Project Report

Computer Graphics I

Aanchal Sharma

aanchal_sharma@student.uml.edu

Abstract

This project aims to create a 3D object by drawing three 2D "elevations". The project implements many features like: modeling, transform object by applying 3D (Translate/Rotate/Scale/SHear) transformations to the created object, viewing the created object from multiple views and generating different projections of the objects. Further, implemented mapping and also explored lightening effect in Computer Graphics.

Introduction

The project will employ the use of WebGL which is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. WebGL is integrated completely into all the web standards of the browser, allowing GPU-accelerated usage of physics and image processing and effects as part of the web page canvas. WebGL elements can be mixed with other HTML elements and composited with other parts of the page or page background. Due to this reason I am choosing to work with it. The project will be able to render user defined images in 3D and also provide the ability to
Transform the object
V_i_e_w_ _t_h_e_ _o_b_j_e_c_t_ _f_r_o_m_ _m_u_l_t_i_p_l_e_ _v_i_e_w_s

Transform the lighting of the object

Generate different projection of the object

Change the perspective projection vanishing points

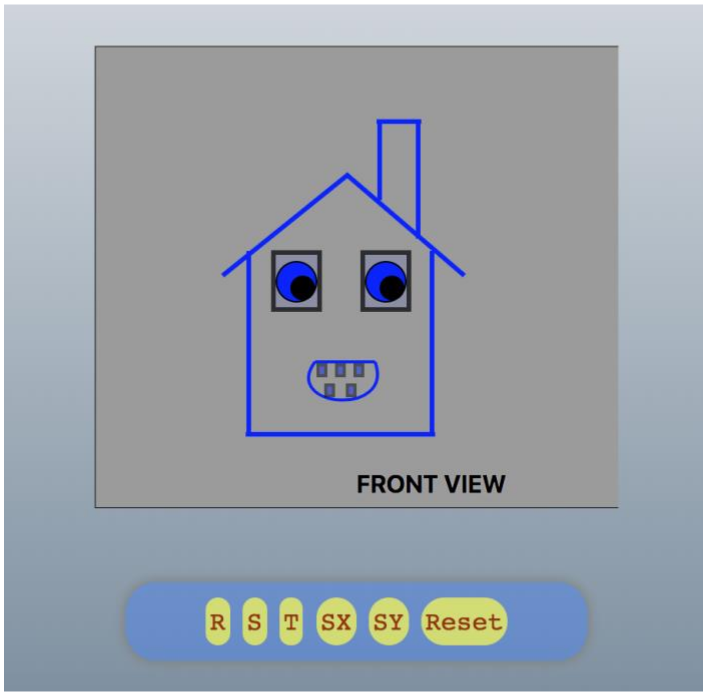
Create texture/bump/environmental mappings of the object

Analysis and Design

Week 1:

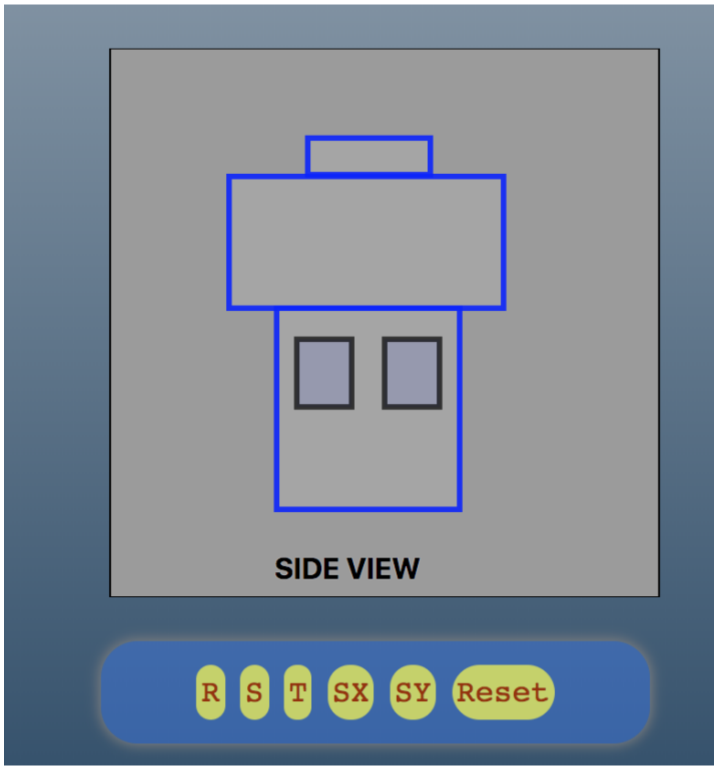
Please refer:
http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v1/finalProject.html
I created the full working model of front and side view of my dream house.

Front View:



Select R to rotate the view; S for scaling, T for translation; SX for Skewing X-axis; SY for Y-axis Skew and Reset to enter the state representing the natural state.

Side View:
Select R to rotate the view; S for scaling, T for translation; SX for Skewing X-axis; SY for Y-axis Skew and Reset to enter the state representing the natural state.



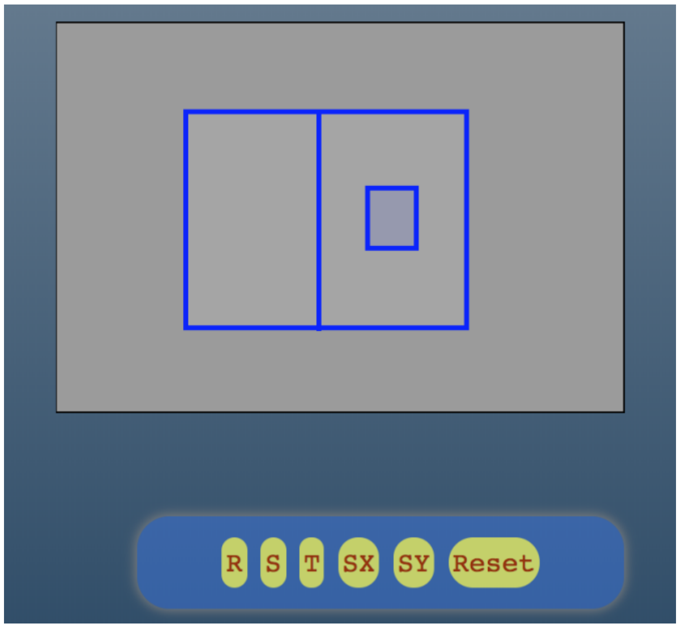
I have svgs in html for the front view of my dream house and rear view of my dream house. I have some global parameters to perform translation, rotation, scaling, and shearing. Everytime the function is called, it uses to jqery and the css function is used to display the effects. I also have a reset option.

For Javascript, I used the jQuery library. And for CSS styling, I used some bootstrap. With jQuery helped me to change the website color and also helped to merge multiple mouse events and functions with ease.

Week 2:

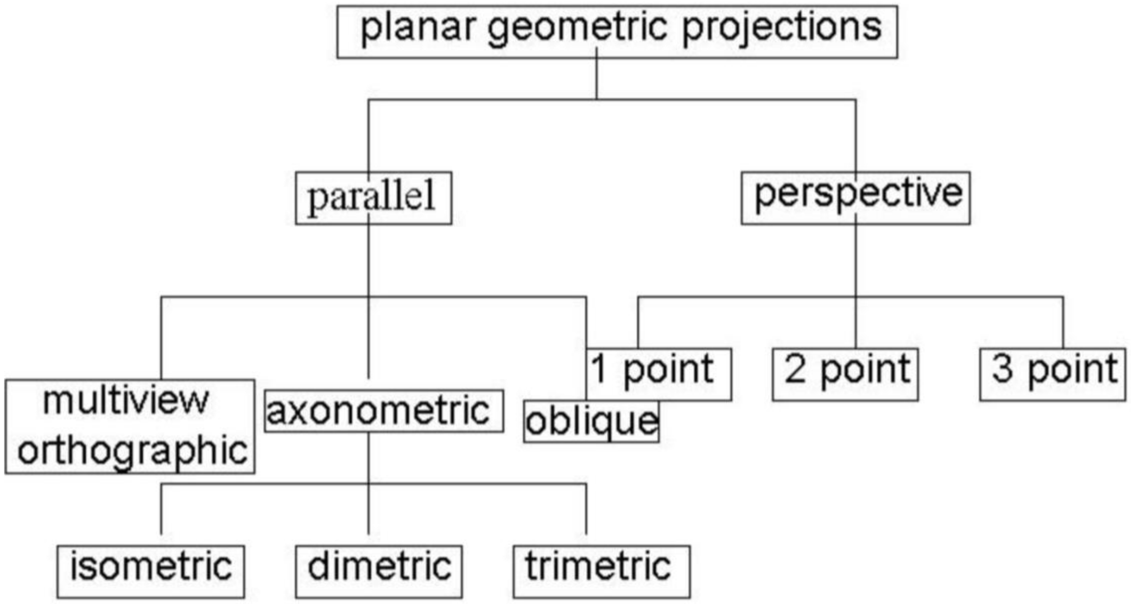
Please refer:
http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v2/2dTransformations.html
I created the full working model of top view of my dream house. Select R to rotate the view; S for scaling, T for translation; SX for Skewing X-axis; SY for Y-axis Skew and Reset to enter the state representing the natural state.

Top View:



Planar geometric projections:

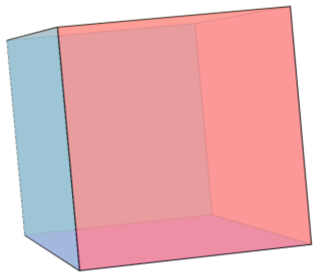
I also added a html page to display various planar geometric projections for a cube which we studied in class. In order to view, please refer: http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v2/projections.html



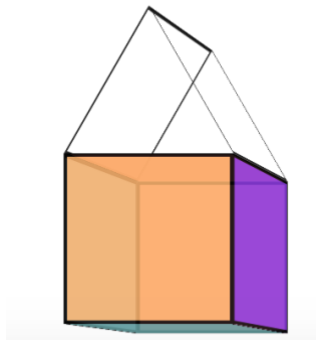
I added the top view of my dream house and the various projections of a cube. 2dTransformations.html deals with all the 2 Dimensional Transformations of dream house and Projections.html has the various Geometric Projections of cube.

Week 3:

1) I created a 3D Transformation of a cube. For that please refer 3dCube.html file



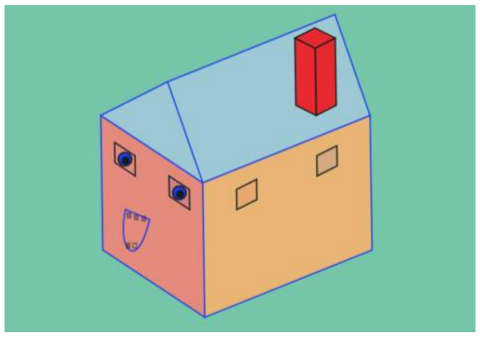
3dCube.html



3DMouse.html

2) I created a 3D Transformation of an object. The transformation changes with the mouse pointer. To view please open the 3DMouse.html file.

3) Created a floating object for a hut. To view please refer 3DFloatingHouse.html file.

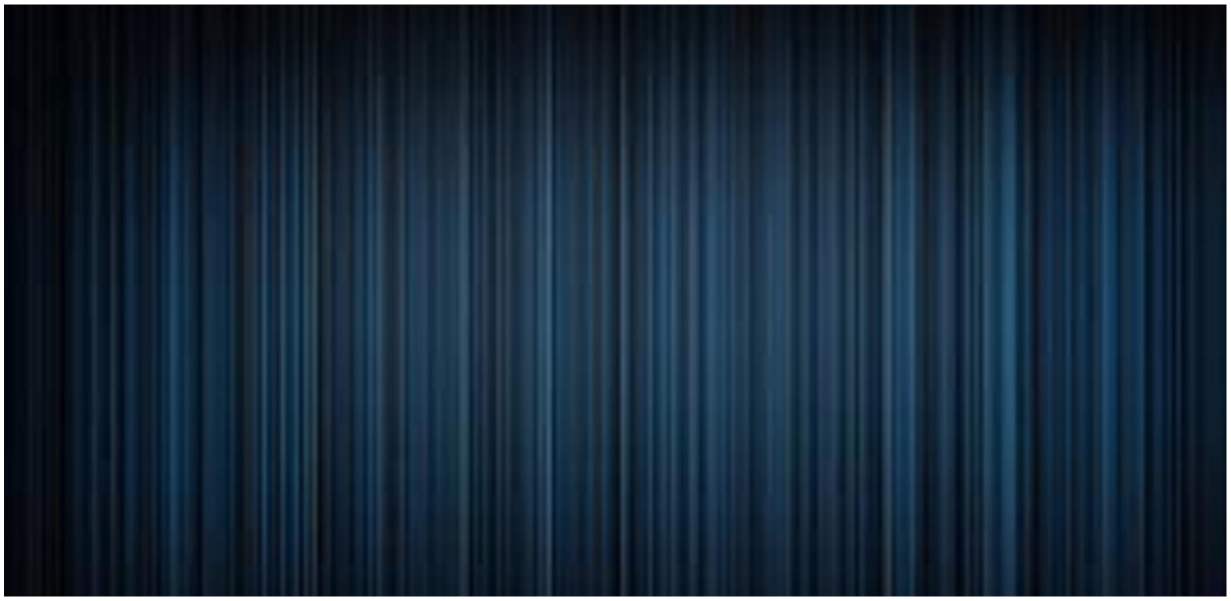


3DFloatingHouse.html

4) Implemented 3D View of a room. The view changes with the movement of mouse. To view please refer room.html file. This is a 3D conversion application.

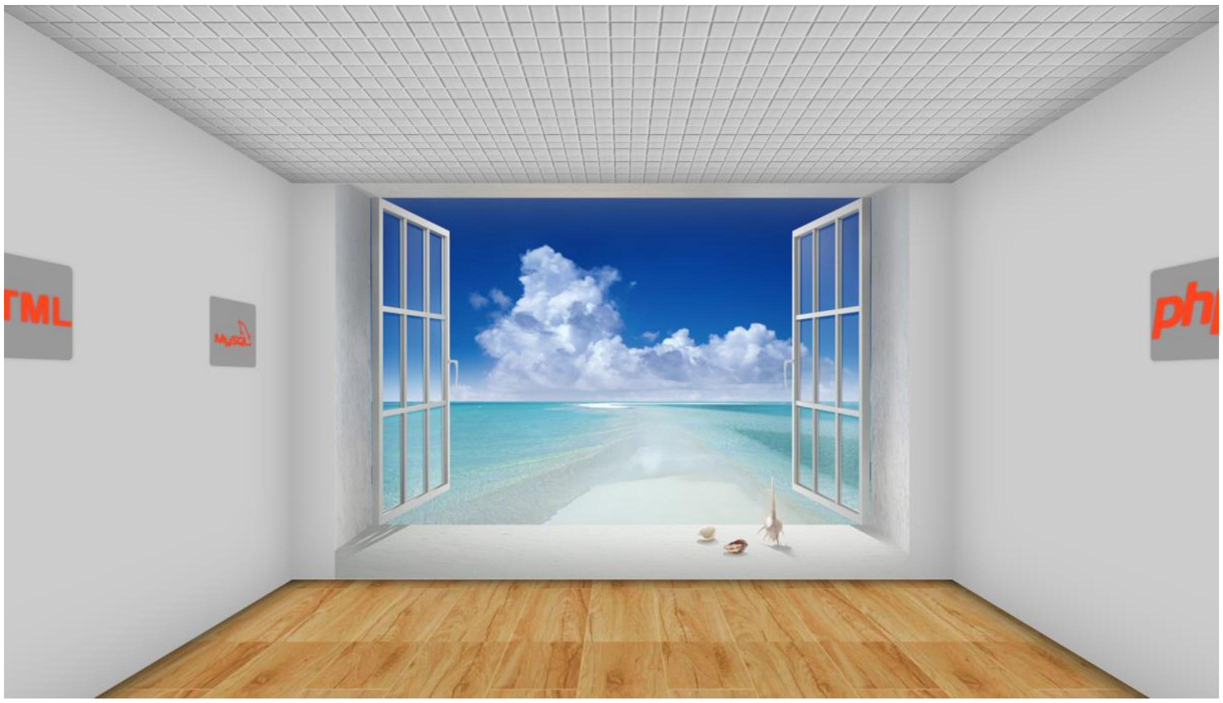
Some of the 3D conversion properties are:
1. Transfrom
2. Transform-origin
3. Perspective
4. Perspective-origin
5. Transform-style
6. Backface-visibility

The initial view looks something like:



room.html

This is nothing but a room’s wall.
Click anywhere on the wall/screen and you will be redirected inside the room.



The inside view of the room

I used Polar and Cartesian coordinate conversion formulas. The Polar coordinates to Cartesian coordinates:

- θ represents the zenith angle theta and φ stands for azimuth phi
- var x=radius*Math.sin(θ)*Math.cos(φ);
- var z=radius*Math.sin(θ)*Math.sin(φ);
- var y=radius*Math.cos(θ);
- var r=Math.sqrt($x^2+y^2+z^2$);
- var θ=Math.atan($\sqrt{}x^2+y^2/z$)
- var φ=Math.atan(y/x)

The concept of vector is used. Vectors are directional and directional. The modulus of the vector, referring to the size of the vector ($M=\sqrt{}x^2+y^2+z^2$). The unit vector refers to a vector with a modulus equal to 1.

- x=x-axis vector/M
- y=y-axis vector/M
- z=z axis vector/M

I used css3 for rotate3d(x,y,z, angle), where x,y,z refer to the unit vector of each axis and the angle refers to the angle of rotation.
The formula conversion was obtained for angle and radians

- Angle = radians*180/PI
- radians = angle*PI/180

Further, some Javascript trigonometric functions are used to obtain sin\cos\tan\asin\acos\atan\atan2. My code only accepts radians as parameters. Math.atan2(y,x) is used to accurately calculate the angle of each quadrant

Transition
Before css3, all the animations in the browser were completed instantly.

```
  animate({property:final value},5000,ease,function(){})
```

Select one or more transformations in css3, the default is immediate, add transitions. If you want to apply the transition characteristics of an element, add a transition attribute to an element

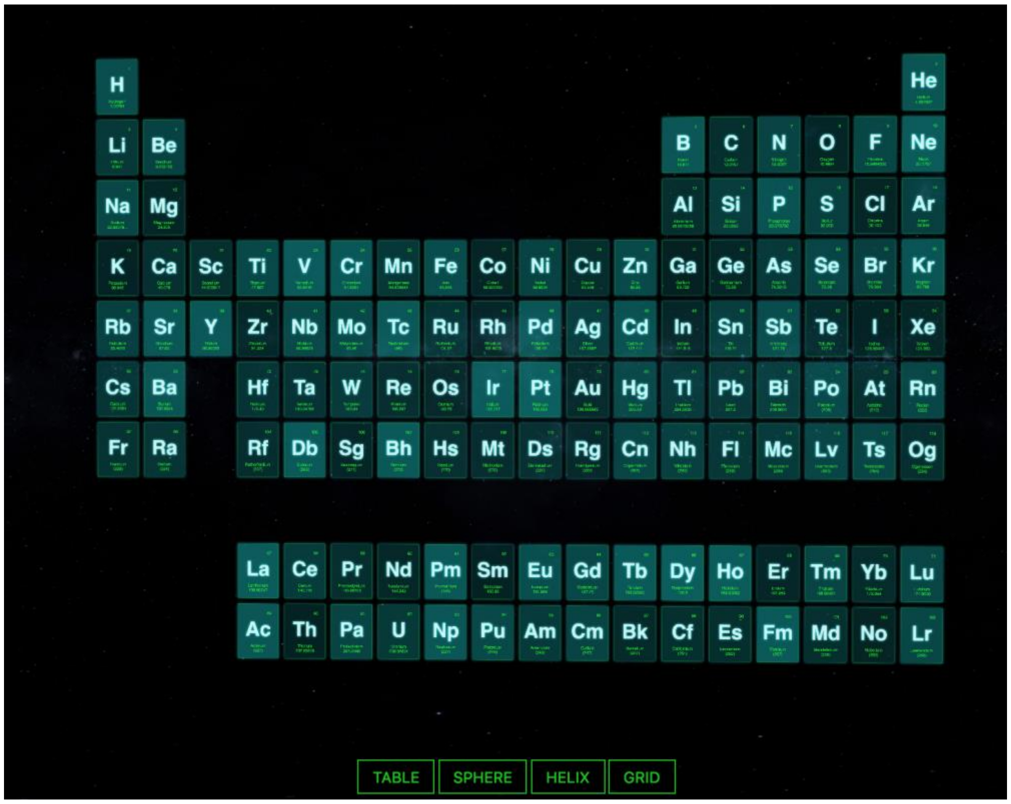
- Add at least two attributes to the transition. I added transition-property and transition-duration.
- There is a property to specify the animation (transition-timing-function).
- There is also an attribute(transition-delay) that specifies the time to wait for the transition.
- Listening to the completed state of the transition via the webkitTransitionEnd event.

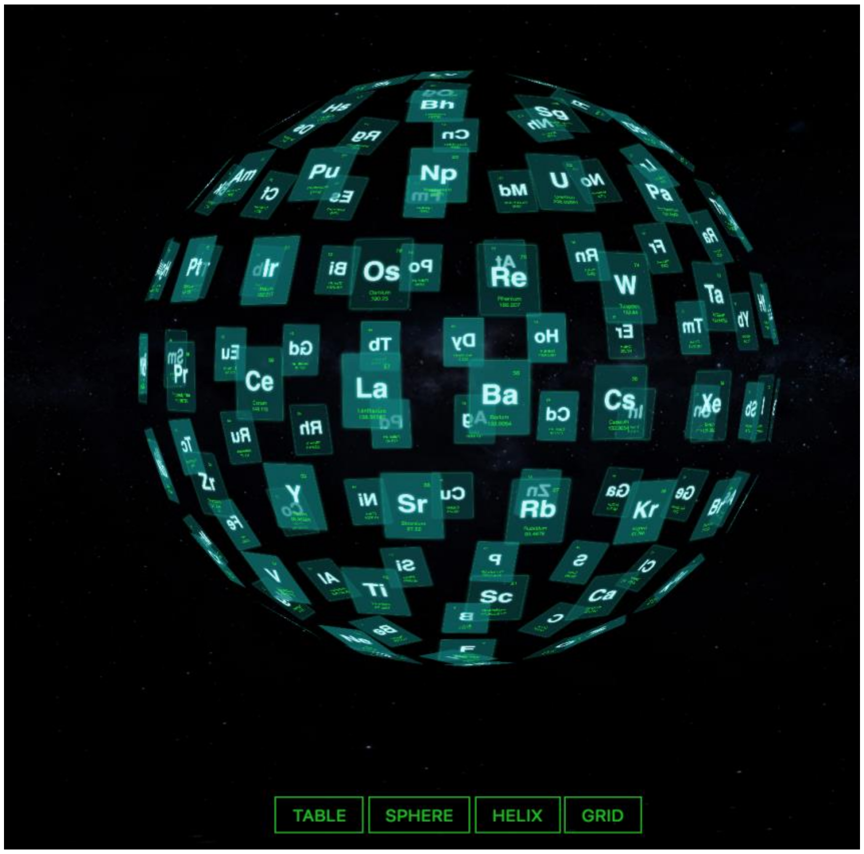
Week 4:

This week I implanted three pages using Three JS. All the JavaScript files used are present in js/ folder. Apart from this, I used audio tag to provide some nice background music to both the pages.

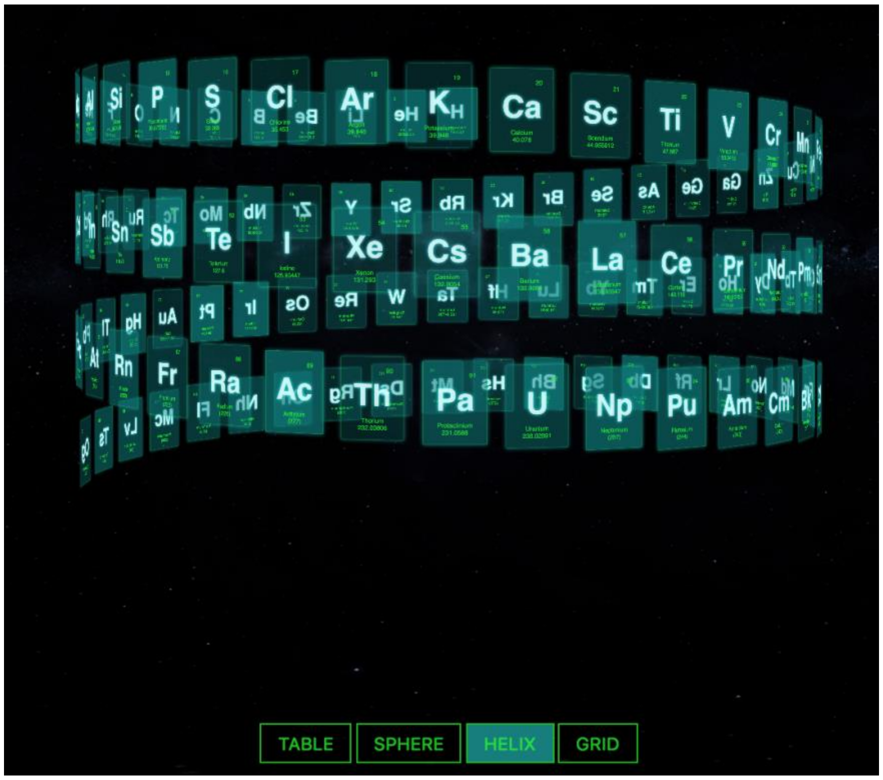
Page1: Periodic Table

The Tags used in creation of this webpage are: HTML5, javascript, canvas, three.js, animation, colors. The script files used here are: three.js, tween.min.js, TrackballControls.js and CSS3DRenderer.js All the periodic tables are saved in a variable called table. And with the help x, y and z axis I'm displaying the periodic elemts with different divs. Further, I used vector variables for displaying the data in table, sphere, helix and grid form.

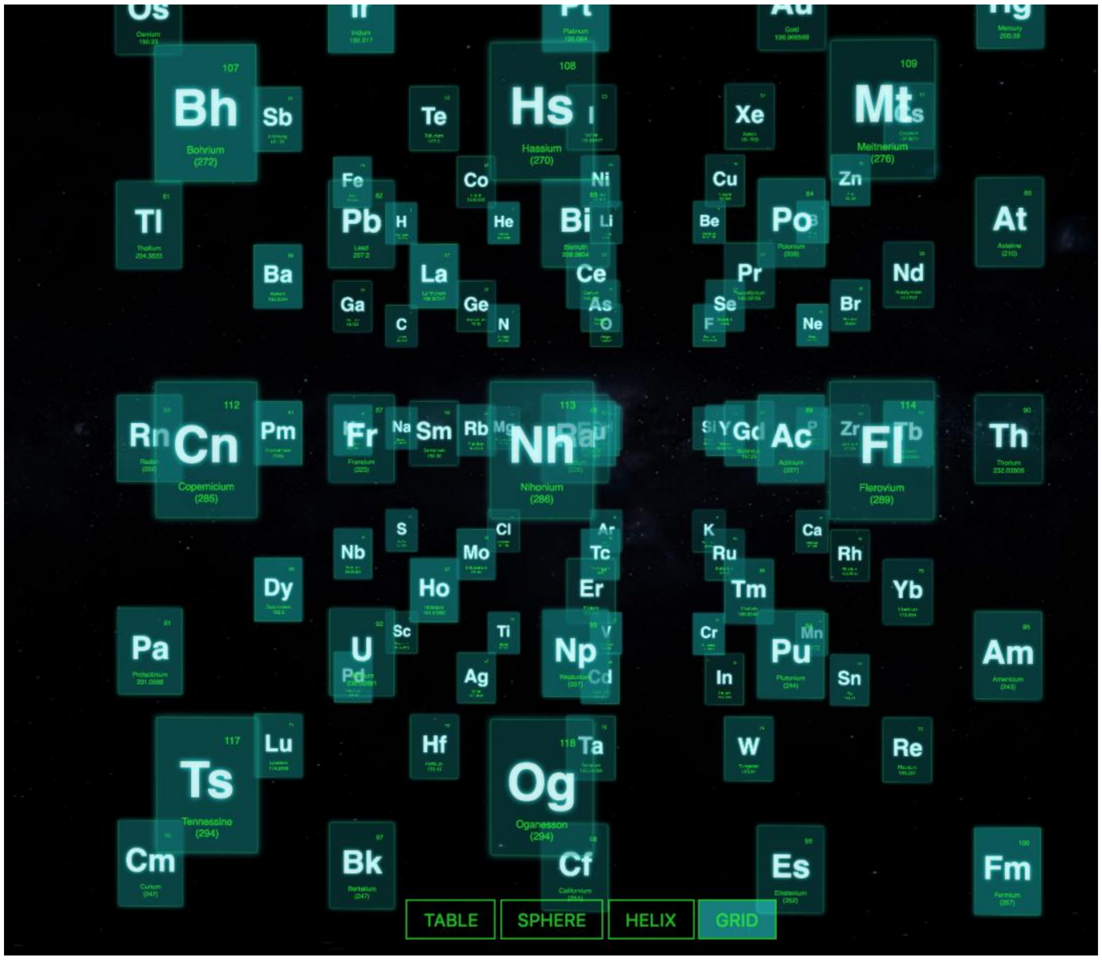

H He
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
Cs Ba Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
Fr Ra Rf Db Sg Bh Hs Mt Ds Rg Cn Nh Fl Mc Lv Ts Og
La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu
Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No Lr
TABLE
SPHERE
HELIX
GRID
Table View



Sphere View



Helix View



Grid View

Page 2: 3D Animation Lines for animated background color

The Tags used in creation of this webpage are: HTML5, javascript, canvas, three.js, animation, colors.

I took a prebuild three.js [example](http://threejs.org/examples/#canvas_lines) and put a fancy animated background color, adjusted some parameters like number of lines, perspective and colors. Now is ready to use for everyone. Check out the online demo at http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v4/3DAnimationLines.html. The script is fully configurable (colors, lines, opacities, perspectives).

HTML
```
<!-- Main library -->
<script src="js/three.min.js"></script>

<!-- Helpers -->
<script src="js/projector.js"></script>
```

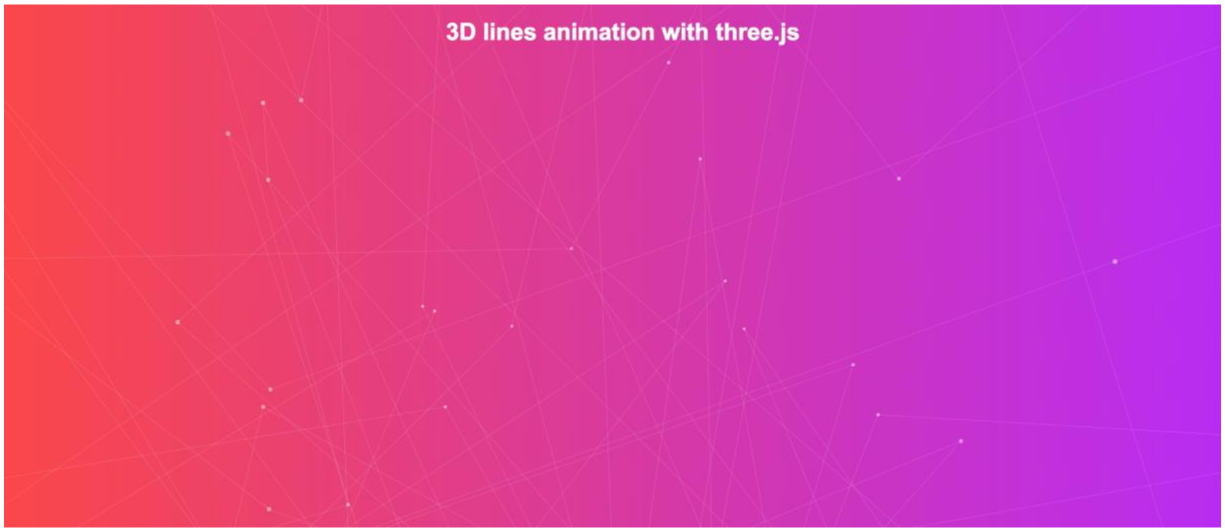
```
<script src="js/canvas-renderer.js"></script>

<!-- Visualization adjustments -->
<script src="js/3d-lines-animation.js"></script>

<!-- Animated background color -->
<script src="http://cdnjs.cloudflare.com/ajax/libs/jquery/2.0.2/jquery.min.js"></script>
<script src="js/color.js"></script>
```
I'm using three.min.js as main JavaScript Library. For Visualization, 3d-lines-animation.js file is used. And then for frequent animated background color, color.js file is used. The Background color keeps changing and 3D Lines can be explored by dragging the mouse onto them.

Layout

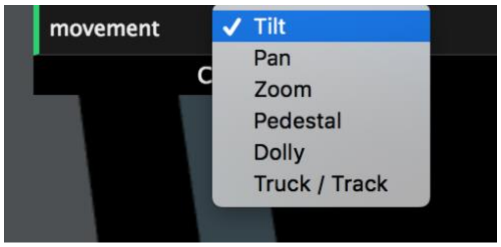
```html
<div class="canvas-wrap">
   <div class="canvas-content">
      <h1>Hello world</h1>
   </div>
   <div id="canvas" class="gradient"></div>
</div>
```



3DAnimationLines.html

Page 3: Basic Camera Movement using three.js and WebGL

The JS file used for for the understanding of the Basics of Camera Movement is Detector.js and three.min.js files. The Tags used in creation of this webpage are: HTML5, javascript, canvas, three.js, animation, camera. I gave the drop-down at the top right to switch between tilt, pan, zoom, pedestal, dolly and truck/track.

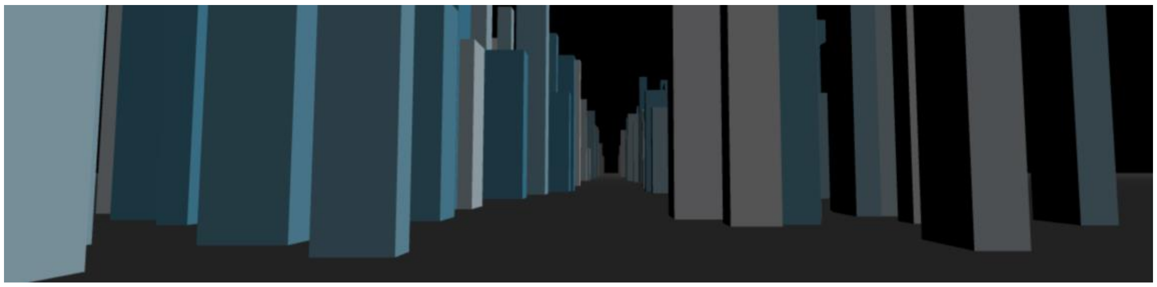


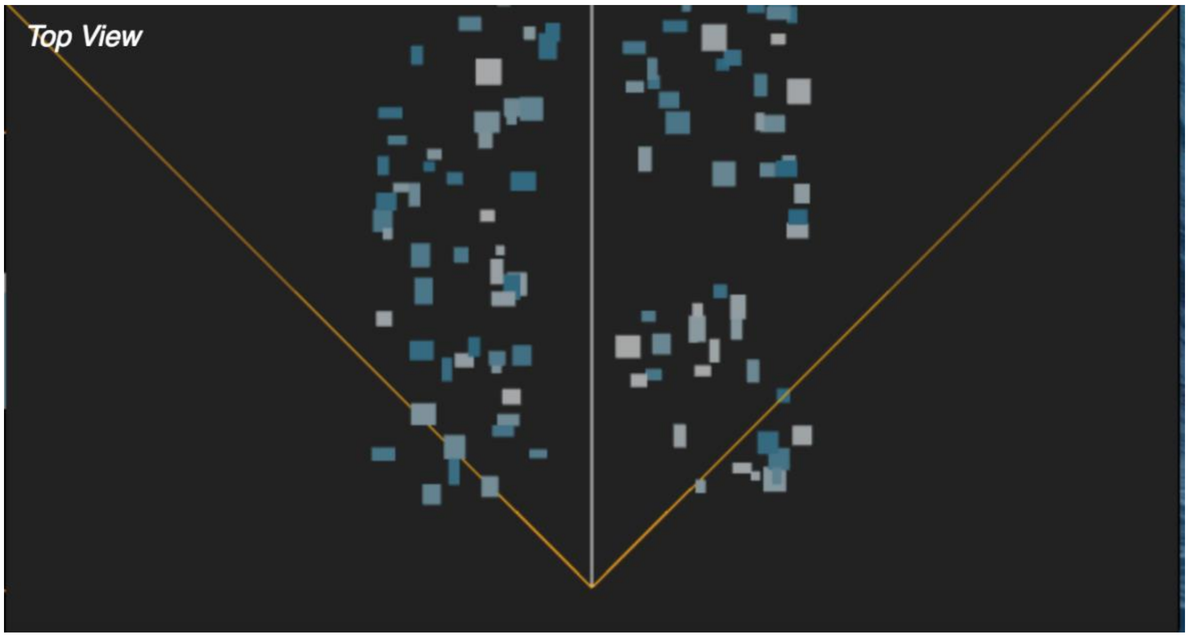
Movement DropDown

For Demo, please click: http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v4/cameraMovement.html.

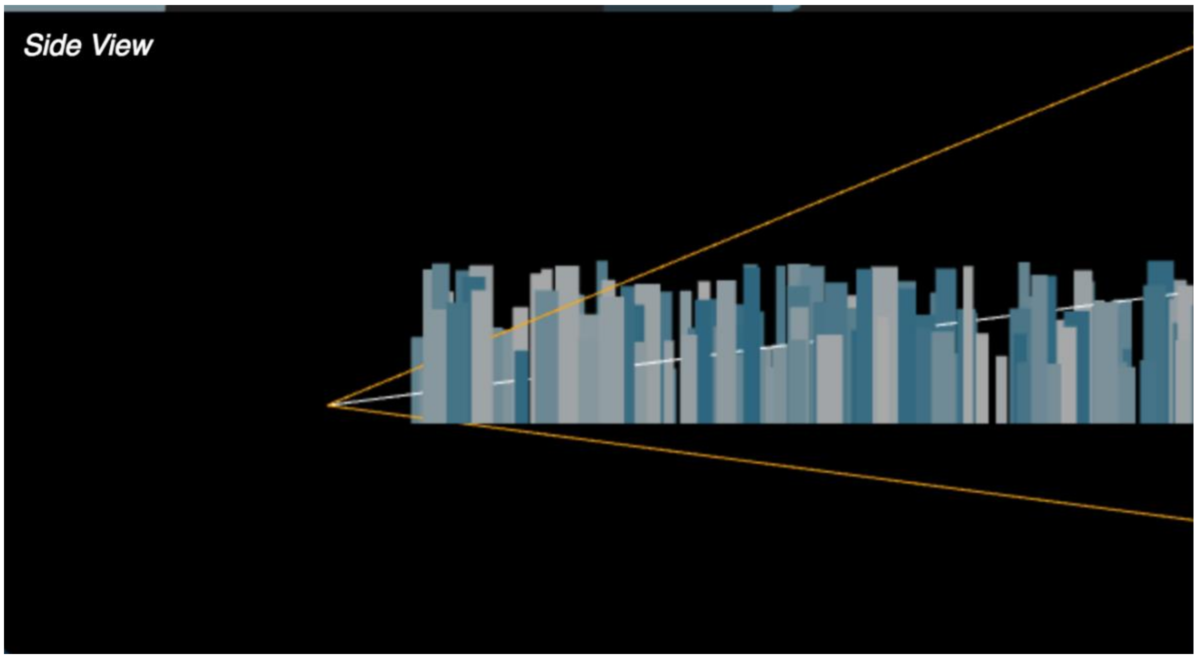
Often times with real-time graphics demos and "creative coding" apps we default to a fixed camera, an orbiting camera, or an interactive Maya-style camera (orbit, pan and zoom or dolly with the mouse). This is a demo of some of the basic camera movements I learn and use in film. My background is not in film, but I think we can learn a lot from film when we work on our virtual worlds and experiences.

I did this because I felt that, everyone should know the difference between zoom and dolly and the visual difference that makes in-camera.





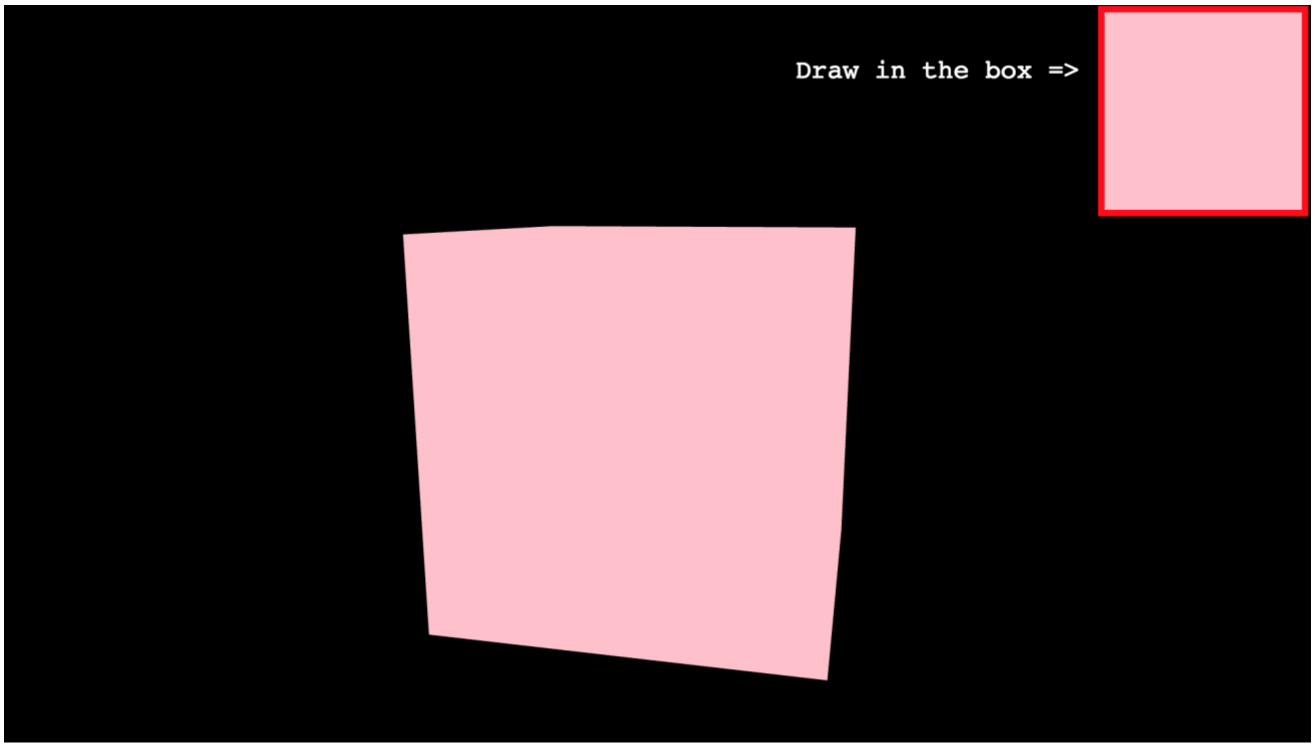
Top View



Side View

Week 5:
This Week I implemented the following Pages:

Page 1: Texture Mapping



textureMapping.html

I am using Detecter JS for this. I am drawing the picture on canvas and later the picture is reflected on to the sides of the cube. Check out the online demo at http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v5/textureMapping.html

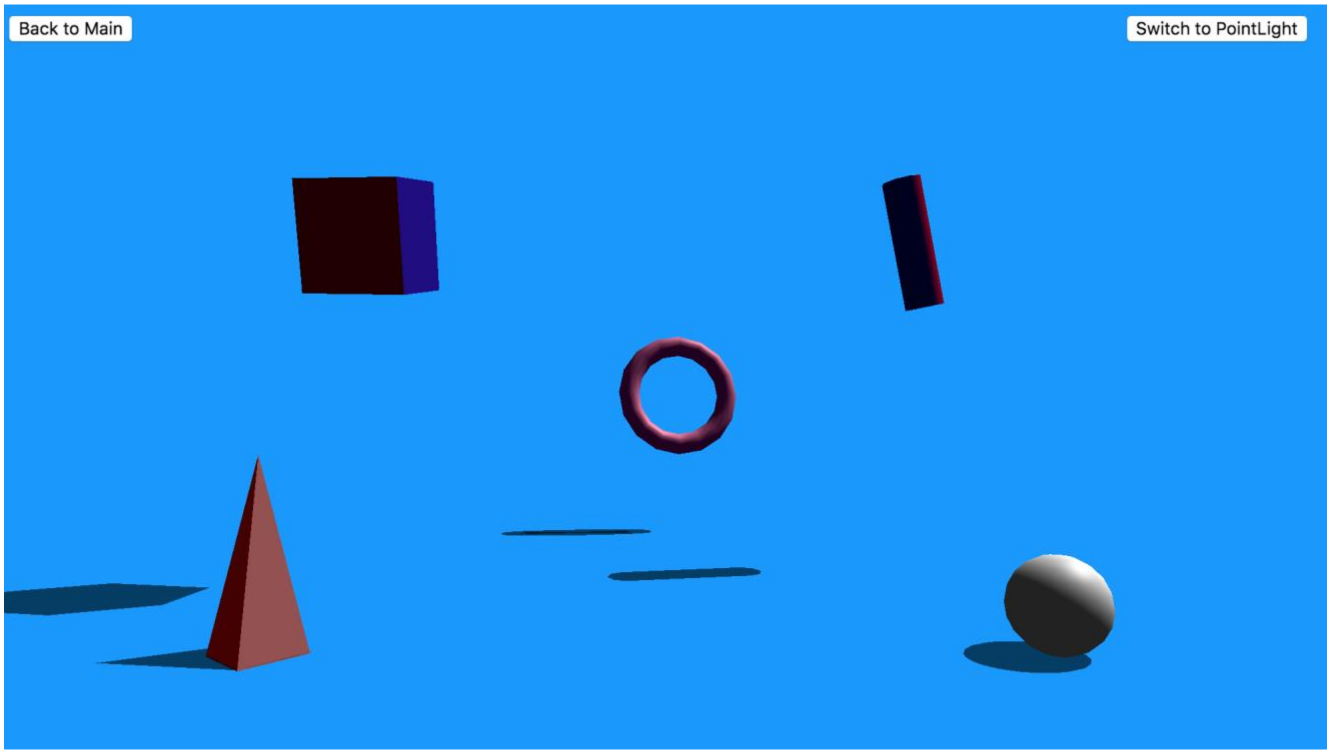
The required script files are:
```
<script src="js/three.js"></script>
<script src="js/Detector.js"></script>
```

Page 2: Shadow Mapping

I created two light bulbs, one cylinder, one pyramid, one cube and one sphere. The sphere, pyramid and light bulbs are static, they don't change their position. Whereas the cylinder and cube moves. And the shadow can be seen on the surface. Check out the online demo at http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v5/shadowMapping.html

The required script files are:
```
<script src="js/three.js"></script>
```

<script src="js/ShadowMesh.js"></script>

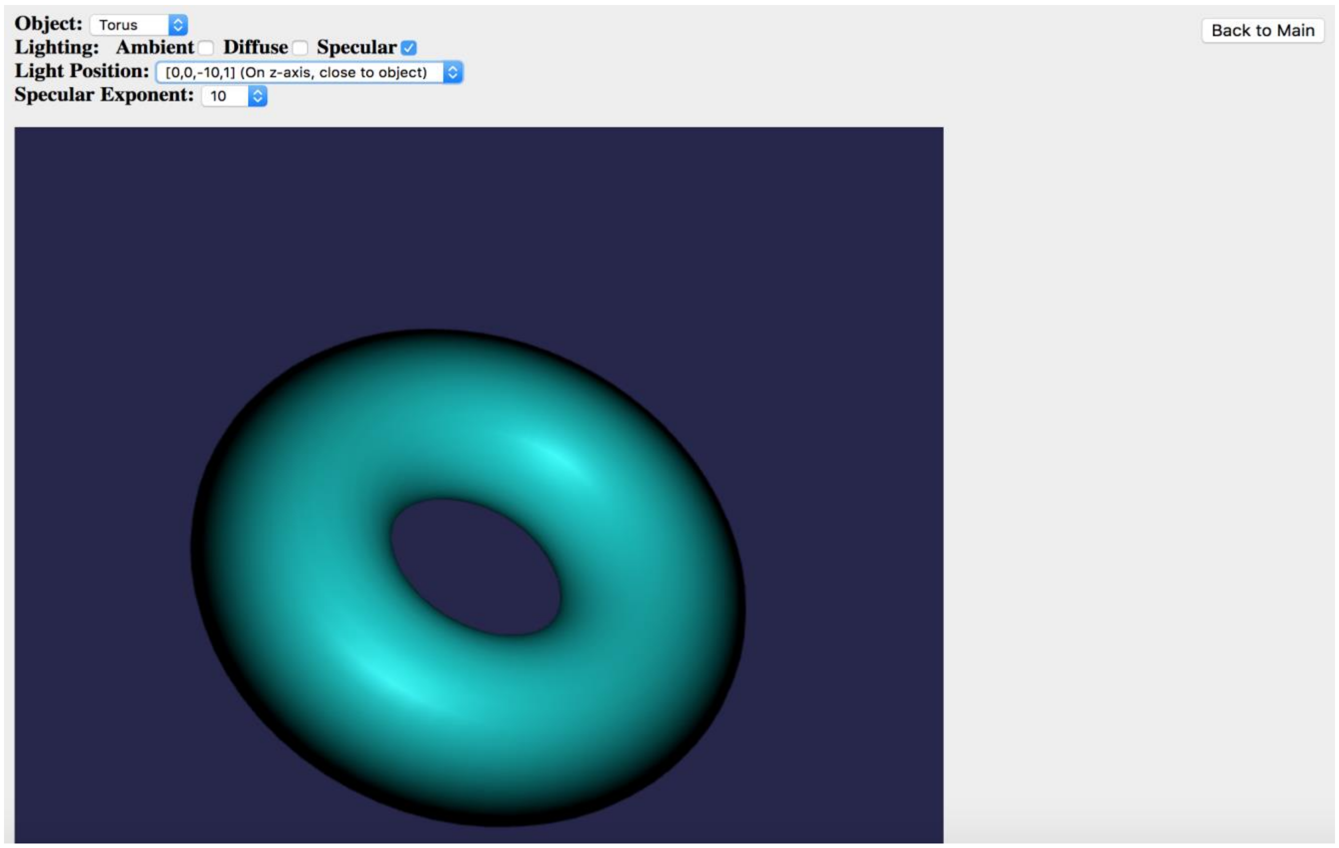


shadowMapping.html

Page 3: pa2.html

In order to get familiar with lighting which is also fundamental to creating any scene, using WebG, this is the best page to visit. In this assignment, I'm writing some code to help me understand on how the lighting equations work. Check out the online demo at http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v5/pa2.html

The required script files are:
<script src="js/gl-matrix-min.js"></script> - utility functions for operating matrices
<script src="js/trackball-rotator.js"></script> - utility functions for rotating the scene using the cursor
<script src="js/model.js"></script> - file describing the models in IFS format
<script src="js/pa2.js"></script> - functions used to draw the scene.


Object: Torus

Lighting: **Ambient** ☐ **Diffuse** ☐ **Specular** ☑

Light Position: [0,0,-10,1] (On z-axis, close to object)

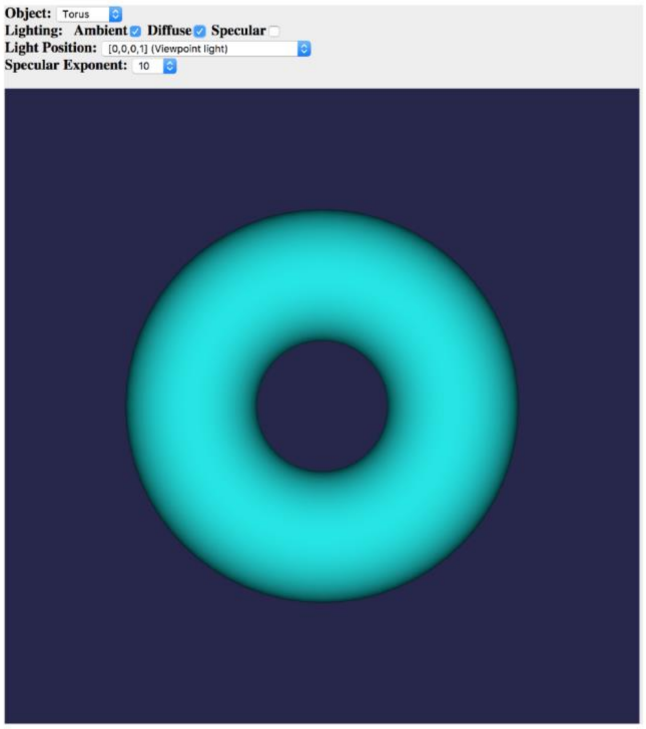
Specular Exponent: 10

Page 4: BatMan.html

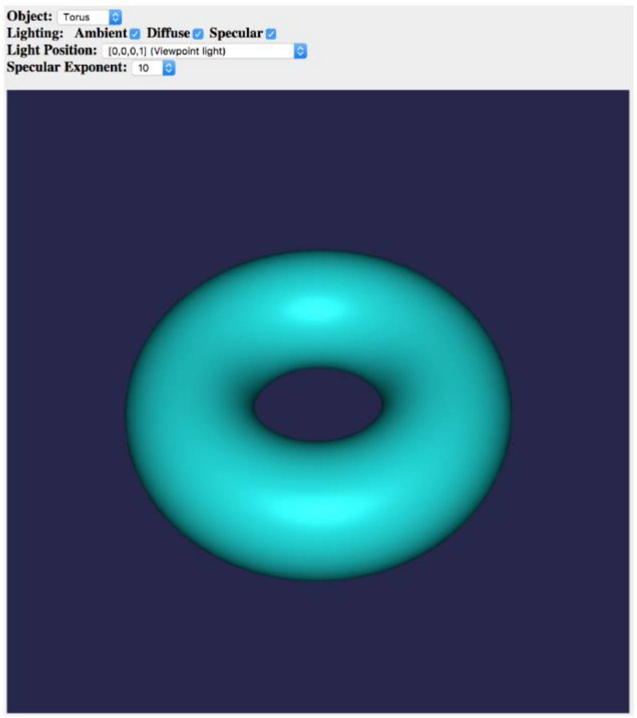
This is for extra credit. I hope you enjoy it! Check out the online demo at
http://www.cs.uml.edu/~asharma/427546s2018/finalProject/finalProject_v5/batman.html

The required script files are:

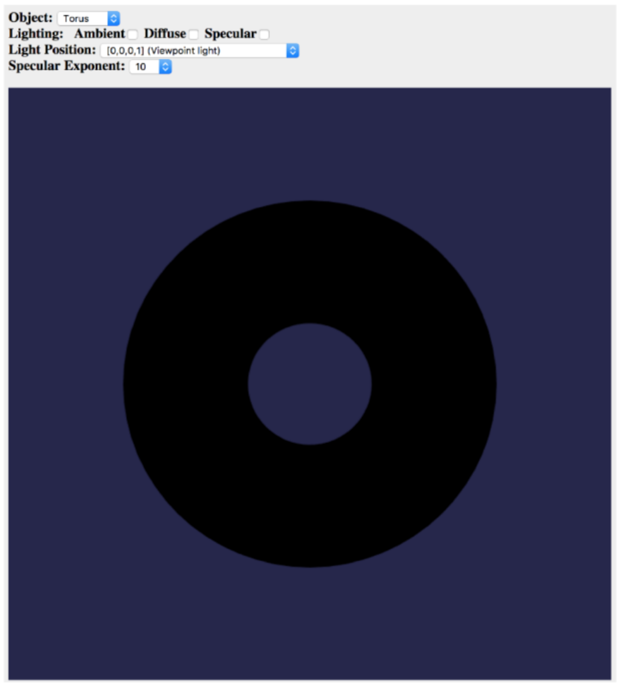
```
<script type="text/javascript" src="js/three.js"></script>
<script type="text/javascript" src="js/stats.js"></script>
<script type="text/javascript" src="js/dat.gui.js"></script>
<script type="text/javascript" src="js/d3-threeD.js"></script>
<script type="text/javascript" src="js/OrbitControls.js"></script>
```


Object: Torus
Lighting: **Ambient** ✓ **Diffuse** ✓ **Specular** ☐
Light Position: [0,0,0,1] (Viewpoint light)
Specular Exponent: 10

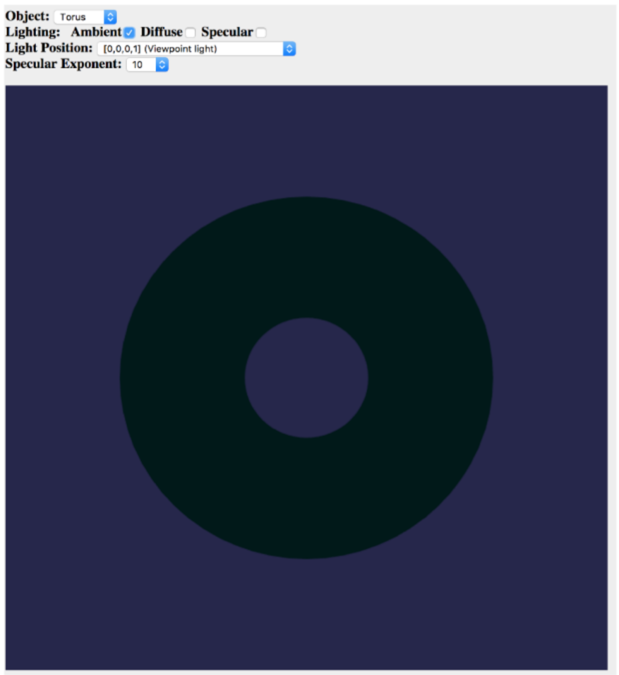
Torus with ambient and diffuse lighting.


Object: Torus
Lighting: **Ambient** ✓ **Diffuse** ✓ **Specular** ✓
Light Position: [0,0,0,1] (Viewpoint light)
Specular Exponent: 10

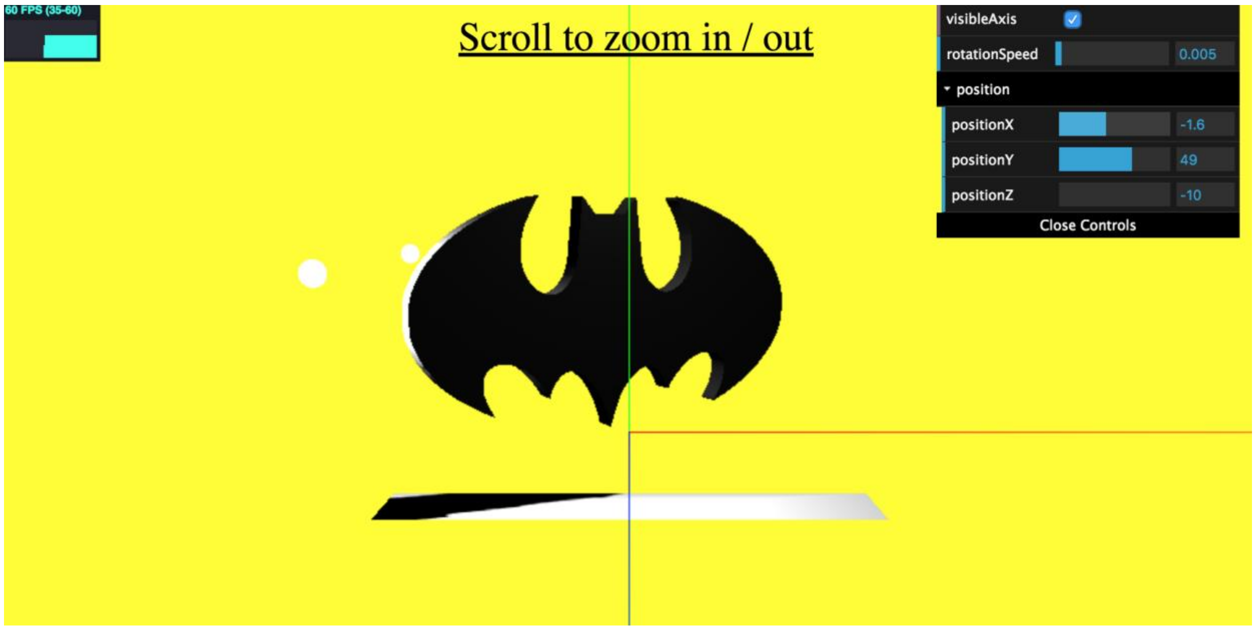
Torus with ambient and diffuse and specular lighting.


Object: Torus
Lighting: **Ambient** ☐ **Diffuse** ☐ **Specular** ☐
Light Position: [0,0,0,1] (Viewpoint light)
Specular Exponent: 10

There is no light. Hence you see a black torus


Object: Torus
Lighting: **Ambient** ✓ **Diffuse** ☐ **Specular** ☐
Light Position: [0,0,0,1] (Viewpoint light)
Specular Exponent: 10

Torus with only ambient light. Cyan colored torus is seen.



visibleAxis ✓
rotationSpeed 0.005
▾ position
positionX -1.6
positionY 49
positionZ -10

Batman.html

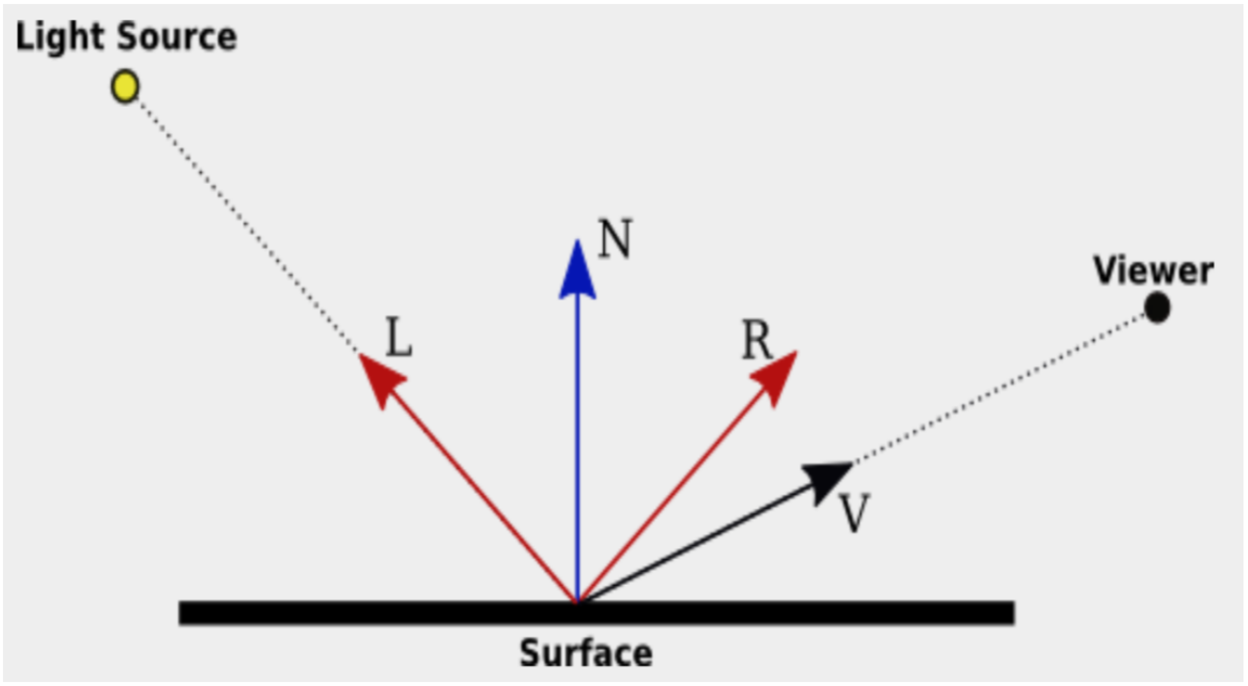
Note: WebGL runs within the browser, so is independent of the operating and window systems. You may finish the assignment using any operating system you like, e.g. Windows, OSX or Linux. Programming language: The assignment will be implemented in JavaScript.

Before working on this week's tasks. I learnt to implement 3 different kind of lighting, namely ambient, diffuse and specular using the Phong shading model. I completed both vertex and fragment shader in pa2.html in order to draw object.

Below is a schematic of a point on a surface which is being illuminated by a point light source where,
L = Light is the direction of the light
N = Normal of the point on the surface
R = Reflection of L about N.
V = viewing direction

Light at a point = Ambient Light + Diffuse Light + Specular Light.
Ambient Light = K" $*$ diffuse_color
Diffuse Light = K/ $*$ diffuse_color $*$ max3(N. L), 0;
Specular Light = K= $*$ specular_color $*$ max $((R.V), 0)$A
R = $2N(N.L) - L$
Use K",K/,K= 0.1, 0.4 and 0.4 respectively. n is the specular coefficient, which the user can change. Note, although the above light equation is not complete as it does not account for the distance from light source (light intensity decreases as we move away from the light source), to keep things simple we would only implement above equation in this programming assignment.

Shaders (Vertex, Fragment, etc) are small programs which are executed on the GPU. They are mainly responsible for determining the color of each object. However, they can be used for many different tasks. In this assignment we will be using two shaders namely vertex and fragment shaders to compute the lighting of our scene. The next thing I used is GSLS, a short term for OpenGL Shading Language. This is used to write code into any shader. It is similar to C/C++. Check out the references #1 and #4 to understand more on how to write code in a shader.



Useful references:

1) WebGL tutorial:
http://learningwebgl.com/blog/?page_id=1217

2) JavaScript
https://developer.mozilla.org/en-US/docs/Web/JavaScript

3) JS style guide:
https://google.github.io/styleguide/javascriptguide.xml?showone=Comments#Comments

4) WebGL Shaders and GLSL (GL Shading Language)
https://webglfundamentals.org/webgl/lessons/webgl-shaders-and-glsl.html

5) Three JS
https://threejs.org/examples/