CGI TP3 Report

68130 Dinis Neves
68613 Pedro Gomes

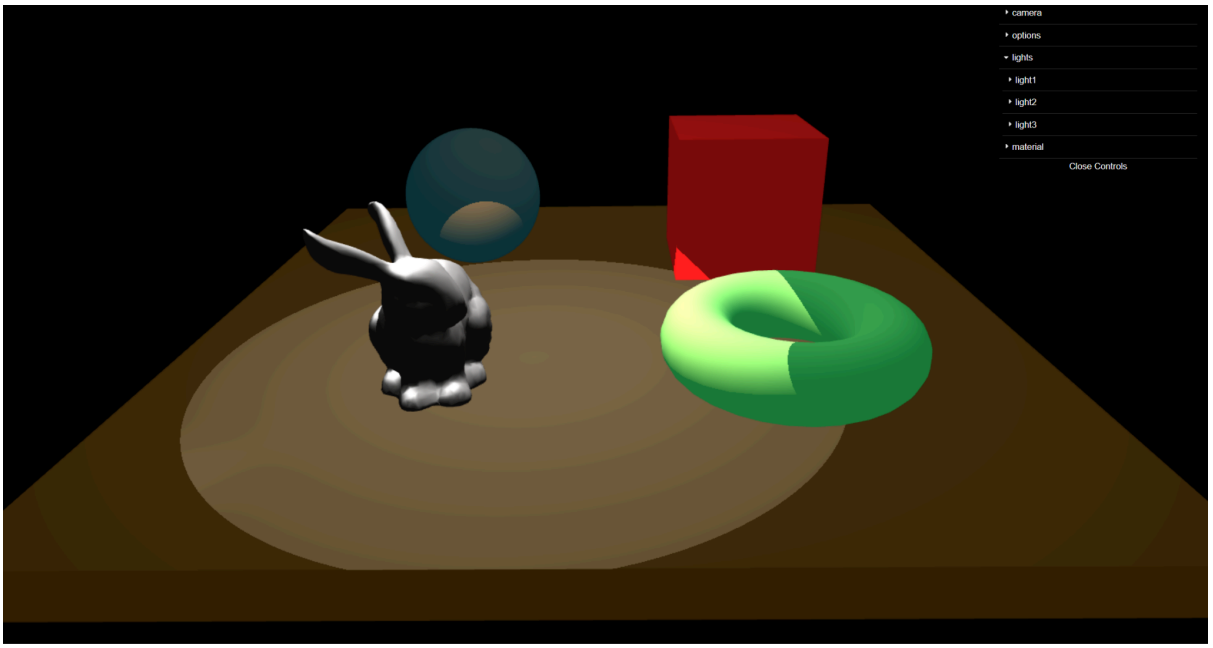

Figure 1.1: Screenshot of the scene

Entry variables for vertex and fragment shaders

(Using world and space interchangeably).

Phong Vertex Shader

- a_position: the position of object vertex in the object's space / world
- a_normal: the normal vector in the object's space

- u_projection: the projection matrix (eye to clip)
- u_model_view: the model matrix (object to eye)
- u_normals: normal matrix (object to eye)

- v_normal: the normal vector transformed to eye space
- v_position: vertex position in eye space

```
1   #version 300 es
2
3   in vec4 a_position;
4   in vec3 a_normal;
5
6   uniform mat4 u_projection;
7   uniform mat4 u_model_view;
8   uniform mat4 u_normals;
9
10  out vec3 v_normal;
11  out vec3 v_position;
12
```

Figure 2.1: Screenshot of vertex shader (shader.vert) uniforms, inputs and outputs

```
6   uniform bool u_use_normals;
7
8   // Camera
9   uniform vec3 u_camera_eye;
10
11  // lights can use struct
12
13  uniform int u_numLights;
14
15  struct LightInfo {
16      // Intensities
17      vec3 ambient;
18      vec3 diffuse;
19      vec3 specular;
20      // Light geomotry
21      vec4 position;
22      vec3 axis;
23      float aperture;
24      float cutoff;
25  };
26
27  // array of lights
28  uniform LightInfo u_L[MAX_LIGHTS];
29
```

```
26
27  // array of lights
28  uniform LightInfo u_L[MAX_LIGHTS];
29
30  // material struct
31  struct MaterialInfo {
32      vec3 Ka;
33      vec3 Kd;
34      vec3 Ks;
35      float shininess;
36  };
37
38  // material
39  uniform MaterialInfo u_material;
40
41  in vec3 v_normal;
42  in vec3 v_position;
43
44  out vec4 color;
```

Figure 2.2, 2.3: uniforms, inputs and outs of fragment shader (shader.frag)

Phong Fragment Shader

- u_use_normals: render colors without shading if true
- u_camera_eye: camera position in world space
- u_numLights: number of lights active

Light

- LightInfo: struct containing all of the light properties:
  ambient, diffuse, specular, position, axis, aperture, cutoff

- u_L: array of lights (using the LightInfo struct)

Object Material

- MaterialInfo: struct of material properties:
  Ka: ambient
  Kd: diffuse
  Ks: Specular

- u_material: material of the object

- v_normal: normal calculated in vertex shader
- v_position: vertex position calculated in vertex shader

- color: pixel color output (final pixel color)

Objects

The structure used to represent the objects in the project was a js object called *sceneConfigurationObject* that includes the camera, options, lights, material and objects.

As seen in the example figures below the

```
// Camera
sceneConfigurationObject.camera = {
    eye: vec3(0, 3, 10),
    at: vec3(0, 0, 0),
    up: vec3(0, 1, 0),
    fovy: 45,
    aspect: 1, // Updated further down
    near: 0.1,
    far: 20
};
```

Figure 3.1: screenshot of snippet of camera configuration

```
88      sceneConfigurationObject.lights = [
89          // Light 1 -> Spotlight
90          {
91              position: { x: 0.0, y: 5.0, z: 0.0, w: 0.0 },
92              intensities: {
93                  ambient: [0.1, 0.1, 0.1],
94                  diffuse: [0.8, 0.8, 0.8],
95                  specular: [1.0, 1.0, 1.0]
96              },
97              axis: { x: 0.0, y: -1.0, z: -1.0 },
98              aperture: 90.0,
99              cutoff: 1.0,
100             turnedOn: true
101         },
```

Figure 3.2: screenshot of snippet of light configuration

```
206     sceneConfigurationObject.objects = [{
207         name: "Table",
208         shape: CUBE,
209         transformations: [
210             {transformationType: "s", measures: [10, 0.5, 10]}
211         ],
212         material: {Ka:[0.4, 0.25, 0.00],
213                    Kd:[0.3, 0.3, 0.3],
214                    Ks:[0.1, 0.1, 0.1],
215                    shininess: 256,
216                 },
217     },
218     {
219         name: "cube",
220         shape: CUBE,
221         transformations: [
222             {transformationType: "t", measures: [2.0, 1.25, 2.0]},
223             {transformationType: "s", measures: [2, 2, 2]}],
224         material: { Ka:[0.9, 0.1, 0.1],
225                    Kd:[0.9, 0.1, 0.1],
226                    Ks:[0.9, 0.1, 0.1],
227                    shininess: 1,
228                 },
229     },
```

Figure 3.3: screenshot of snippet of objects being defined in the scene (the base, table and the cube)

Additional features

Our additional features include: WASD camera movement, changing between Phong and Gouraud shading, and 3 lights and Light toggle.

WASD movement.

It is possible to move the camera using “w” where the camera is looking at, “s” the opposite direction of “w”, “a” to the left, and “d” to the right. The “r”
key can also be pressed to reset to the initial state of the camera.
Swapping shading.

There is the option to swap from 2 different shaders.

There are 2 different shaders and an option to the option menu was added with a list of two options to choose from. Gouraud and Phong

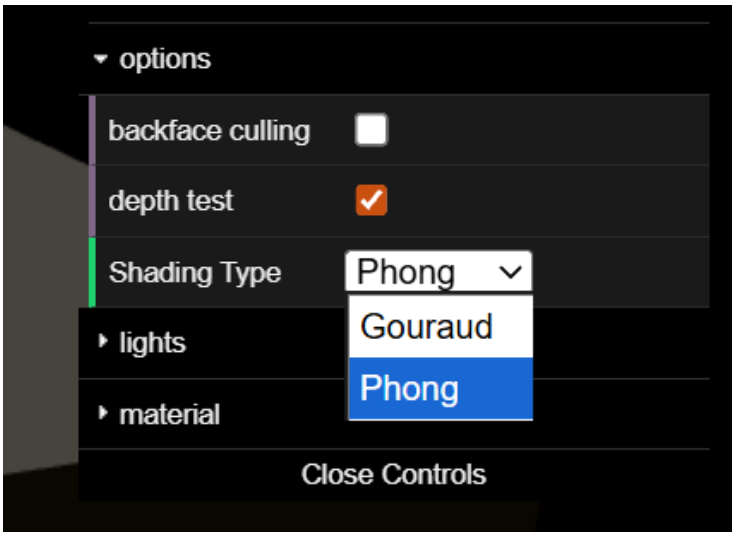

Figure 4.1: screenshot of shading swapping option menu

The number of lights

The number of lights is increased to 3 instead of 1. Allowing the interactions of many different lights at different intensities to be seen on the surface of the objects.

Toggle on and off lights.

On the Gui all lights have the options to toggle off and on. To see a clearer impact of just one light and to facilitate turning of lights instead of setting their intensities to 0. Below can see where the toggle is.

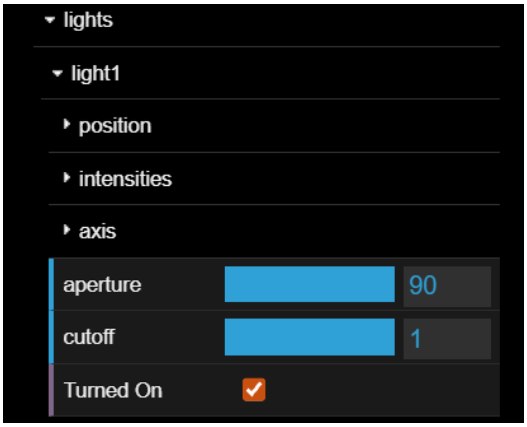

Figure 4.2: screenshot of GUI showing Turned On toggle on the bottom