Brendan Jacques 01419198

Computer Graphics Final Project

Table of Contents

1. Objective

Objective: The object of my final project was to, using a programming language of my choosing, create a program that could:

1. Modeling: create and store a 3D object by any number of these means:

a. Draw three 2D "elevations" (front, top, side – see, for example, "my dream house" or "my dream car (front, side, top)"; your implementation should be able to "accept" any reasonable generic object, not just "my house" or "my car"). Upon drawing, store coordinates of the elevations in a way that will allow you to create a 3D model of the object from them.
b. Enter coordinates: choose your model format(s) (e.g., vertices, edges, primitives, other).

2. Transform object: apply 3D (Translate/Rotate/Scale/Shear) transformations to the created object.

3. Viewing: view your created object from multiple views.

4. Transform camera/viewer/light sources(s).

5. Generate different projections of the objects (refer to class discussions about different projections, see projection "tree" see figure).

6. Edit/Change perspective projection vanishing points (1, 2, 3).

7. Create texture/bump/environmental mappings for the object.

2. Introduction

For my final project, I chose to implement my program in Javascript using the Three.JS Library. In my completed program, the following features are included:

1) Generate a 3D model of a simple house and project it onto a canvas within the web page.
2) Provide controls to the user that allow them to transform the object's position within the canvas, rotate the object, scale it, and shear it.
3) Provide the user with multiple camera views from which to see the object from.
4) Provide controls to the user to modify the positions of light sources within the canvas's environment.
5) Provide Camera views to the user that conform to the specific projections discussed in class. These include:
    a. Orthographic Projections from the object's front, side and top.
    b. An Isometric Projections of the object's front-right side.
    c. Dimetric Projections along the "X and Y" axes, the "X and Z" Axes, and the "Y and Z" axes.
    d. Controls that allow the user to simulate oblique projections through shearing of the object's back viewed from the object's front.
    e. A Parallel Projection with controls that allow the user to simulate manipulation of the object's vanishing points through changes to the camera itself.
6) Create textures for the object and environmental textures for the skybox surrounding the object.

In terms of completion, I feel that my program fulfills the requirements of every feature listed on the rubric with the possible exception of proper vanishing point manipulation in Parallel Views. Three.JS doesn't have native support for manipulating vanishing points, so I tried to replicate the effect by altering the camera's field of view dynamically alongside camera position/rotation, but the result is a bit buggy and I'm not entirely certain if it qualifies. I should also note that the feature for texture/environmental mapping ended up being only partially successful: While the textures given to the program are applied properly in Firefox and Microsoft

Edge (i.e. the current version of internet explorer), the current version of Google Chrome has hardline CORS policy restrictions that prevented textures from being loaded in Three.Js without a local server. I was unable to find a way around this restriction prior to the due date, though to reiterate, the textures given do work on browsers other than Chrome. I was unable to test if Safari works in this case due to working solely on a PC.

I would also like to note that it is my understanding that this project submission does not conform to the original intent of the rubric as written. While my program generates a 3D model as described in feature 1 on the rubric, it does not allow for the user to draw their own custom objects with either three drawings on a 2D plane or via coordinate entry. However, after speaking with Dr. Levkowitz on the subject, he has given me permission to pursue this version of the project regardless.

3. Progress Report Summaries

Before continuing, I would like to clarify that the project submission I am presenting here was not my original attempt at completing the project. For the first four weeks of our allotted time, I worked on a version of this project that allowed the user to create their own 3D objects by giving them the tools to draw 2D objects into three different canvases, which would then be combined into a singular 3D model. This version of the project was created in Javascript using the WebGL basic library and had features 1 through 3 of the rubric included before I made the decision to start over due to quality concerns and confusion over what was required by the final submission. For the sake of posterity, the code for my original project will be included in this submission within the "old-code" folder, though no code from this version of the project is present in the final version of the project.

With that in mind, here is a brief overview of what I worked on each week of the project's duration:

A. Week 1

No submission was given for Week 1 as I did not feel at the time that my project was far enough along to warrant any credit.

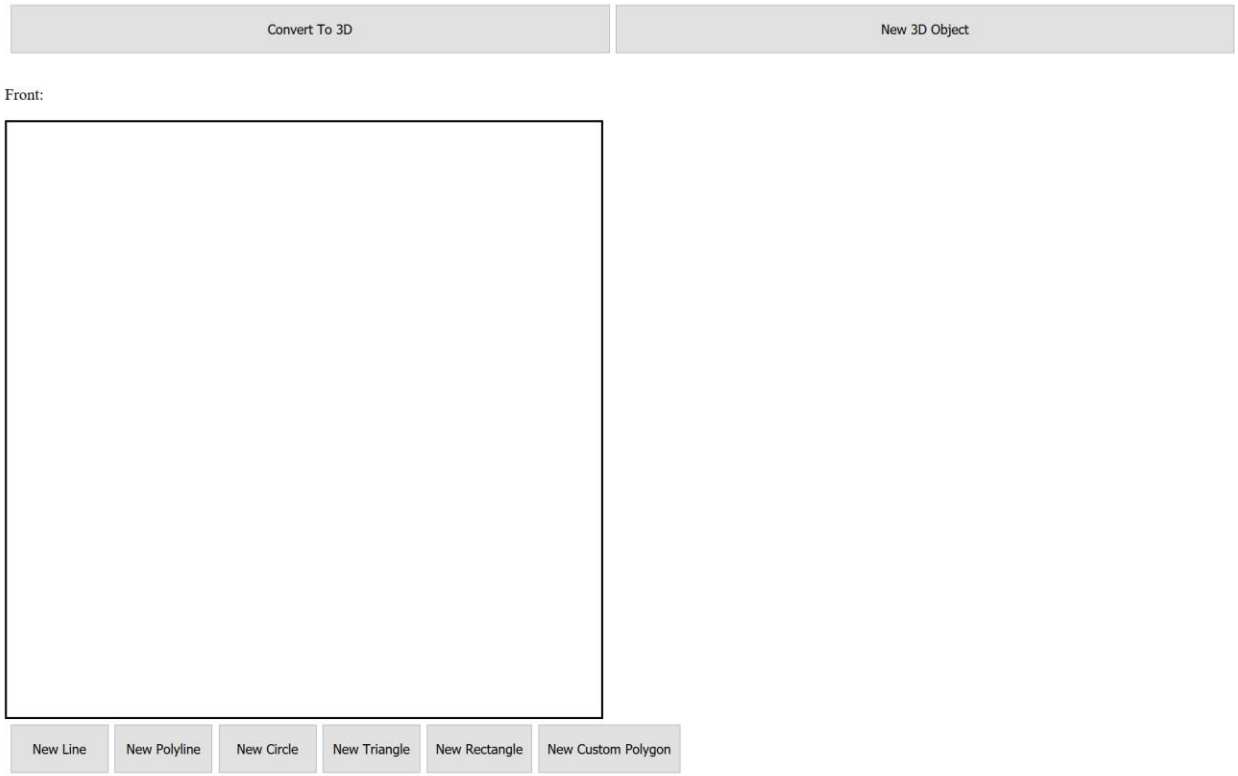
B. Week 2

This week, I completed the UI for Version 1 of the project (as pictured below), which included:

- Three canvases powered by 2D WebGl, one for the front view of the 3D object, one for the side view, and one for the top view.
- Drawing tools that allowed the user to make drawings in each canvas using lines, polylines, and triangles (with other shapes to come later). At this point, the object's color was also relegated to a default pink shade.
- A "Convert to 3D" button which attempted to generate the 3D model by combining the 2D objects drawn by the user in the given canvases. By this stage of the project, the program was not able to create the 3D object visibly, but it was able to calculate the coordinates that would make up the object.
- A "New 3D Object" button, which reset all canvases to allow the user to draw a new object.

This version of the project was implemented a system of arrays which stored the coordinate data of every shape that made up the user's drawings on each canvas. Similar arrays were also kept to store what type of shape each set of coordinates stood for. The idea was that, when the user presses the "Convert to 3D" button, the separate arrays of each canvas would be combined into a singular "Master" array, which also converted each coordinate into a 3D point.

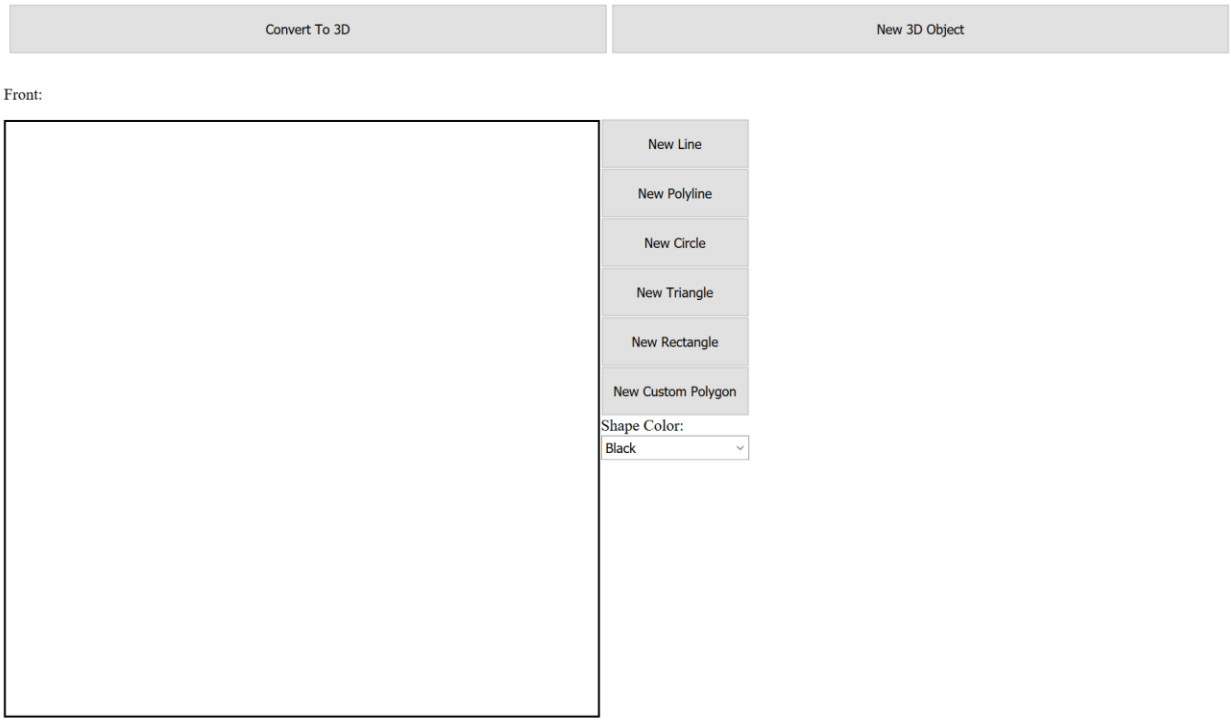
Week 2 UI Example for the Front Canvas:



C. Week 3

This week, I was able to get the 3D canvas to properly display the combined 3D model. Also added experimental controls that allowed the object created in this way to be translated around the canvas, rotated, and scaled (though the scaling by this point was unfinished and glitchy) along the x, y and z axes. On top of this, I also made some adjustments to the project's UI and provided a control to allow the user to change the colors of the objects drawn. By this stage in the project, however, the 3D model generated was not aesthetically correct: the 3D object generated was just three 2D planes that were being projected into 3D at different 90 degree angles from each other.

Week 3 UI Example:

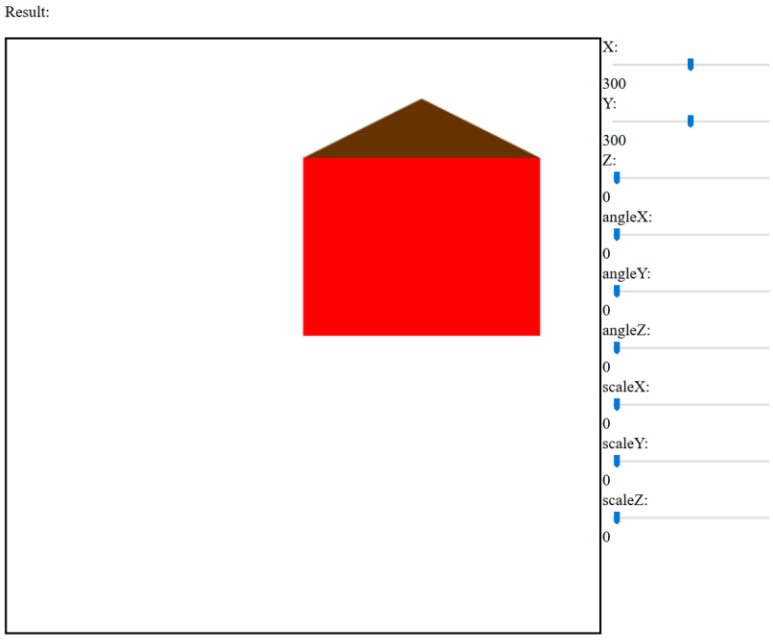


D. Week 4

For Week 4, I mostly stuck to fixing issues with my drawing function, in particular a glitch where only certain shapes drawn by the user were being properly converted into 3D or a glitch where the given colors of objects were being altered beyond the user's control. Also added functions that allowed the user to make their drawings using rectangles, custom polygons, and circles of variable radius alongside the shapes provided previously. I also intended to improve the generated 3D object so that it would look more aesthetically pleasing, but this week was also when I had decided to start the project over from scratch in ThreeJS. Due to confusion over the rubric that was caused by a misunderstanding between the TA and myself, I ended up taking out most user creation features of the project for my submission that week.

To explain, for my original project, I was under the impression that the assignment was to allow the user to create their own custom 3D models by combining 2D drawings together. But that week, the grader informed me that this version of the project was incorrect and that the project was actually just to generate and transform a pre-made 3D object. This turned out to be incorrect, but this is what I was told at the time and what informed my decision to start the project over from scratch.

Week 4 Submission Photo:



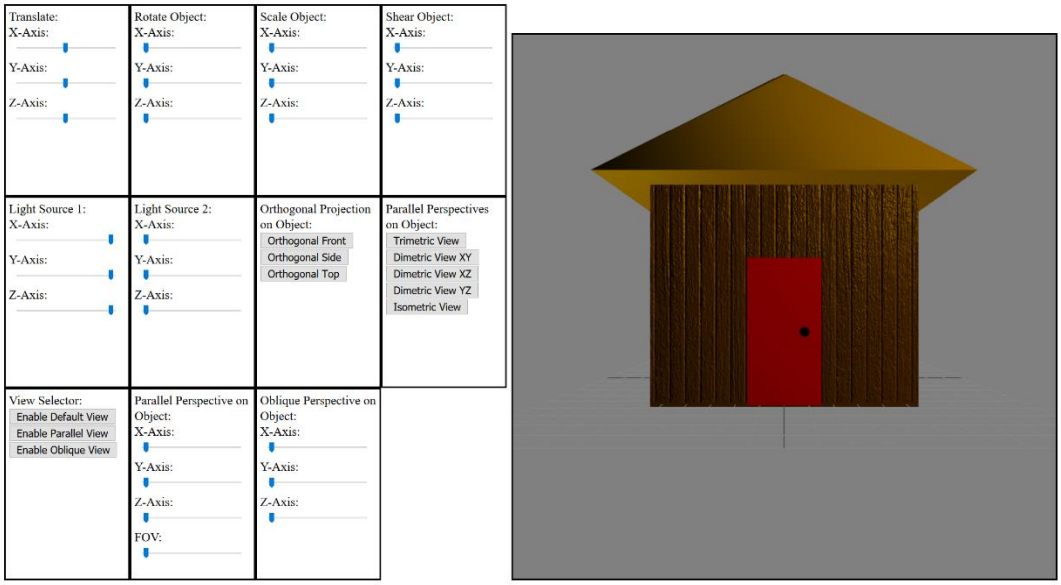
E. Week 5

This week was spent creating version 2 of my project using the Three.Js library instead of basic WebGL. By the submission deadline this week, I was able to complete steps 1 through 5 and 7 of the rubric in my new program whilst also drastically re-designing the UI and without re-using any code from the original project. This includes the creation of a simple 3D house model within a given Three.Js Canvas, controls for manipulating the object (Transform, Rotate, Scale, and Shear), controls for manipulating two distinct light sources within the scene to illuminate the model house, and a series of buttons that swapped the view displayed on the canvas between a default perspective shot, an orthographic front, side, and top view, a trimetric side view, an isometric side view, and three different dimetric views. Shear wasn't working properly at this phase, but all the other above features were.

The transformations for both the model and the light sources were handled by positional sliders that would send the new values to a function that would generate the proper transformation matrix based on the user's selections. A new version of the house would then be created with this transformation matrix plugged into it, which would then be rendered in the next frame.

On top of this, this version of the assignment also included wood textures for the walls of the house through Three.Js's texture loader, though as noted previously, these textures don't work properly on Chrome.

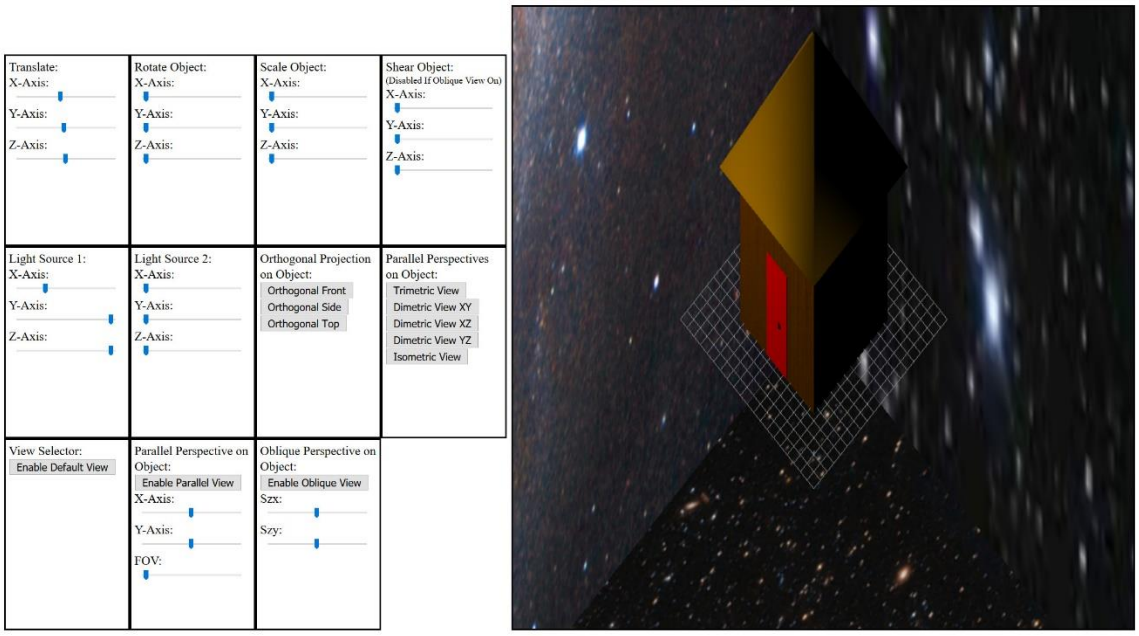
Week 5 Photo:



4. Final Submission State

The final version of this project is similar to the one given in Week 5, but with the additions of controls for simulating an oblique view of the house, controls for a parallel perspective of the object with the ability to manipulate vanishing points along the x, y, and z axes, and a skybox surrounding the model (that also does not function in Google Chrome and is a bit buggy).

Final Project Example Picture:



5. Impress Me Features

- I'm not sure if this qualifies as an "impress me" feature, but I've included the WebGL version of my project in the "old-code" folder. While that version isn't nearly as feature complete as the Three.Js version that is my primary submission, I think the data structure I set up in that version to organize shape coordinates, colors, and shape types for each canvas as well as the method I used to merge each canvas' individual list together is worthy of credit.
- Included three different Dimetric Camera Angles.

Sources:

1) https://threejs.org/

The documentation for the ThreeJs programming language. No specific tutorial code was taken for the assignment, only used as reference for proper implementation.

2) https://codepen.io/rachsmith/post/beginning-with-3d-webgl-pt-1-the-scene

The original framework in which I built the project in came from this website. This consists of the render loop and the initial creation of the scene.

3) http://danni-three.blogspot.com/2013/09/threejs-skybox.html

Re-used js code given in this tutorial in order to get my scene's skybox working properly.

4) https://stackoverflow.com/questions/26021618/how-can-i-create-an-axonometric-oblique-cavalier-cabinet-with-threejs

Used the code given in the jsfiddle given as the solution here as a base for implementing my oblique perspective controls. This example gives how to generate a single oblique view, I iterated on it to allow the user to control the view dynamically.

Outside of the sources listed above, all of the code in this project submission is my own.

P.S. Below are the sources used in my WebGL version of this project.

1) https://www.html5canvastutorials.com/advanced/html5-canvas-mouse-coordinates/

I took the getMousePos function used in this tutorial for use in calculating the mouse's position on the canvas.

2) https://stackoverflow.com/questions/42309715/how-to-correctly-pass-mouse-coordinates-to-webgl

Modified the getMousePos above with additions from this tutorial in order to make the function work with WebGL.

3) https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html

Used this website as a learning tool for understanding how to create 2D models in WebGL. The createShader() and createProgram() functions in my code were copied from here. The majority of code for draw2D() and draw3D() were also taken from here, though with modifications to suit my project's needs. Also re-used their code for calculating transformation matrix changes.

Commented over the functions I re-used in my js file.