Basic Overview: The project available in this GitHub repository is a modeled scene with a variety of shapes and effects included. The shapes include a modeled head/neck, eyes, glass, and wood. The effects included are the "bloom" effect or glowing of the eyes. The particle system which slowly fills up the encapsulation of head. And the final effect is a refraction of light effect which takes place when the image is rotated. The user can control how intense the "bloom" effect is using the GUI controls.

Head Modeling: Using code taken from a GitHub repository under the MIT license. A head is modeled based on the Lee Perry Smith models included within the ThreeJS framework. The modeling procedures for this head are included within the init() and createScene() functions within the FinalProject.js file

Window Modeling: Using the PlaneGeometry methods included in the ThreeJS framework I was able to create a rectangular plane which would then be made transparent. This was considered the simplest solution for creating a window because a plane acts as surface to which the refraction can later be applied to and could make a shape that would resemble a window. One window would then be placed in front of the head and one in the back of the head.

Wood Modeling: Each wooden object within the code is simply a BoxGeometry (included in the threejs framework) object. Each was rendered to the desired height, length, and width; and then placed around the head to enclose it. With the wood being "connected" to the glass to make the entire enclosure seem sound.

Wood Texture: Using the TextureLoader class within the threejs framework, a texture was created by loading the crate gif included in the threejs framework. Once this texture was created it was then added to the mesh of the wooden objects, and they would appear wooden. This seemed the simplest way to achieve a wooden texture because it was included in the threejs framework and allowed for the mesh to be changed to another texture if needed.

Eye Modeling: The eyes were modeled using the SphereGeonmetry class within the threejs framework. The sclera, iris, and pupil were modeled as spheres and then placed where the eyes should be within the face. By modeling each part of the eye separately, there could be different colors for each part. Both eyes were generated the same.

Bloom Effect: The BloomEffect takes advantage of the UnrealBloomPass class included within the examples of the threejs framework. By attaching this pass to the composers of the scene in your project you can give the effect of glowing objects. This effect was only attached to the eyes. The challenge in implementing this task was that there needed to be a way to add the pass to only the modeled eyes and not the other rendered objects in the scene. This was solved by discovering a way to add the pass only to the eyes and not the other objects within the scene, but still including the "bloom" object in the composer so that it is included in the scene.

Refraction: By using the Refracor class included within the threejs framework, it served as an object that could be placed within the windows that would then alter the light visible within the scene. While this effect could be placed on any surface, even the wood, it served best on the

window as that was the medium through which you see the face and the window refraction effect also effects the view of the head and particle system.

Particle System: Using the Points class within the threejs framework, it allowed for the creation of a particle system of identical particle that could fill a designated area. The method createParticleSystem() creates a system of particles of size 1000 that are then place within the enclosure. However the animate function allows this method to be called based on a timer system, so 1000 particles are added to the scene for a limit of 3000 times (to avoid crashes from overflow).