Computer Graphics Assignment 2

Team

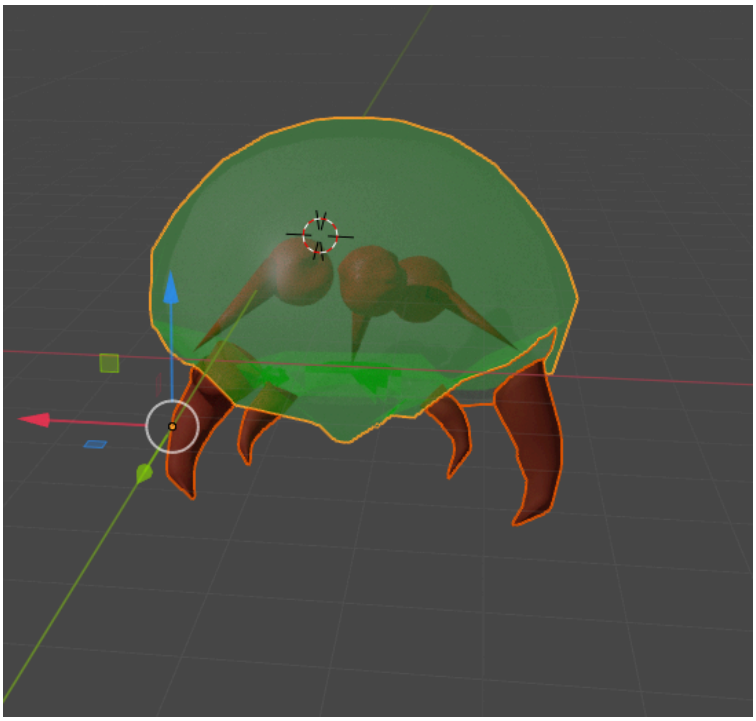
Name	ID Number
K Venkat Anoop	2017A7PS0271H
M Trinadh	2017A7PS0061H
C Mahesh Babu	2017A7PS0235H

The Scene

The scene depicts two astronauts guy0 and guy1 of Bits pilani Hyderabad Campus, who on their voyage through space discover aline life on the planer SR388. They are overjoyed and plant the BITS flag to mark their achievement.

Scene Artefacts

All the objects were modeled in blender 2.82 and use solid colors. The scene initially contained objects made in OpenGl like cubes and tetrahedrons but were later removed to accomodate more detailed blender models. Nevertheless ther source code still contains the vertex data for the objects

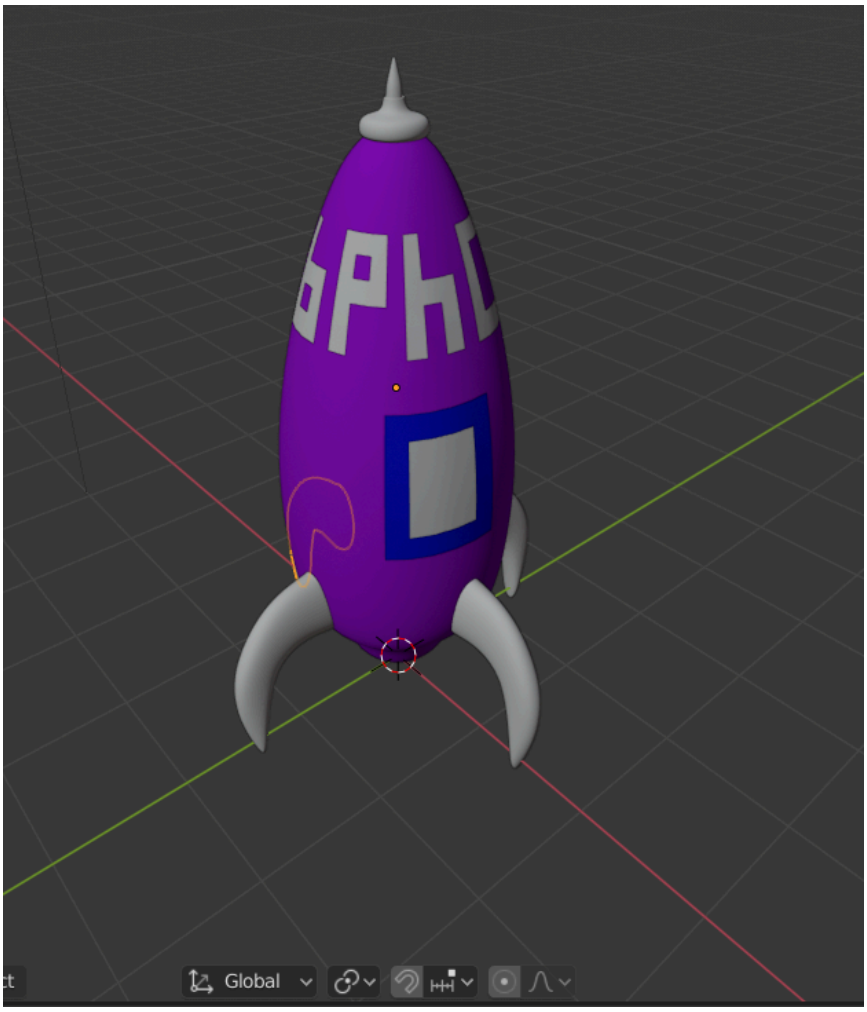


Alien

Click on the image for more details

Alien*close*

The alien we designed uses transparent materials for the green sheathe, so we had to implement alpha blending for proper results. As can be seen the aliens internal organs are outwardly visible.

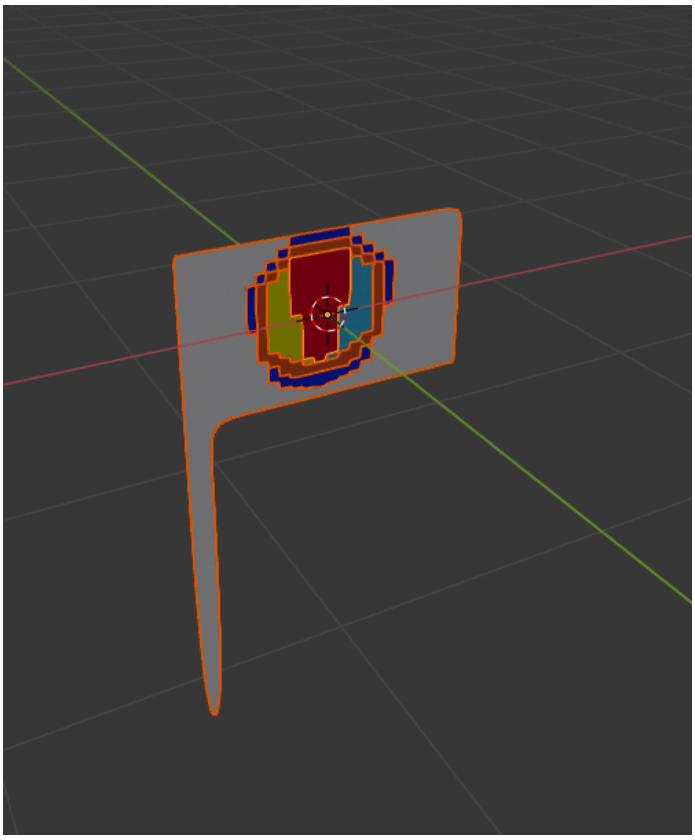


Rocket

Click on the image for more details

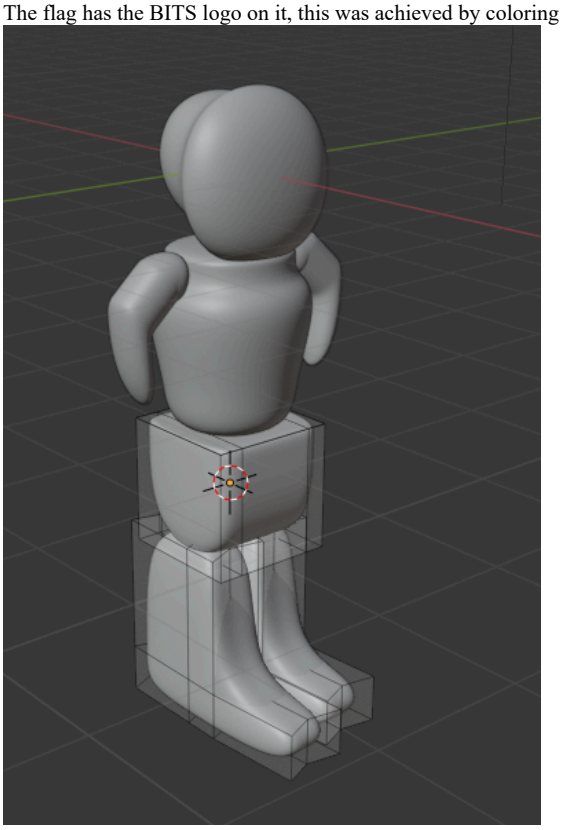
Rocket*close*

The rocket's window is transparent and in the final scene an astronaut inside the rocket can be seen from the window. The rocket also contains the letters 'BPHC' which are a separate mesh.


Flag

Click on the image for more details
Flag*close*

The flag has the BITS logo on it, this was achieved by coloring individual faces since textures weren't allowed .

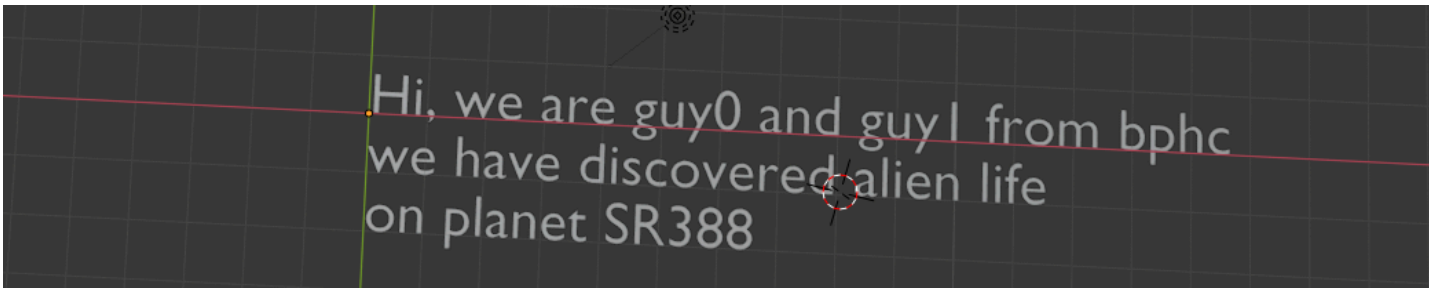


Astronaut

Click on the image for more details

Astronaut*close*

The astronaut in the picture seems to be missing colors, this is because the materials from eevee aren't compatible with the cycles renderer. The scene uses a .mat file that was handcrafted to give the astronaut colors.



Text

Click on the image for more details

Text*close*

Since the scene can be quite hard to understand at a cursory glance, we have added text to the scene to aid viewers, the mesh was created in blendertext .

Note : Utah's teapot is not a primitive in GLEW because GLEW is a lightweight extension and only essential function pointers to the OpenGL implementation(drivers : Nvidia 445.75) are defined

The Camera

The scene is allowed to have multiple cameras, and the provided camera class can be sued to achieve the same. The camera objects do not allow oblique viewing angles and are always initially at z = 3, x,y = 0. The camera is moved around with 'W, A, S, D', and panned around with the mouse axes. The scroll wheel can be used to zoom in and out.

```
glm::mat4 freecam::view()
{
    glm::mat4 view = glm::mat4(1.0f);
    view = glm::lookAt(CamPos, CamFront + CamPos, worldUP);
    return view;

}
```

Viewing

The core of the camera. This is the normalizing transform section of the 3D Pipeline

```
float currentFrameTime = glfwGetTime();
deltaTime = currentFrameTime - prevFrameTime;
prevFrameTime = currentFrameTime;
float speed = 5.f;
glm::vec3 velocity_cr = speed * CamFront;
glm::vec3 velocity_strafe_right = speed * (glm::cr
if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
    CamPos += velocity_cr * deltaTime;
}
if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
    CamPos -= velocity_cr * deltaTime;
}
if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
    CamPos += velocity_strafe_right * deltaTime;
}
if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
    CamPos -= velocity_strafe_right * deltaTime;
}
```

Controls

The above controls of the camera change the CamPos value, which in turn changes the viewing matrix. Other controls are similar and can be queried from the documentation

Lighting

The scene uses the phong lighting model. Each material has a diffuse, ambient, specular, and shininess component. The transparent materials require alpha blending, but since we couldn't implement order free alpha bending, we just order the objects in descending order of transparency. Its not possible to estimate the distance of an object from the camera without raycasting and the OpenGL zbuffer doesnt recognize objects(it only recognizes fragments).

```
#version 460 core
out vec4 color;
in vec3 normal;
in vec3 fragment_position;
struct Material {
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float shininess;
};
struct Light {
    vec4 position;

    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
};
uniform Material material;
uniform Light light;
uniform float alpha;
uniform float ambient;
uniform vec4 Camera_Position;
void main() {
    //ambient component first
    vec3 ambient = vec3(light.ambient) * vec3(material.ambient);
    //Then the diffused component
    vec3 Normal = normalize(normal);
    vec3 light_incidence = normalize(vec3(light.position) - vec3(fragment_position));
    //because when the dot is negative i.e the angle of incidence is greater than 90, the fac
    float diff = max(dot(Normal, light_incidence), 0.0);
    vec3 diffuse = vec3(light.diffuse) * (diff * vec3(material.diffuse));
    //Finally the specular highlights
    vec3 view_direction = normalize(vec3(Camera_Position) - fragment_position);
    vec3 reflect_direction = reflect(-light_incidence, Normal);
    float spec_highlight = pow(max(dot(view_direction, reflect_direction), 0.0), material.shi
    vec3 specular = vec3(light.specular) * (spec_highlight *vec3( material.specular));
    vec3 frag_final_factor = ambient + diffuse + specular;

    color = vec4(frag_final_factor, alpha);
};
```

No issues found

The Fragment Shader

As can be seen above the pixel color is calculated based on the material properties, normals, light position, and viewers position

Final Results

The video embedded below is a preview of the scene

0:00 / 0:21

Final Note

The libraries used were GLM, GLEW, GLFW and assimp. GLFW needs to be statically linked, whereas GLEW, and assimp need to be dynamically linked for the code to work(dll files need to be next to executables). The include folders in each dependency folder have to be added to the include paths abecuase the source code addressing relative to the visual studio solution directory. GLM is a header only library and the source code is found in src/vendor. Finally, assimp was build using the 64bit debug configuration in visual studio and won't work on 32bit machines.