Project 3

David A Robbins

CMSC 405 6380

University of Maryland University College

February 16, 2018

This project was even more difficult than the last, as expected but I really had fun doing it. I started out just looking at the examples so I could do my best at writing my first program in JavaScript, and luckily the syntax is very similar to Java, which is the only language I know well. The first hurdle I had was which IDE to create this project in; I am used to NetBeans from my Java programming in previous classes, so went for that first, but ran into a few issues. Looking online I found that Atom was a really good one, and after downloading that and figuring out how to run a local server from it, I was editing and running the example files in no time. I did a great deal of research on the three.js documentation website as well, and eventually got to a point where I wasn't really creating anything of my own, but was just copying the code provided. I didn't feel like I was learning it that well, so started from square one and started watching a youtube tutorial guide provided by Sonar Systems (youtuber - link to tutorial series: https://www.youtube.com/playlist?list=PLRtjMdoYXLf6mvjCmrltvsD0j12ZQDMfE). This was very helpful, as he started with the very basics of setting up a project and running it on a local server, and built it up from there. After about half of the series I was comfortable enough to start doing my own thing, and from there it was just debugging and testing until it was working the way I wanted. My plan was to have six or more shapes show up on the screen, moving around in some way, then for buttons to be associated with each shape, so that when a button was pressed the shape showed up on its own in the middle of the screen, doing its own rotation transformation. The reason I wanted these particular types of transformations was so that the various lighting types that I used with mostly the Lambert material could be debuted. The result was exactly what I wanted. I even through in a couple of bonus objects; the first was a TextGeometry object and the other was a PlaneGeometry object. The Text one is static and

doesn't move other than disappearing when a particular shape is debuted, or reappearing when the Shape button is pressed. The PlaneGeometry one is connected to the Phantom Zone button, and if you are a fan of the old Superman movies, you will understand why. If not, google 'phantom zone superman' and you will get why I attached the images I did to the two sides of that object and rotated it the way I did. One other element I included was being able to zoom in and out with a middle mouse click as well as rotate and move the screen with left and right mouse clicks, with the use of OrbitControl.js. In the testing, I tested all of the buttons as well as the mouse clicks, and everything by the end was running the way I wanted it to. I won't list the functions that I tested, as the buttons themselves call on multiple functions, some of them overlapping, so will instead include in my test cases a screencap of the image after pressing a button, as well as a few screencaps of vantage points taken with the mouse clicks.

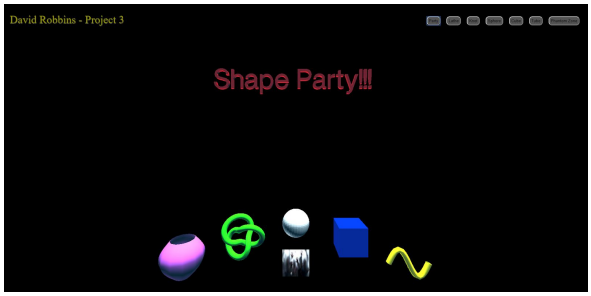
TESTING

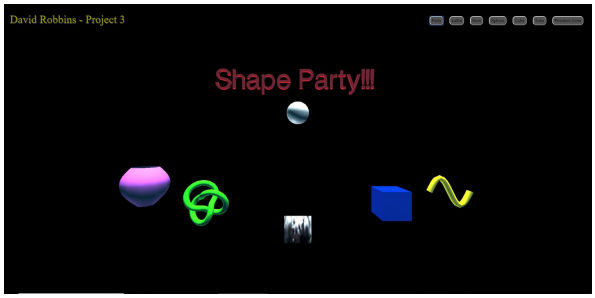
Initial Load of ‘Project 3’

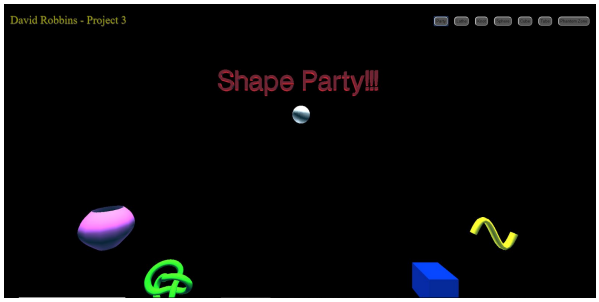
This is just what it looks like before any buttons have been pressed, and the program has been loaded in a Chrome browser on a local server.

Shape Button

This is the main button which puts a TextGeometry object at the top of the screen and has all of the other objects float up towards the text together from outside of initial viewpoint. When the objects arrive at the text object, they each change direction and go back the way the came to exit the screen, except for the SphereGeometry object, which instead scales down to disappear right when the other objects leave the screen. At this moment, the scene loops and starts again





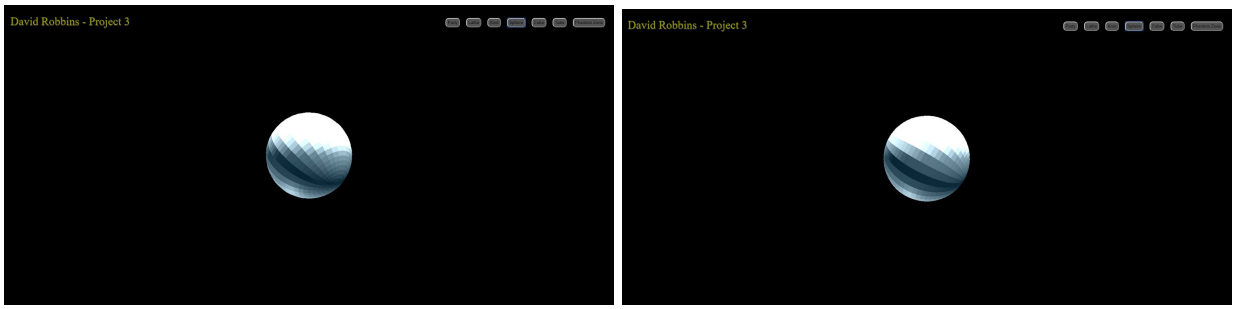


The following six buttons all have the same type of actions on them as their associated buttons are pressed which is to have them scaled up and rotated in the middle of the screen, so as to debut the effects the various lighting types have on the objects. I will allow this description to cover the following tests

Lathe Button

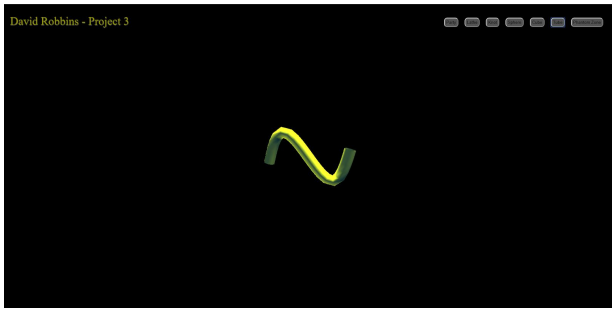
Knot Button

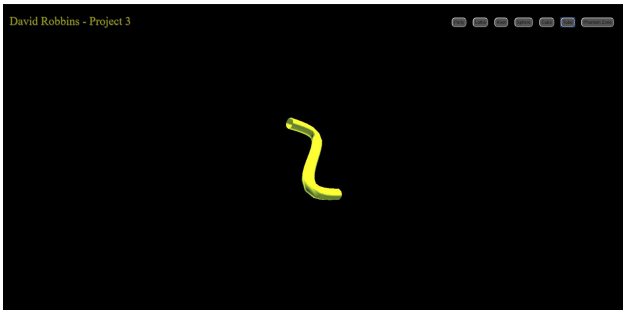
Sphere Button



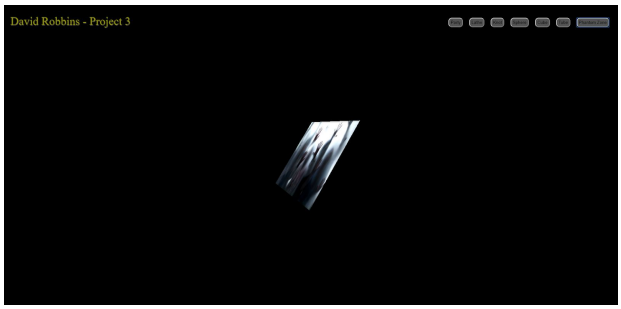
Cube Button

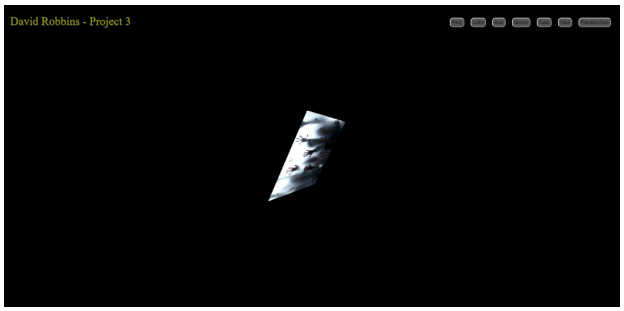
Tube Button





Phantom Zone Button

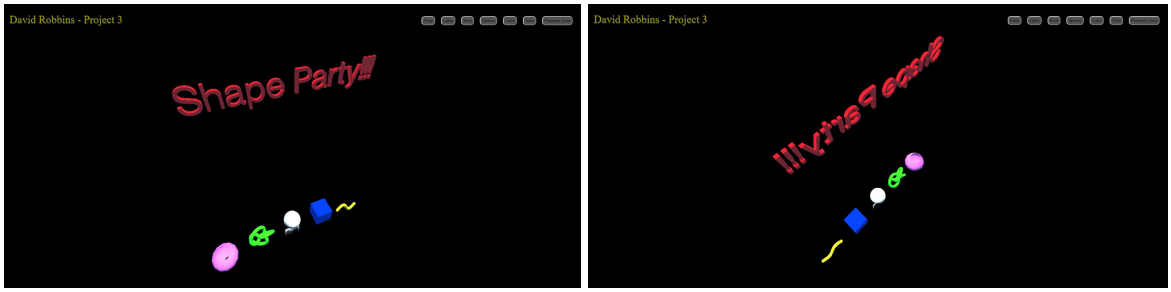




The following tests were conducted after the initial loading of the program, so you can see that the initial positions of the main objects (besides the text object) are not able to be seen yet. This was intended, as the Party button is meant to have them float in and out of the screen from their initial position. It also allows for easy testing of the mouse functions, as each function lets me adjust the viewpoint of the screen in a way that lets me view those objects that are initially unviewable.

Left Mouse Hold, and Move Mouse

This rotates the screen around the scene in accordance with the motion of the mouse while the left mouse button is held down.



Right Mouse Hold, and Move Mouse

This translates the scene while holding down the right mouse button and moving the mouse.

Middle Mouse Hold and Move, or Middle Mouse Scroll Wheel

You can zoom in and out by holding down and moving the middle mouse button or using the scroll wheel.