in, zoom out and controls camera.
- States.js
  Contains the states of canvas for undo and redo operations
- GUI.js
  Contains the GUI elements
- Sculpt.js
  Contains the logic of different types of brushes
- Picking.js
  Contains the settings information regarding intensity and radius controllers etc.

Hephaestus uses the sculpgl.js object and manipulate this object according to the needs. New GUI features and user interactions are implemented using this object. Custom functions are given in index.html, which provide Kinect friendly user interface

3.5.2 Hephaestus Menus

Hephaestus contains one main menu and four sub menus. These menus were selected based on the usability and feedback from the students. These menus are shown below.

3.5.2.1 Main Menu

1. Brushes Sub Menu
2. Skins Sub Menu
3. Settings Menu
4. Import/Export Menu
5. Rest option
6. Re-do
7. Un-do

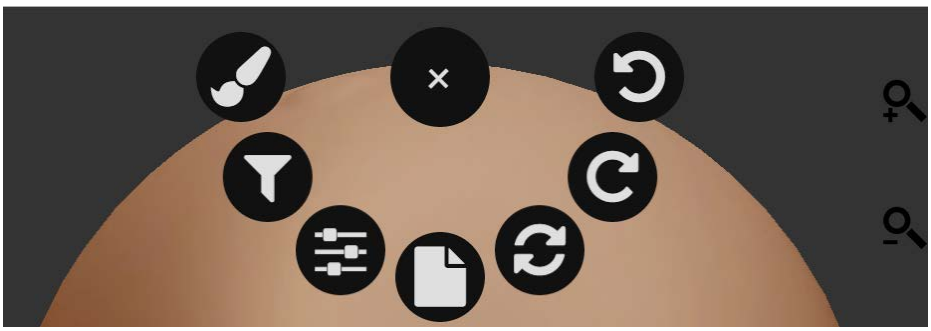


Figure 23: Hephaestus Main Menu

3.5.2.2 Brushes Sub Menu

1. Brush
2. Scale
3. Color
4. Drag
5. Flatten
6. Crease
7. Pinch
8. Rotate
9. Smooth
10. Inflate

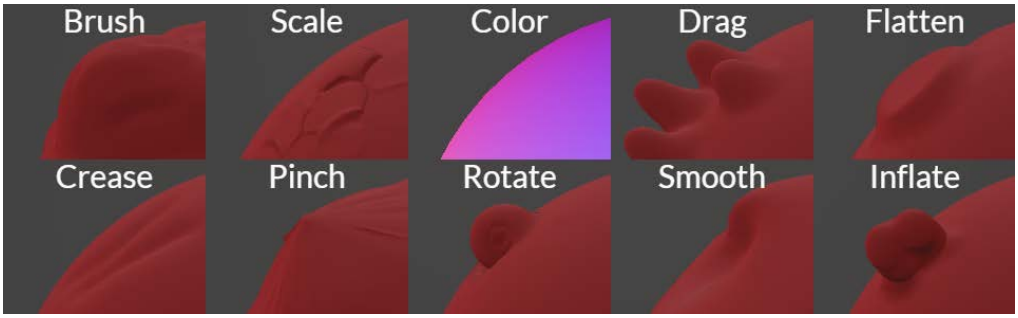


Figure 24: Brush Selection Menu

3.5.2.3 Skins Sub Menu

1. Chavant
2. Phong
3. Transparent
4. Red Velvet
5. Wireframe
6. Drink
7. Orange
8. Bronze
9. Skin
10. Clay
11. Normal Shader

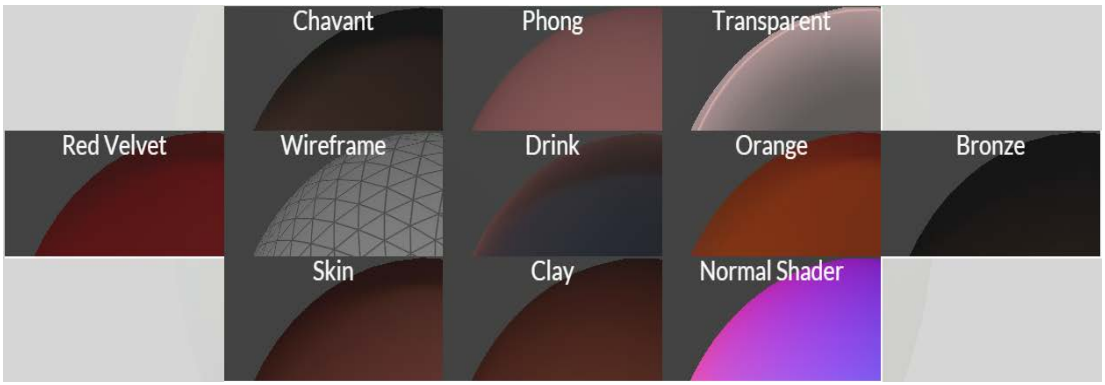


Figure 25: Skin Selection Menu

3.5.2.4 Settings Menu

1. Brush Radius Setting
2. Intensity Setting
3. Symmetry
4. Negative
5. Continuous

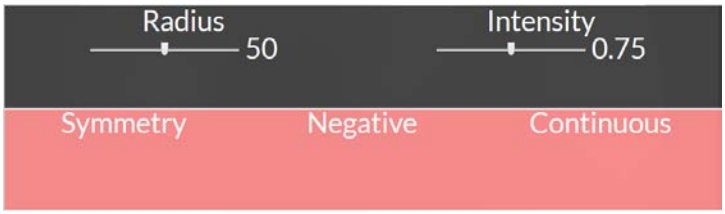


Figure 26: Setting Menu

3.5.2.5 Import/Export Sub Menu

1. Import Models
2. Export Models

Figure 27: Import/Export Menu

3.5.3 KINECT APPLICATION DESIGN

As we are sculpting using Kinect so we need to create a Kinect application to control the gestures and other hand movements. In order to do this we developed a smaller version of mouse control application as to make an interface between inputs from users to web base sculpting application. For that matter, we kept the design simpler by creating the following modules.

- Main window
  It contains 2 buttons which opens and closes the Kinect sensors. It also contains Kinect controls object.
- Kinect Controls
  Main functionality of Kinect Controls is to implement the gestures. Gestures are build using Kinect gesture builder. This modules helps to use to gestures and implement hybrid gestures. It also contains body drawer object.
- Body Drawer
  Main function of the body Drawer is to locate the body as seen by the Kinect sensor. Body drawer gets the co-ordinates of the camera whose center is at 0, 0, and 0. It also draws the pointer on the main window to test the values.
- Canvas Cod Mapper
  As the body drawer object gives camera points which are not according to the screen this function helps to convert camera points to screen pixels. In this way the resolution problem is solved.

3.5.3.1 ARCHITECTURE OVERVIEW

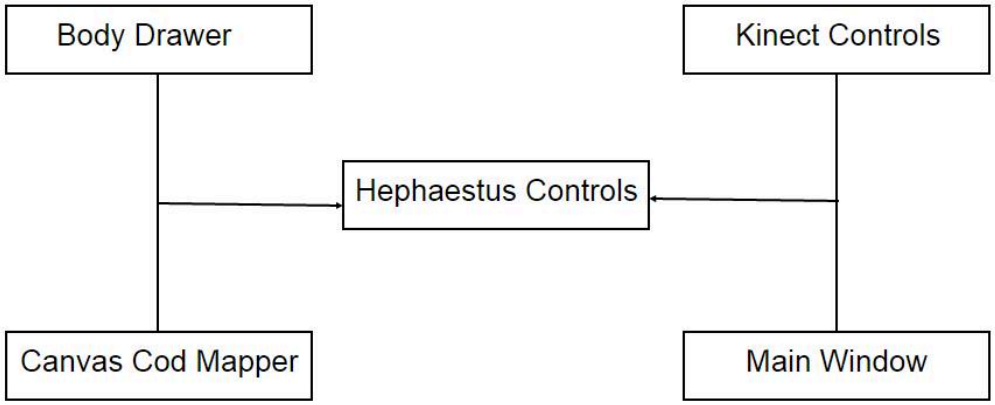


Figure 28: Kinect Architecture View

CHAPTER 4

IMPLEMENTATION AND RESULT DISCUSSION

4.1 Design Implementation

We implemented the proposed design by dividing the project into two smaller independent projects. The first part was to implement the sculpting application as a whole. The second part was to implement Kinect interaction for our project. By doing this we achieved maximum cohesion and minimum coupling in our project. Another thing that was achieved through this was that we focused our efforts on one part of the project at a time and this resulted in a project that can be extended quite easily.

The first part was completed by the mid-defense and we had a complete sculpting and 3D modelling application ready. This was a completely standalone system that worked without the Kinect as well and did not depend on a singular input device. A complete product on its own it could be used with a mouse and we tested it using the same. The three of us spent the first half of the project duration working on this part and divided the work regarding only the sculpting application. Our first task was to thoroughly understand the original code which we all did. After about a month we knew exactly what was going on in what part of the code. This enabled us to move forward in the project and helped us immensely.

When all of us had full working knowledge of the code we decided it was time to start experimenting with the original application. We wrote our own code, re-used functions from the original and generally played around with the code. When we had decided what all we wanted to include in our application we started to write our own version of the application. When we had completed what we had set out to do we tested it and found our application worked but it had a design flaw. While it was a complete sculpting application the interface that we had implemented was not viable to be used with the Kinect. So we had to change the interface later on to make it less intrusive and more Kinect friendly.

We moved on to the second part where we developed a Kinect interface to be used with our sculpting application. Initially, we wanted to implement gestures to be used with the application but we later found out that the number of natural gestures available were limited and we needed to implement heuristical gestures as well. So we used a hybrid of both types of gestures which made the application easier to use and the user input became more natural. This was found when we tested with the users after we had implemented the second part of the project and had also successfully integrated it with the first part. We also found during the testing that we need to change the interface and make it usable with Kinect.

At first we wanted to develop the Kinect functionality in JavaScript but we found that not a lot of resources were available for it so we decided to move to C#. There was a lot of material available for programming with Kinect v2 in C# and that helped us a lot. We also found that Kinect performed better when we wrote the application in C# and it did not lag at all. Compared to JavaScript this was an improvement and added another reason why we chose C#.

4.2 Implementation Challenges

We faced several challenges during the implementation for example, we had to figure out the best suited interface for use with the Kinect, we did not have a background in graphics, and we had to think up gestures to be used with the application. When we started the project we were unsure whether to implement a Windows application or a web application. After we had reached consensus about going with a web application we had to figure out how to go about it. We came up with the solution of splitting the project into 2 parts. The two applications were mostly independent and could be developed in multiple languages so we had options.

Another thing that we had to think about in the beginning was that we lacked any background in graphics. We did not take any course in it and also did not have any experience working with graphics. Our advisors advised us to look for an open source application to use as there was not much time to learn and then develop our

own sculpting application which would be able to manipulate and create 3D models. We looked for many solutions and chose SculptGL to adapt into a custom application for use in our project.

Being students of science and having no experience with any tool or software related to art or one used by artists we did not have a clue what existing sculpting applications looked like. This was something that needed considerable research as we wanted to develop a solution that would afford users the best experience. For this we looked around and tried to work with many available sculpting studios and applications. Some were small and freely available while others were complete suites which had support for other things like animation as well. Gradually, we figured out what the important elements of a sculpting application are and what were common in all the applications we tried. We made a list of that functionality and set out to implement it in Hephaestus.

The last challenge for us was to map gestures to our application. We needed to keep the interaction natural as well as achieve all the functionality without touching anything. There were many options that users had to use and we had to assign gestures to all of them and they also had to remain distinct. What we did was that we used a mixture of gestures and heuristics to cover everything. This let us access everything and also made for improved accuracy which was paramount.

Some key results that were found during the course of this project include findings about the platform and the use of Kinect with applications. We will touch upon them in the paragraphs that follow. Our first lesson was that implementing a web application whenever possible is a better option as opposed to creating a desktop application because of the number of people you can reach. We are grateful to our advisors to suggest using an open source existing application. This saved us a lot of trouble and made for an overall better product. Not only that we also developed working relations with the owner of the original application. This has also taught us to look for existing solutions and complete a thorough research before we actually start a project. Another aspect of research should be to map or chart out a way to complete the project. We had to evaluate and look for different

ways to implement the application like, the programming language to use and the platform to use.

We also found that users needed an interface design that was suitable with the Kinect. Some users initially complained about the small size of the buttons and this led to the improved design that is seen in the final version of the application. If we had not listened to them we would not have found this out. So it is important to listen to the users and to test your product repeatedly.

4.3 Result Discussion

Once we were done with the project, it was time for us to get feedback from actual users of the system. The application deals with a broad set of users, from novice users to people who know how to sculpt. The procedure we undertook in order to get our system tested and get valuable feedback was through a carefully designed questionnaire. This questionnaire was asked to be filled out by anyone who had tried out the system in hopes of making the system more easy to use.

In order to get a variety of users, the system was tested out in different locations. The first session was in School of Electrical Engineering and Computer Sciences (SEECS). The response from this session was good, but since most of the users lacked any formal education of art the models they made were pretty basic. The second Session was held at School of Arts Design and Architecture (SADA). The response from the students and faculty from SADA was overwhelming. They were excited to see the blend of technology with the basic forms of sculpting.

Both the sessions, started with an initial explanation of how the system works, followed by hands on experience by students and in the end a questionnaire was asked to be filled in. The question were constructed to get a view of the overall experience of the users. The ease through which they were able to operate Hephaestus and the amount of effort that was needed to put in, in order to get a decent output.

4.3.1 Sample Questionnaire

1) Your overall experience with Hephaestus was good?

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

2) Hephaestus's interface is user friendly.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

3) Hephaestus is easy to use.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□

Strongly Agree	□

4) The desired output was generated using Hephaestus.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

5) Interaction with Hephaestus was natural.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

6) You would like to try Hephaestus again if given the chance.

Strongly Disagree	□

Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

7) You would recommend your friends and family to try out Hephaestus.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

8) You would welcome similar technologies e.g. Oculus Rift, Leap Motion, Holo Lens.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

9) You would like to print the models you created with Hephaestus.

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

10) You would like to use Hephaestus professionally (if applicable).

Strongly Disagree	□
Disagree	□
Neutral	□
Agree	□
Strongly Agree	□

Comments:

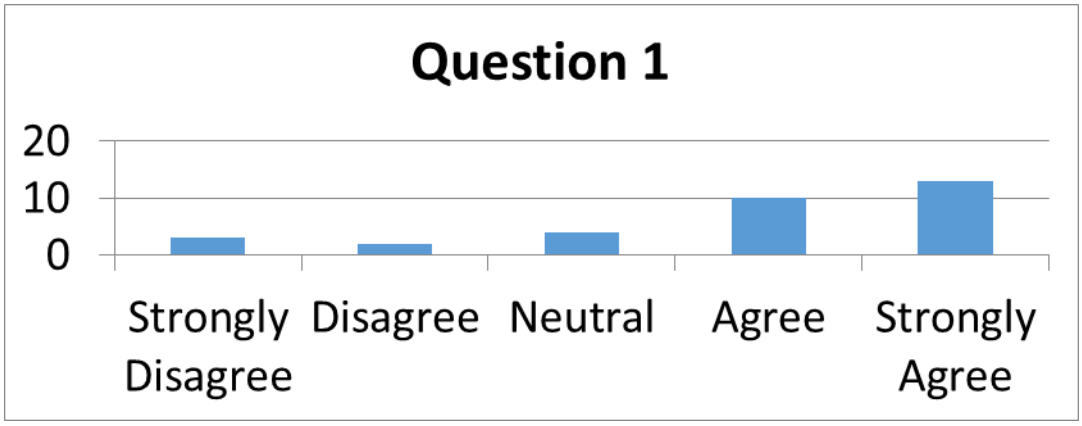
4.4 Result Comparison

The questionnaire was filled by a total of 34 people, from both SEECS and SADA. The results of some of the questions are summarized below. These are the questions that we feel were important for our analysis of the system and how the users perceived it.

4.4.1 Your overall experience with Hephaestus was good?

This was one of the most important questions which was included in the questionnaire. This question gives a general idea of the reaction users had towards Hephaestus. These reactions mainly ranged from Agree to Strongly Agree which was a good sign. This means that the overall reaction towards Hephaestus was fairly positive. The introduction of Kinect based sculpting is a new concept which has not been previously touched upon. The statistics of user experience of such form of sculpting can be seen in the figure below.

Table 1: User Experience Graph

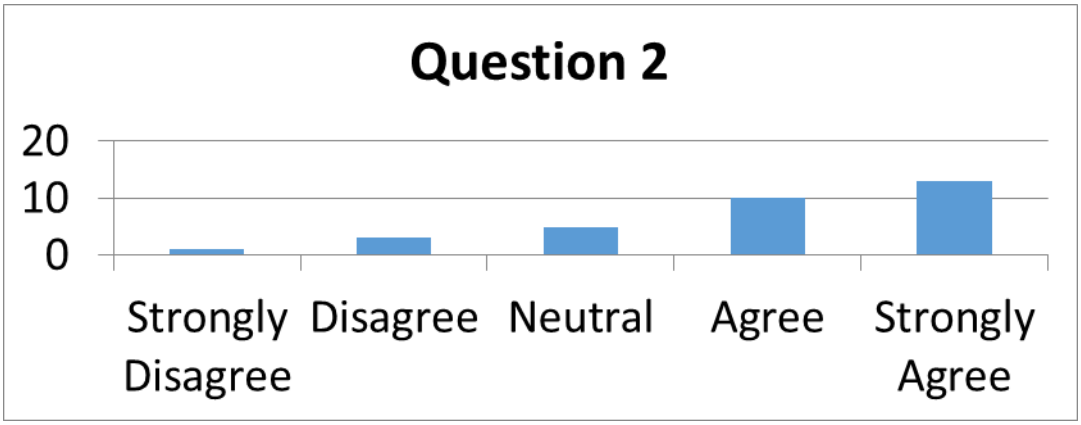


4.4.2 Hephaestus's interface is user friendly.

Interface is one of the most important thing an application consists of, the look and feel is essential. This question target user's experience of the interface. The interface was developed especially for Kinect based interactions. The users

were very happy to see the intuitiveness of the software. The gestures which were implemented for clicks and sculpting were very famous among the users. The statistics below illustrate the user response of the interface.

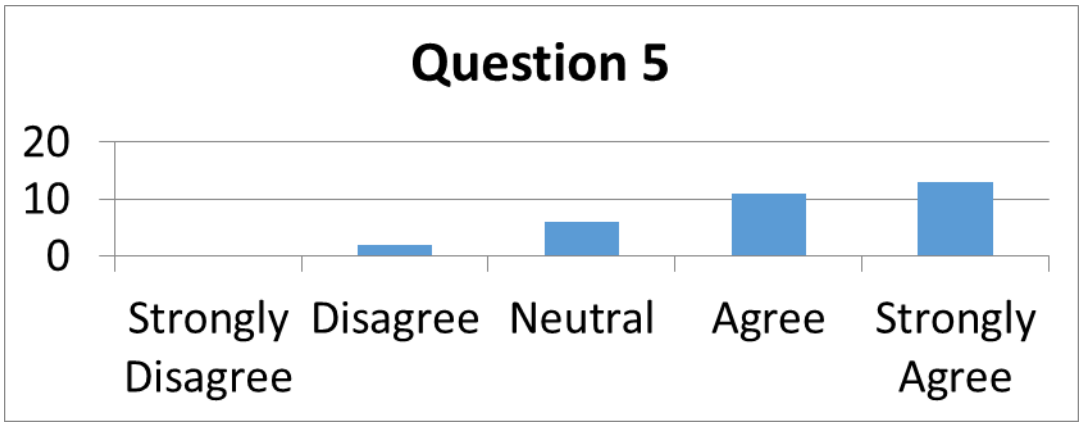
Table 2: User Friendliness of Interface Graph



4.4.3 Interaction with Hephaestus was natural.

The main purpose of Hephaestus was to provide a natural way of sculpting in 3D space. This is one of the most important aspect of the software. Sculpting without actually being able to touch is pretty difficult. We wanted to develop a software that would be more natural and intuitive for its users. Kinect based interactions require its users to user their upper body to interact with the 3D object. It is very important that the interaction is natural in order to make the experience fun. The results of the question can below. The overall response of interaction was mainly positive with a large majority of people agreeing to the natural-ness of the system.

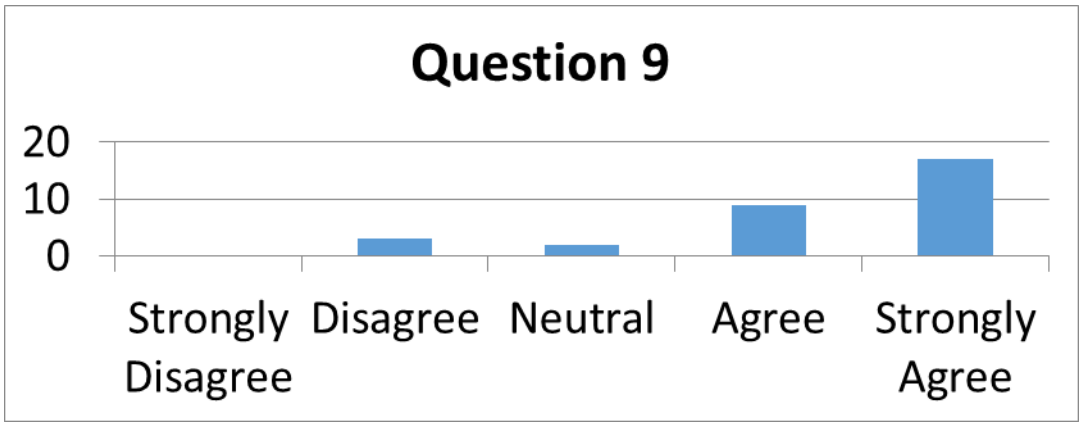
Table 3: Natural Interaction Graph



4.4.4 You would like to print the models you created with Hephaestus.

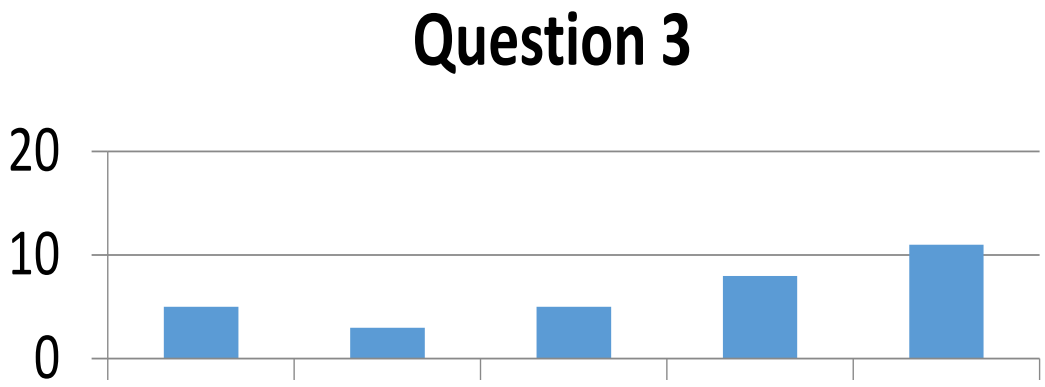
The models that are created by Hephaestus, using Kinect, are made 3D printable. This means you can export the models that are made in formats that are supported by 3D printers. Formats like .stl and .obj are two such formats that are supported for exporting these models. The printing of these models mean the people would actually be able to have their creations in hand physically. The results of user response to these printing can be analyzed below. We can clearly see that a large majority of people thought it would be great if you can get your models printed.

Table 4: Printing Models Graph



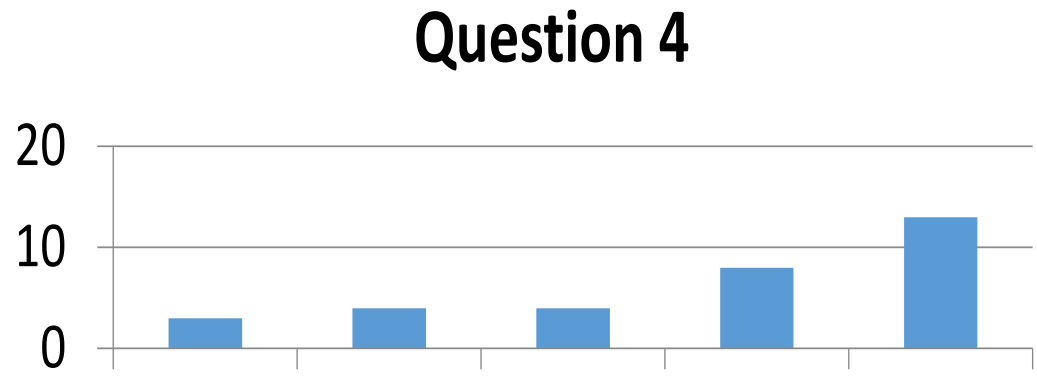
4.4.5 Hephaestus is easy to use.

Table 5: Ease of Use



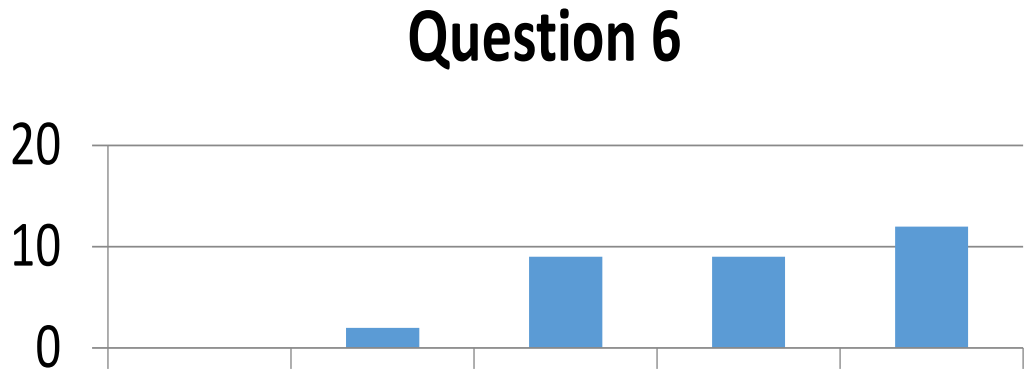
4.4.6 The desired output was generated using Hephaestus.

Table 6: Desired Output



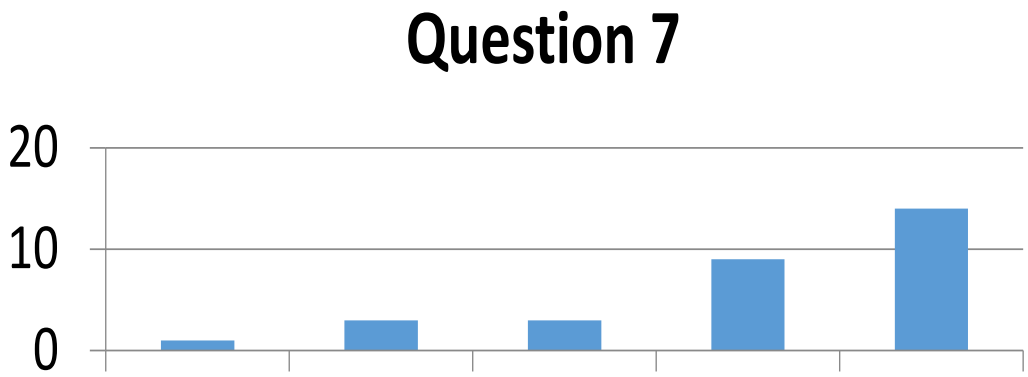
4.4.7 You would like to try Hephaestus again if given the chance

Table 7: Try Hephaestus Again



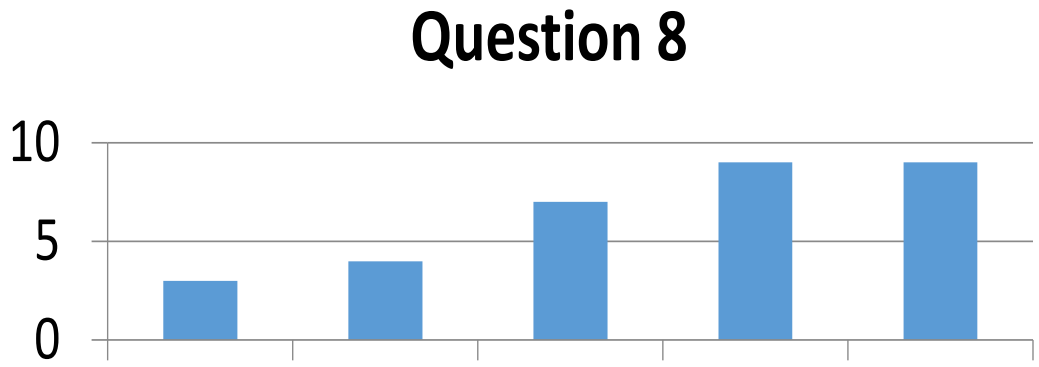
4.4.8 You would recommend your friends and family to try out Hephaestus.

Table 8: Recommend Hephaestus



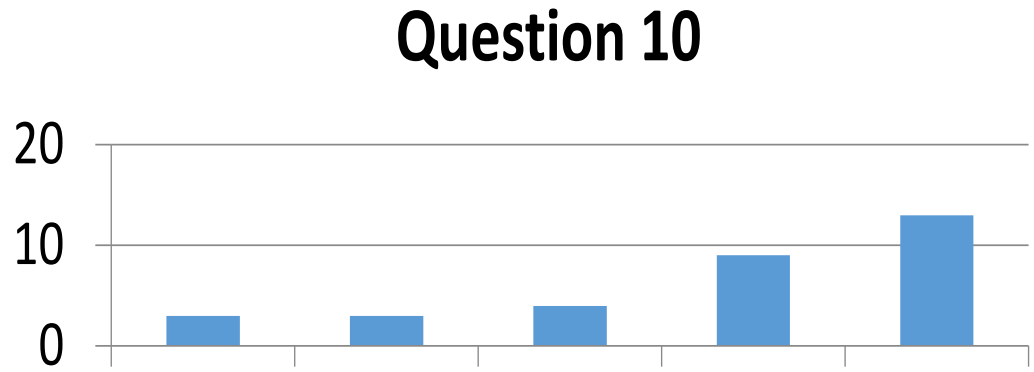
4.4.9 You would welcome similar technologies e.g. Oculus Rift, Leap Motion

Table 9: Welcome Technologies



4.4.10 You would like to use Hephaestus professionally (if applicable).

Table 10: Use Hephaestus Professionally



4.4.11 Conclusion

The results that we see with the survey indicate that users want to use such innovative systems if developers would develop them. We found that it is important to have a target audience in mind and listen to them after testing your product with them. This is exactly what we learned during out degree program.

CHAPTER 5

CONCLUSION AND FUTURE RECOMMENDATIONS

5.1 Theoretical Contribution

Hephaestus changes the way sculpting has been done in the past. It provides a new paradigm through which artists as well as people who wish to use it for fun, can create models and print them out using 3D printers. Previously sculpting was done by hand, sculptors used their hands to create piece of art. With the invention of computers the art of sculpting from the real world transferred into the digital world. The word “Digital Sculpting” or “3D modeling” came into existence. The people who created these models used mouse and other devices to create models.

With the creation of Hephaestus, the roots of sculpting using hand gestures and movements have been blended with technology to let people sculpt. This hybrid combination of technology and natural sculpting gives a new way to what was formerly known. With Hephaestus, users can stand in front of their workstations and create art with their hand movements. In short Hephaestus gives a new meaning to the word 3D Sculpting.

5.2 Engineering and Daily Life

Hephaestus allows you to create models out of thin air, this was the concept behind the project. Engineering disciplines like Civil, Architectural and Mechanical can be benefited with Hephaestus. Although currently Hephaestus lets you create pieces of art with very low emphasis on accuracy of the object as well the usefulness of data. This can extended to serve the above stated disciplines. Its users can use the 3D Modeling techniques used in software that let you model small scale prototypes for these engineering disciplines. Civil Engineers may be able to benefit from Hephaestus, if it lets them create structures and extend them

using gestures. Other engineering disciplines may benefit in a similar fashion by modifying the original Hephaestus.

In daily life, Hephaestus has the power to revolutionize entertainment. When we were kids we used paint as a canvas to express ourselves. With the rise of 3D printers, it is only fitting that we provide our younger generation a brand new way of expressing their selves. Hephaestus, could provide its users a 3D canvas where one can create models as fun and print them as pieces of art. Hephaestus has the potential to be the next Paint.

5.3 Limitations

Kinect is a great invention, but there are some limitation that need to be considered while using it. One of the major limitations of Kinect is the interference, since Kinect is very sensitive to data that comes in front of its field of view. Any form of interference caused can affect the way it reacts. Kinect tracks bodies as skeleton of joints, overlapping of joints are not recognizable for Kinect input. This means if you cross your hands, Kinect won't be able to judge which the right hand is and which one is the left one. Kinect tracks 6 bodies in the latest version, this means when a body comes in front of Kinect, it automatically detects it as input. Hephaestus is designed for one body in Kinects Field of View hence multiple bodies would cause disruption.

Kinect hand tracking is very intelligent, it can identify between various states of hand, from being open to close to unknown and to lasso. These states are determined judging the hand state. The frames are analyzed and Kinect judges whether the hand is open, close or in any other state. One of the major limitations of Kinect hand tracking is the angle at which you place your hand, for example if your hands are open but they are not facing Kinect at approx. 90 degree angle the Kinect might not detect. The hand being parallel to Kinect would be taken as closed since it only sees on skeleton part facing Kinect hence it would it think the hand is closed.

The Kinect has a great camera, the field of view is very big, and this allows the user to interact with a greater area compared to Leap Motion which has a very

limited field (LeapMotion, 2014). This large field of view has its limitations as well, it is very important that the field of view is kept out of any interference that might end up affecting the output. Hence in order to use Hephaestus, the users have to make sure no human interference is present in the field of view.

5.4 Conclusion and Future Recommendation

Hephaestus was created using SculptGl, the parent sculpting software that Hephaestus used, in order to allow Kinect Sculpting to take place. Once the software was completed the owner of SculptGL was informed about the success and Stephane Ginier hosted the project on the following website link. Team Hephaestus was also mentioned on the page as a fork of the original SculptGL

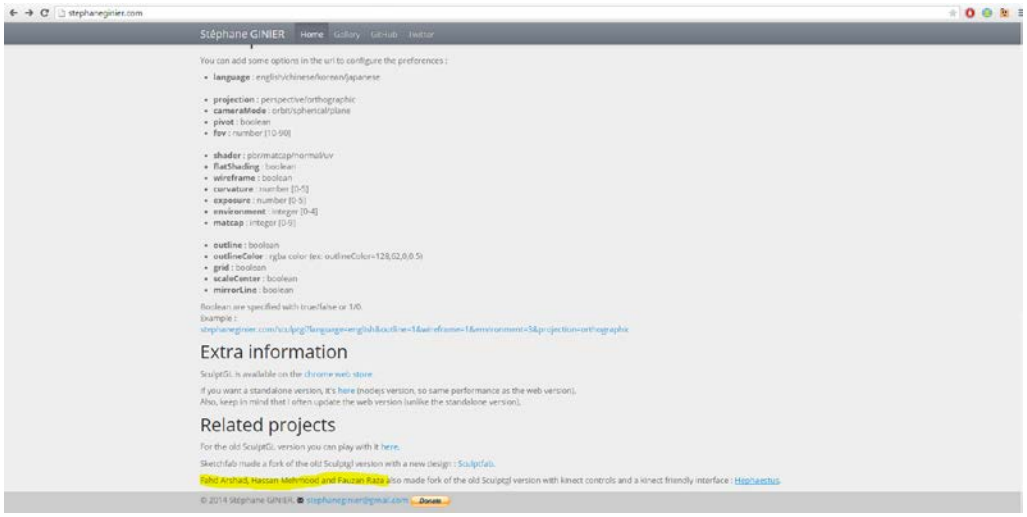


Figure 29: Stephane Ginier Website (Ginier, Homepage, 2015)

The work can be extended in the future by allowing users to interact more actively in the 3D space. The introduction of more interaction gestures could allow the users to work smoothly with the software. Hephaestus could be implemented for areas other than sculpting like civil engineering and other such disciplines. The interface could be made voice controlled in order to help the users to change brushes with their voice rather than moving their bodies.

REFERENCES

8bitjoystick. (2009, June 1). *E3 2009 : Microsoft at E3 Several Metric Tons of Press Releaseapalloza*. Retrieved from SeattlePi: http://blog.seattlepi.com/digitaljoystick/2009/06/01/e3-2009-microsoft-at-e3-several-metric-tons-of-press-releaseapalloza/

Autodesk. (2015). *Digital painting and sculpting software*. Retrieved from autodesk: http://www.autodesk.com/products/mudbox/overview

Blender. (2014). *Features Blender*. Retrieved from blender: https://www.blender.org/features/

Dylla, K. (2013, March 22). *The Digital Sculpture Project.* Retrieved from digitalsculpture: http://www.digitalsculpture.org/

Fahd, Hassan, Fauzan. (2015, May 21). *Hephaestus*. Retrieved from Stephaneginier: http://stephaneginier.com/archive/Hephaestus/

Ginier, S. (2014). *SculptGL - A WebGL Sculpting App*. Retrieved from stephaneginier: http://stephaneginier.com/sculptgl/

Ginier, S. (2015). *Homepage*. Retrieved from stephaneginier: http://stephaneginier.com/

Grant, C. (2010, July 20). *Kinect bundled with slim 4GB Xbox 360 Arcade for $300, new console for $200 in August*. Retrieved from endaget: http://www.engadget.com/2010/07/20/kinect-bundled-with-slim-4gb-xbox-360-arcade-for-300-new-conso/

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. (2011). Real-Time Human Pose Recognition in Parts from a Single Depth Image. *CVPR.* IEEE.

Jungong Han . (2013). Enhanced Computer Vision With Microsoft Kinect Sensor. *Cybernetics.* IEEE Transactions.

LeapMotion. (2014). *3D Motion and Gesture Control*. Retrieved from leapmotion: https://www.leapmotion.com/product

Maxon. (n.d.). *Organic Surfaces Made Easy*. Retrieved from maxon: http://www.maxon.net/products/cinema-4d-studio/sculpting.html

Microsoft. (2012). *Tracking Users with Kinect Skeletal Tracking*. Retrieved from msdn: https://msdn.microsoft.com/en-us/library/jj131025.aspx

Microsoft. (2014). *HandState Enumeration*. Retrieved from msdn: https://msdn.microsoft.com/en-us/library/windowspreview.kinect.handstate.aspx

Peckham, M. (2013, November 20). *Xbox One Review: Microsoft's Ambitious One-Stop Shop*. Retrieved from time: http://techland.time.com/2013/11/20/xbox-one-review-microsofts-ambitious-one-stop-shop/

Pixologic. (2014). *Sculptris*. Retrieved from pixologic: http://pixologic.com/sculptris/

Pixologic. (2015). *ZBrush Features*. Retrieved from Pixologic: http://pixologic.com/zbrush/features/overview/

Warren, T. (2012, March 27). *Kinect for Windows 1.5 will feature '10-joint' skeletal tracking and four new speech recognition languages*. Retrieved from theverge: http://www.theverge.com/2012/3/27/2906181/kinect-studio-kinect-for-windows-1-5-sdk