OIDYNAFACE

A Dynamic Face Tracker

Team 11

Derek Lam	40132682
Feike Qi	40079084
Adam Richard	27059329
Samuel Tardif	40051573
Lam Tran	40088195

A report submitted in partial fulfillment of the requirements of

COMP 371: Computer Graphics

Concordia University, April 28, 2021

Contents

1. Introduction 2
2. Project Overview 2
   2.1. Setting Up the Environment 2
   2.2. Including the OpenISS Library 2
3. Commentaries 5
   3.1. Goals 6
   3.2. Accomplishments 6
   3.3. What We Learned 7
4. OIDYNAFACE’s Future 7
5. Conclusion 8
References 9

1. Introduction

This document serves to outline the process of developing the third portion of the main project for COMP 371: Computer Graphics at Concordia University. The first two parts of the project were spent creating an environment in which we could implement part 3, which was the OIDYNAFACE implementation. More specifically, this document serves to describe how this project is built, our process when creating it, and what we learned.

2. Project Overview

This section contains a description of the project as whole, including our implementation and the OpenISS library.

2.1. Setting Up the Environment

This encompasses both the first and second part of the term project. In the first part, we implemented the ability to render objects in a scene and apply different colors to them. In addition to this, we implemented controls to manipulate those objects by rotation, translation, and scaling. We also added a first-person camera to the project for easy traversal of the digital space.

In the second portion of the project, we introduced more functionality by adding texture mapping and lighting. In addition to this, we added a stage with a screen, on which different textures would be displayed sequentially. We also updated the shaders to be able to render the lighting as specified in the project guidelines, that is, a point-light positioned above the stage. New controls were added to enable the shearing of models from their base.

We were, at this time, also supposed to implement shadows in the project. Unfortunately, this was a complicated task that we were not able to see to completion. While we do feel we were close to achieving generating shadows for our objects, we ran out of time and were forced to submit without it.

For both steps, we relied heavily both on the labs presented as part of this course [1] and on the OpenGL tutorials available online [2]. Much of our understanding and implementation of the code was based off these two sources.

2.2. Including the OpenISS Library

This is the library that allowed us to create the face animation for the final project, and was taken from our BitBucket project repo, originally forked from the OpenISS GitHub repo [3]. The OpenISS static library contained two classes that we used in order to implement the face tracking functionality. For the purposes of this assignment, we did not track our own faces. Rather, we used an image of our face and hard-coded the landmark reference points that were then used to create the face animation in the final product.

The first step of including the library was to decouple the OIFace and OINullFaceTracker from OpenCV, another open-source library that we were not using in this project. These are the classes we needed to implement the face tracker. Fortunately, the types that were called from OpenCV had also already been implemented in OpenISS in a class called OIType, so decoupling was easy.

OIFace is a simple class that creates an object made of facial landmarks, 72 points that mark specific points on the face. These are the points that are moved in order to animate the face. OIFace then, being a collection of these points, can be thought of as a face object.

OINullFaceTracker is a class that creates and keeps track of OIFace objects. This is done through a series of points that “can be delimited as anatomically corresponding points,” [4]. While it is theoretically possible to import a face and determine its points based on input, for the sake of rendering a face that will smile and frown we decided to hard-code points into a new class called OINullDFaceTracker. We determined where these points would be by taking a picture of one member’s face, identifying the points on his face, placing those points on a grid and extracting the values from that grid.

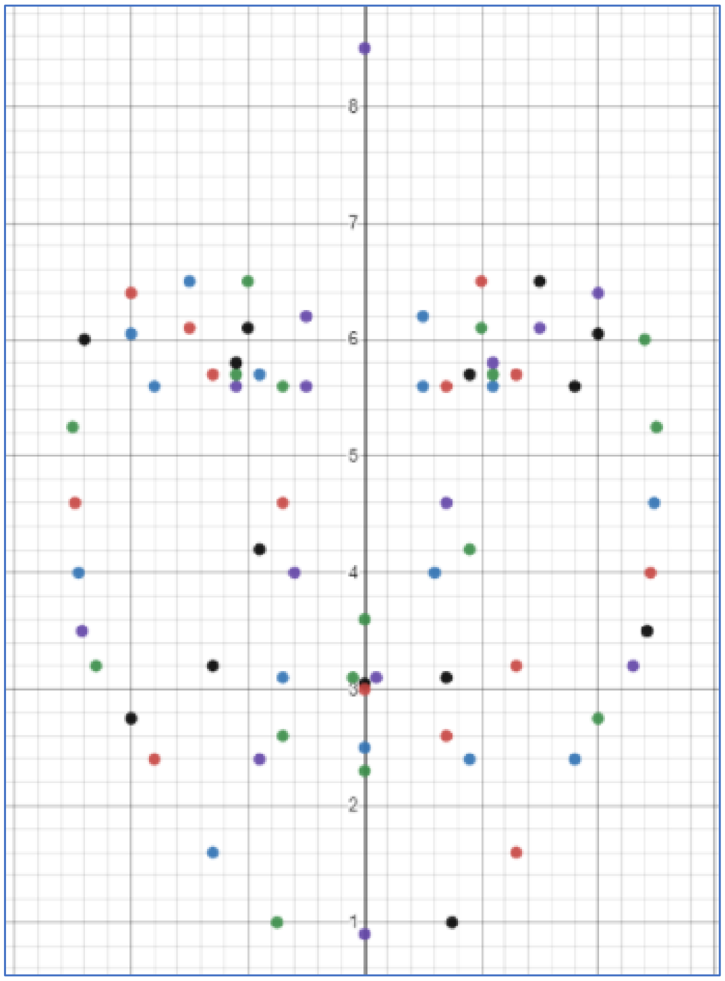

Figure 1: Face Points on a Grid (Smile)

The size of the grid on which the points were placed was decided based on the size of the "screen" object that was already available in our application environment that was implemented in part two of the project. The screen object is about ten units high (units being an arbitrary unit of measurement based on floor grid in our application). It was then decided that the rendered face would also be a maximum of ten units high, and so all the 72 points that make up the face were placed on a grid with no point exceeded ten units.

These points were then numbered and applied to the program. In our research we noticed that there was no standard for which numbered point corresponded to which point on the face. We thus determined that it was up to us to determine this for our specific project. Points 1 to 19 would make up the jawline and sides of the face. The forehead would be denoted by a single point at the top, which is point 20. The left and right eyebrows are points 21 to 36, the left and right eyes are points 37 to 52, and the nose is points 53 to 59. The remaining 13 points, 60 to 72, make up the mouth. These last 13 points are perhaps the most important as those are the ones that we would need to change in order to create the animated smiling and frowning face. The first 59 points would remain their positions between all versions of the face.

In order to animate the mouth then, we only needed the new points for this part of the face. We took two new pictures, one with a neutral face and another with a frown.

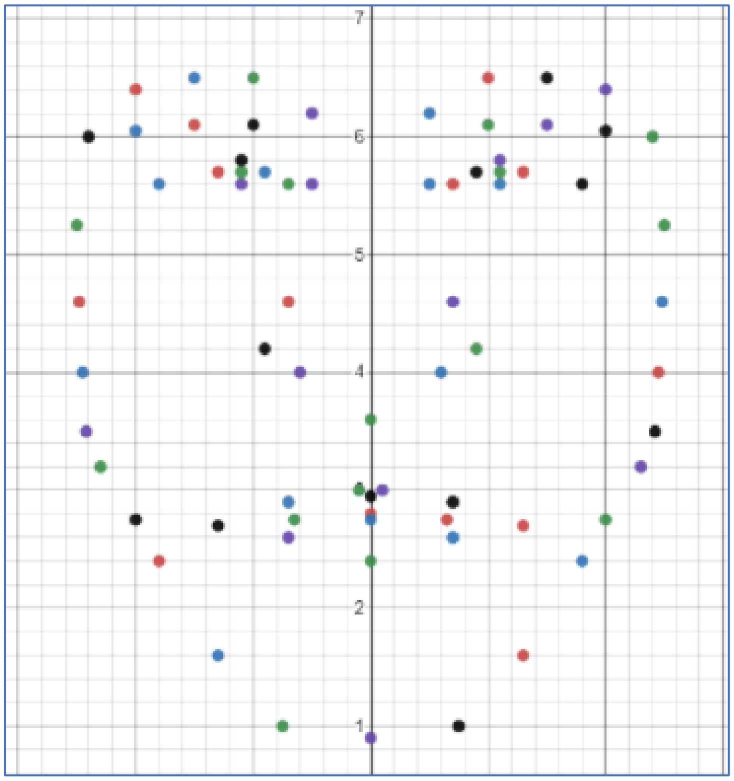

Figure 2: Face Points on a Grid (Neutral)

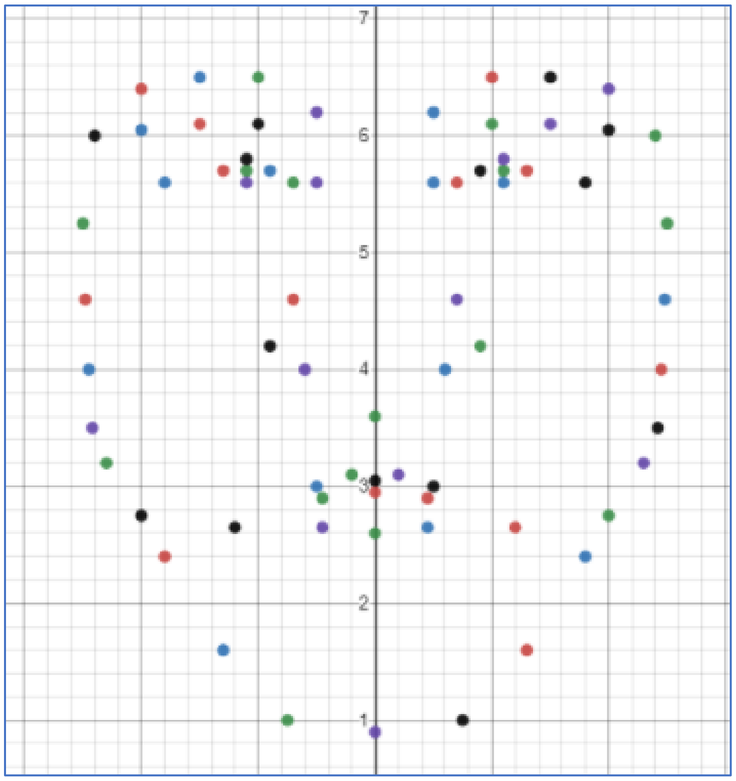

Figure 3:Face Points on a Grid (Frown)

We now had the first, middle, and last position for all the points in the face, but we wanted a total of seven positions, not three. In order to obtain the other four, we interpolated the points in between to generate new faces that would serve to animate the face as its mouth changed position. Thus, we created seven different frames by making seven different face objects.

Once the seven face objects were made, it was only a matter of rendering them in the scene and having them alternate between themselves over time. These points were defined in a new class called OINullDFaceTracker. This would also be the class where any new face to be rendered is defined. This allows for the library and our application to be extended further than the purposes of this project. While we did hard-code these points into the library as a default face and animation, it should be noted that this class should, in the future, allow for importing new points to be used.

3. Commentaries

This section serves to describe our experience while creating the project and implementing the OpenISS library. It will discuss our goals, accomplishments, what we learned, and our interest in this project.

3.1. Goals

Our main goal while doing this project was to gain knowledge of computer graphics using the OpenGL library. This was achieved by the end of part two of the project, when our environment had finished being set up. Most of our knowledge had been fermented by this point, and so our implementation of part three was more of a way to put that knowledge to the test by using that knowledge to include yet another library with more functionality.

The new library was OpenISS, in which there were classes that defined how we would track a face and animate it. From here, our goal was to create an implementation that would create a type of "skeleton" for the face that could be animated [5]. This would not work like a typical model skeleton for animation, however. Instead of being a collection of lines connected by points, we would only use the points and animate those.

3.2. Accomplishments

For the most part, we were successful in accomplishing our goals. There was a single shortcoming when creating our environment, that being that we were not able to implement shadows. We were however successful in creating lighting, so some realism exists in our project in that sense. When implementing part one, we were also able to create a first-person camera, which made it easy to explore the 3D space and observe it from any angle we wanted to. This ended up being hugely beneficial in the coming parts, when we needed to test quickly and effectively, as we could observe our work to see if our new implementations were successful. We managed to implement this feature by following on online tutorial on OpenGL's website [2].

After doing this, we noticed that when we selected one of our models, the camera would only be facing the correct direction when holding down the button to select it. The reason for this was that the first-person camera would have its own values for where the camera should be looking. We resolved the issue by editing these values to match the camera when selecting a model.

When it came to including the OpenISS library, we had some initial difficulty importing it and getting it to work with our project. Ultimately, this was solved by downloading the folder instead of cloning the repository on our BitBucket. Once this was done, we were able to compile and build the library and add it to our project. From there it was a matter of altering the classes discussed in section 2 of this report.

Another of our accomplishments from part three of the project was identifying the points that created the outline of the face. This was accomplished by taking a picture of one group member's face and identifying points on it that we then implemented in the code. This would ultimately serve as the outline for the rendered face in our project that becomes animated. During this process, we also determined the points that would exist for each of the faces that would make up the animation of the face smiling and frowning.

3.3. What We Learned

While we did learn a great deal about OpenGL and computer graphics in general, it is also important to note that we learned a great deal about how to import new libraries into a project and about the theory behind animation.

Animating the face taught us a great deal about how movies and videogames implement these techniques in order to create realistic movement in their models, and about how motion capture works when it is used. We learned that creating a model skeleton and facial landmarks works similarly when moving different parts of the model. We believe this will be useful in our careers when developing graphics for various projects.

4. OIDYNAFACE's Future

After completing this project, there is a clear path that can be taken to continue OIDYNAFACE in the future. While we did implement that functionality of an animated face in OpenISS and tested it using the environment we developed through the course of the semester, there is still the open-ended question of whether this software can identify the correct landmarks on a given individual's face and animate that face in real time.

We believe this is possible, but some steps would need to be taken for it to be accomplished. First, the hardware would be needed. This may be accomplished with a camera with a high enough resolution, but a motion capture suit with specific identifiable points would make it even easier. Second, the OpenISS library would need to be modified such that when identifying the landmarks on a given face, it can organize which landmark creates which vertex when animating that face. Only then will it be able to move the face on-screen at the same time as the individual moves their own face. This could be done directly in the OINullDFaceTracker or in a new class that then sends the data to OINullDFaceTracker.

One of the reasons this identification process would need to take place was because we found no standard way of numbering the landmark points on a face. This is understandable, as there could be a variable number of points depending on the accuracy intended. Since there is no standard, the library would need to be capable of matching which landmark belongs at which point in the software. This means determining the coordinates of each point in real time. It would otherwise be the same as the process we went through to animate the smiling and frowning face.

While this may be a difficult challenge, it is still an obvious and possible future for this library, and it would be interesting to see such a development and implementation come to fruition. Perhaps a middle step to this end would first be to make sure the software can create a set of coordinates based on an input face. This would need to be completed in the OINullDFaceTracker class, as that is where that information is handled in the library.

5. Conclusion

This project challenged us and pushed us to learn difficult material in a short amount of time. Ultimately, we were successful in learning the skills necessary to build the project, but there were a lot of challenges along the way. We are proud of the work we did to generate the environment that the OpenISS library would be added to, and finally the work done capturing and animating the face. We learned a lot about libraries in addition to the OpenGL library in specific and extended our knowledge beyond this base to animation as well.

References

[1] J. Llewellyn. COMP 371. Laboratory Lecture, Topic: “Lab 04.” Gina Cody School of Engineering and Computer Sciences, Concordia University, Montreal, QC, Feb. 10, 2021. [Online]. Available: https://moodle.concordia.ca. [Accessed April 28, 2021].

[2] OpenGL-Tutorial, “Basic OpenGL,” *OpenGL-Tutorial.* [Online]. Available: http://www.opengl-tutorial.org/beginners-tutorials. [Accessed Apr. 28, 2021].

[3] OpenISS, *OpenISS.* [GitHub Repository]. Montreal, QC: OpenISS, 2020. Available: https://github.com/OpenISS/OpenISS. [Accessed Apr. 28, 2021].

[4] K. Kleisner, “Fig 1,” *researchgate.net,* para 1, 2021. [Online]. Available: https://www.researchgate.net/figure/a-Configuration-of-72-landmark-locations-on-the-face-landmarks-that-can-be-delimited_fig1_222556791. [Accessed Apr. 28, 2021].

[5] S. Mokhov, “Initial Project Meeting,” Apr. 14, 2021. [Recording].