COMPUTER GRAPHICS
PROJECT REPORT

Submission this Week

- I have used Three JS to Simulate 3D objects with change of scale and Transformation and Shear a Object .

The Modules I have Submitted this week :

- Shearing : I have created a Module to Shear a Primitive Object by using Shear Matrix available in the Three JS build.
- Transformation : I have used Transform Controls to transform the object in the canvas which is projected.I have used transform controls to transform all primitive objects in a set plane.
- A 3D Mapped(Texture and Bump) planetary System.Have used Hemisphere Lights to highlight the texture and Bump Maps

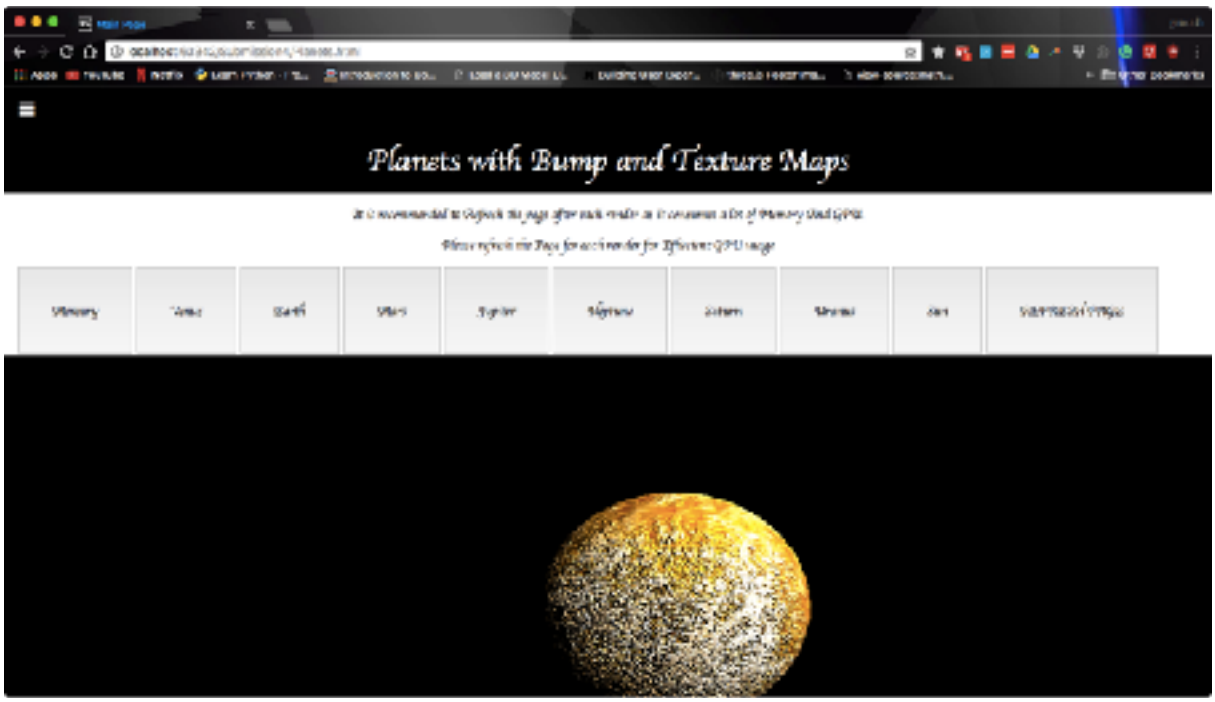
NOTE : Please refresh After Each render as it consumes a lot of Memory and GPU.

- A 3D rendering of a car Axle with the base of a car using Cylinder and Torus Geometry.

Thus Far, I have managed to complete the following Steps mentioned in the project:

1. Modeling: create and store a 3D object (Cube, Sphere , Tree(first Submission), etc.)
2. Viewing: view your created object from multiple views. I have used orbitcontrols.js and Transform.js which is available through the three JS Build to help us rotate and used Mouse and Keyboard controls to view the rendered 3D object through various angles.
3. Light Sources : I have used Ambient Light in most of my implementations (eg: texture with earth map blended to light a view) and also hemisphere lights in solar systems.
4. Create Texture/ Bump Maps : I have use texture and bump Maps in this weeks Submission to render a 3D Earth and the solar system .
5. Scaling and Transformation of Objects : I have tried my best to scale and transform the 3D object using dat GUI in this weeks submission.
6. Rotation of Objects : I have been able to achieve rotation of objects in all my implementations of the 3D rendering in my submissions.

SCREENSHOTS:



Previous Week's Submission :

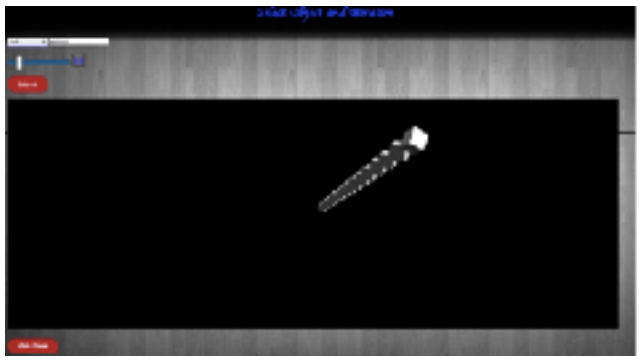
Week 1 :

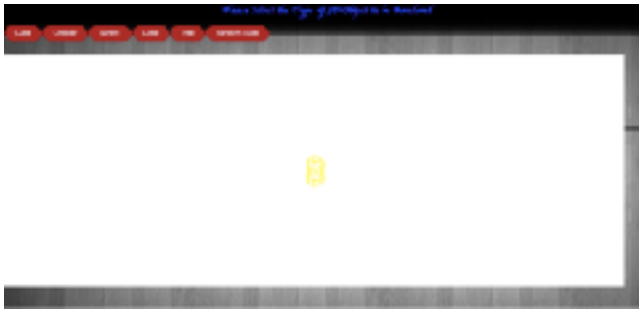
Modules Submitted on Week 1 :

- Rendering of Primitive 3D Objects
  A. Cube.
  B. Sphere.
  C. Cylinder.
  D. Cone.
- Rendering a Primitive Tree with Cylinder as the Stem and Cone as the Leaves.
- Multiples object Renderer:
  A. Cube
  B. Sphere
  C. Cone
  D. Cylinder
  E. User Input on the Number of Iterations

Details of Implementation :

I. This submission I have tried too implement the basic rendering of primitives using Three JS.
II. This Implementation served as a Introduction to Three JS and its API's.
III. I have used to basic API's such as Sphere , Cube, Cone and cylinder to construct the Assignment.
IV. I also tried to implement a basic structure of a tree which can be modeled using primitives and apply rotation and elevations in the submission.
V. I tried to give user Feedback and Input in the submission my rendering and position multiple renders as the per user Input.





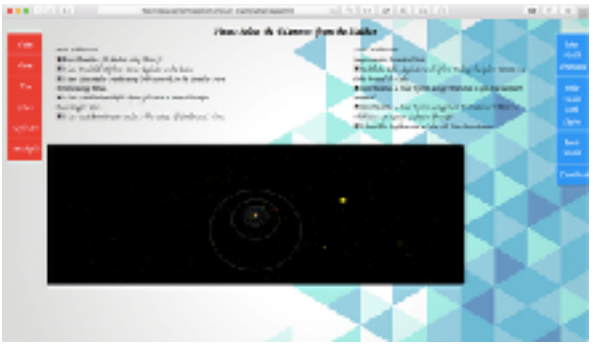
Week 2 :

Modules Submitted on Week 2 :

- A 3D Cube Rendered with User Co-ordinates with Dynamic HTML DOM Manipulation using Javascript and Three JS to Accomplish the Task.
- A primitive Rendering of a Solar System with Spheres Placed in a orbit and with the Orbits Drawn Around the sphere representing a planet in our Solar System.
- A 3D rendering of a solar system using MeshLambert Material in three Js and with the orbits rotating to form a constructive basic Solar system.
- A Basic rendering of a cube , Cylinder and 2 Spheres with Animation.

Details of Implementation :

I. 1. Used the Cube Geometry function readily Available in the Three JS API to Render a Cube.
II. I have created Node Elements which are number Boxes to get user inputs for the width, length and height of the cube to rendered.
III. I had to learn how to inject HTML elements dynamically through W3 Schools.
IV. The infected Elements are modeled and Styled using CSS to be placed at particular Positions.
V. The 3D rendering is done once the user Enters all the three fields.
VI. There are No checks placed to force the user to enter all Inputs.
VII. Used Primitive renderings of a sphere to pose as the planets in our solar system,
VIII. I have user particle geometry and Randomly spread the particles which are cubes to form Stars surrounding my Solar System.
IX. I have manually set the Position of the orbits and Planets for a Better View of the Rendering .I have created a Three Group to add All the Planets .I used Mesh Basic material in this rendering which does not have any effects when lights are introduced using the API.
X. I have user particle geometry and Randomly spread the particles which are cubes to form Stars surrounding my Solar System.I have manually set the Position of the orbits and Planets for a Better View of the Rendering . I have created a Three Group to add All the Planets .
XI. I used Mesh Basic material in the above rendering which does not have any effects when lights are introduced using the API.So, I used Mesh Lambert Material which has lighting features on them in the second rendering.
XII. I have Made the 2 Spheres move around the cube and Cylinder. This was the basic render which helped me create the solar System.



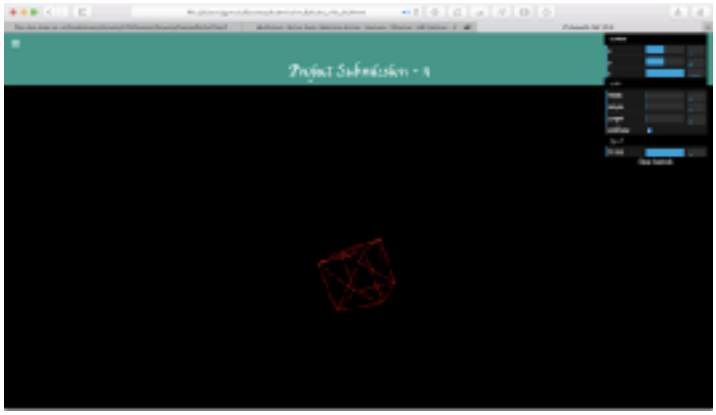
Week - 3

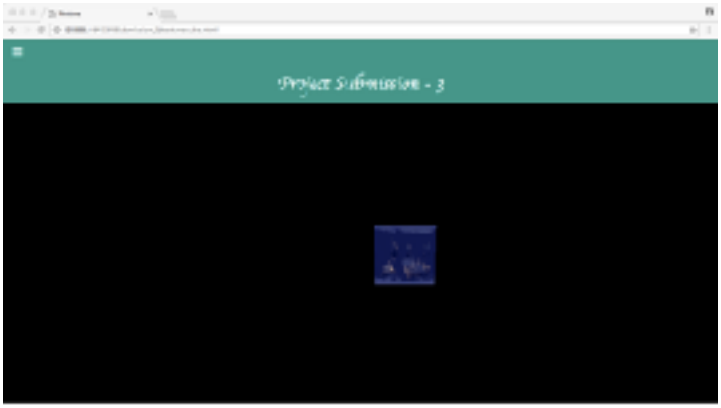
- I have used DAT GUI with Three JS to Simulate 3D objects with change of scale and Transformation.
- DAT GUI is a graphical user interface which is a light weight controller in javascript library to provide a User interface which can help scale and transform or perform operations on the Three JS rendered simulation.

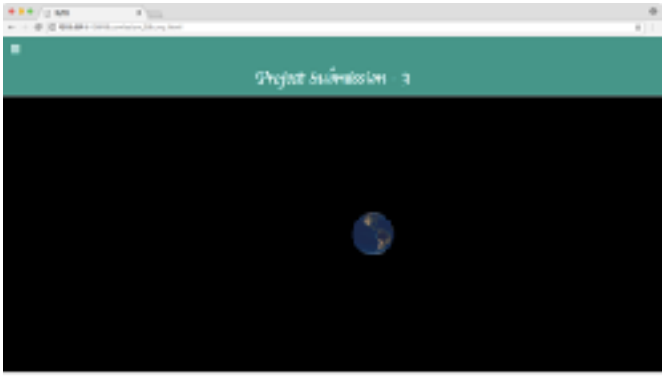
Modules:

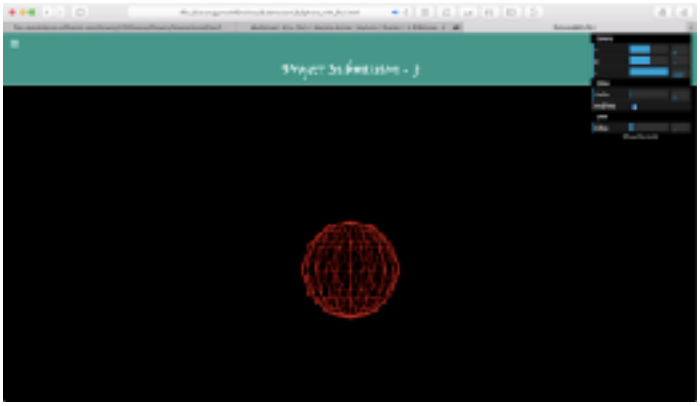
- Cube Rendered in three JS with GUI. It has options of changing width, height and length of the cube along with scaling the cube in X,Y,Z directions.
- Cone Rendered in three JS with GUI. It has options of changing width, height and length of the cube along with scaling the cube in X,Y,Z directions.
- Sphere Rendered in three JS with GUI. It has options of changing width, height and length of the cube along with scaling the cube in X,Y,Z directions.
- Cylinder Rendered in three JS with GUI. It has options of changing width, height and length of the cube along with scaling the cube in X,Y,Z directions.
- Simple Animation of a cube and Sphere with Bouncing and Rotation Speeds controlled through the GUI.
- A 3D Mapped Sphere resembling a Earth. I have Blended the texture of Earth onto a Sphere with Orbit controls to turn and view the Sphere in elevated Angles.
- A 3D Mapped Cube. I have Blended the texture of Earth onto a Sphere with Orbit controls to turn and view the Cube in elevated Angles. I have used the same texture that has been used on the sphere.
- A 3D Mapped Sphere resembling a Earth. I have Blended the texture and bump Maps of Earth onto a Sphere with Orbit controls to turn and view the Sphere in elevated Angles. I have added bump map of the earth to have a better rendering Earth.

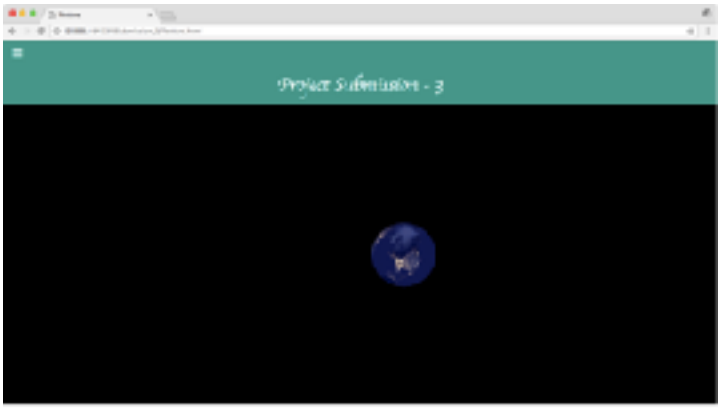
SCREENSHOTS:

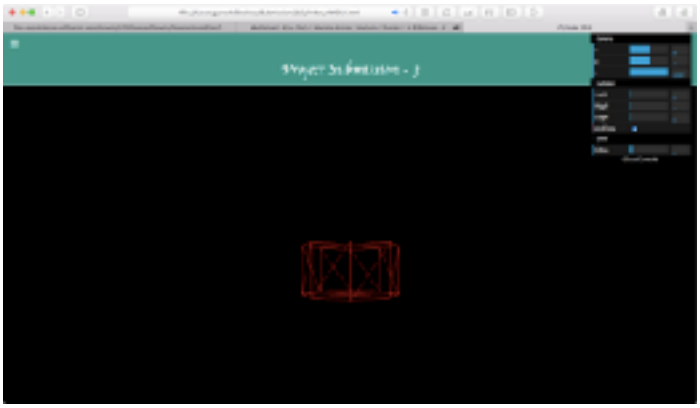

Project Submission - 4


Project Submission - 3


Project Submission - 3


Project Submission - 3


Project Submission - 3


Project Submission - 3

SOFTWARE REQUIREMENTS :

- Brackets (Text Editor for HTML,CSS,JS)
- THREE JS Build.
- DAT GUI Build.
- Chrome Web Browser.

INSTRUCTIONS TO EXECUTE:

Please Have Internet connection as the Javascript for Three JS is a http Link .
Please view the Website In Full screen.
Please Run the Solar System and refresh the system as it requires a lot of GPU.

REFERENCES :

- https://threejs.org
- https://www.youtube.com/watch?v=YKzyhcyAijo
- https://www.youtube.com/watch?v=lshPMbN5ws8
- I have used various web resources.
- Three Js Beginner tutorials to construct a Cube on Youtube.
- I have used the Official Three JS module from the Three JS website to download the Build and orbit control Files.
- I have visited three JS website to find out about the materials and different types of Geometry's available to construct the Website.
- I have Used CSS tricks.com to style my Web Page.
- I have used Stack over Flow to figure out the geometry of a orbit.
- I also used Youtube videos to learn on how to light a canvas using Three JS.