Trabalho 2 CG 2022-23

Grupo CG15

- 56282- Tomás Alexandre Regolta Barreto

- 56292- João Ricardo Silva Matos

- 56302- Diogo José Madureira Pereira

Relatório

i) Implementação dos objetos em 3D

Figura 1 - Visão global da cena implementada, com destaque á Terra e a sua lua, no canto inferior direito

Foram representados todos os principais planetas do sistema solar, bem como o Sol, Plutão e a lua da Terra. Ao todo, existem 14 objetos na cena, sendo 12 dos quais esferas e os restantes 2, toros, que representam os anéis de Saturno e Úrano. Foi ainda incluída uma esfera que engloba toda a cena, que atua como *skybox*, acrescentando estrelas no fundo da cena.

ii) Texturas

As texturas são carregadas usando a função *createTexture*, da seguinte forma:

```
const texSun = twgl.createTexture(gl, { src: '/texturas/sun.jpg' });
```

Após serem carregadas, todas as texturas são colocadas num *array*:

```
const textures = [texSkybox,texSun,texMercury, texVenus,texEarth,…
```

Que é acedido ao desenhar os objetos na cena, colocando a textura correta no campo *uniform u_diffuse* de cada objeto:

```
uniforms.u_diffuse = textures[i];
```

Por fim, é criado um objeto *texture2D* com o *u_diffuse*, que por sua vez é aplicado ao objeto na iluminação do mesmo:

```
vec4 diffuseColor = texture2D(u_diffuse, v_texCoord);
```

iii) Animação

As animações de rotação e translação dos planetas foram implementadas através das funções *translation, rotation e multiply*, do módulo *m4*, bem como o valor de *time*, passado á função *render*. Por exemplo, a simples rotação do sol em torno de si mesmo foi conseguida da seguinte forma:

```
const sunTranform = m4.rotationZ(time*0.0002);
```

No caso da Terra, fizemos da seguinte forma:

```
var earthTransform=m4.multiply(m4.translation([3,0,0]),m4.rotationZ(time*0.0003));
    earthTransform = m4.multiply(m4.rotationZ(time*0.0003),earthTransform);
```

Onde primeiramente é aplicada uma rotação em torno de Z, tal como no Sol, de seguida uma translação que define o raio da sua órbita e por fim uma nova rotação, que dita o seu tempo de órbita. A lua da Terra tem a animação mais complexa de toda a cena:

```
var moonTransform = m4.multiply(m4.rotationY(time*0.001),m4.translation([-0.4,0,0]));
    moonTransform = m4.multiply(earthTransform, moonTransform);
```

Começamos por aplicar uma translação e rotação, que representam a sua órbita em torno da Terra e, por fim, multiplicamos pela transformação da Terra, para que esta esteja sempre no centro da sua órbita. Tal como os objetos, após serem definidas, estas transformações são colocadas num *array* e passadas aos *shaders* através dos campos *uniforms*:

```
uniforms.u_world = tranformations[i];
uniforms.u_worldInverseTranspose = m4.transpose(m4.inverse(tranformations[i]));
uniforms.u_worldViewProjection = m4.multiply(viewProjection, tranformations[i]);
```

E dentro do *vertex shader* são aplicadas aos objetos:

```
gl_Position = u_worldViewProjection * position;
```

A animação pode ser vista no seguinte vídeo:

iv) Iluminação

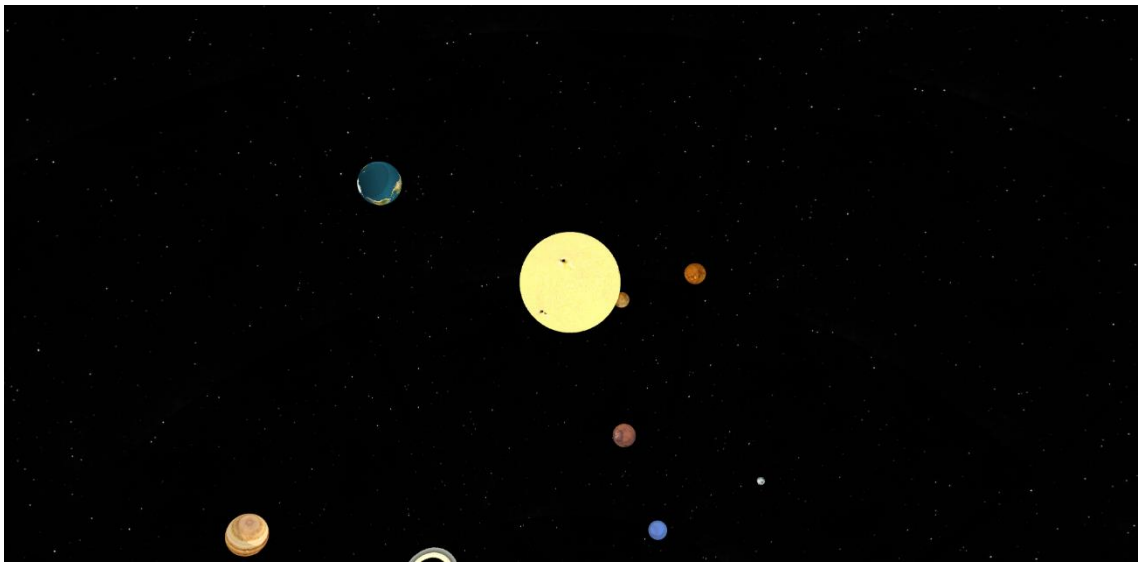

Figura 2 - Shading de Gouraud

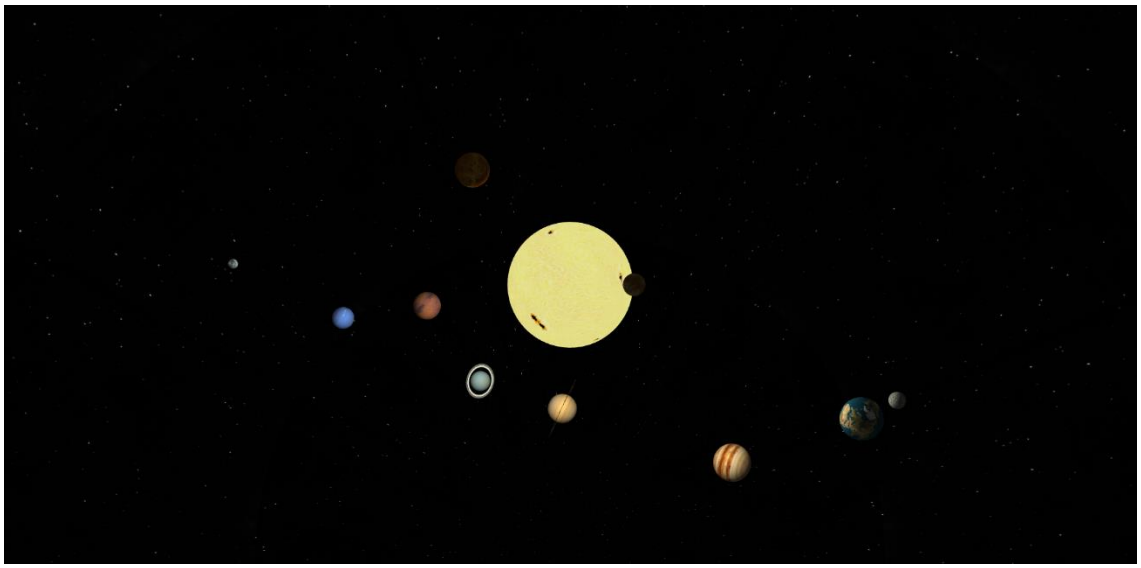

Figura 3 - Shading de Phong, sem componente especular

É possível notar algumas diferenças entre os dois algoritmos de *shading*, nomeadamente na sombra dos planetas e no brilho do mesmo, no entanto, as diferenças seriam mais significativas em objetos com um número menor de faces, pois no algoritmo de *Gouraud* as intensidades são calculadas a partir das normais aos vértices, apresentando piores resultados que no algoritmo de *Phong*.

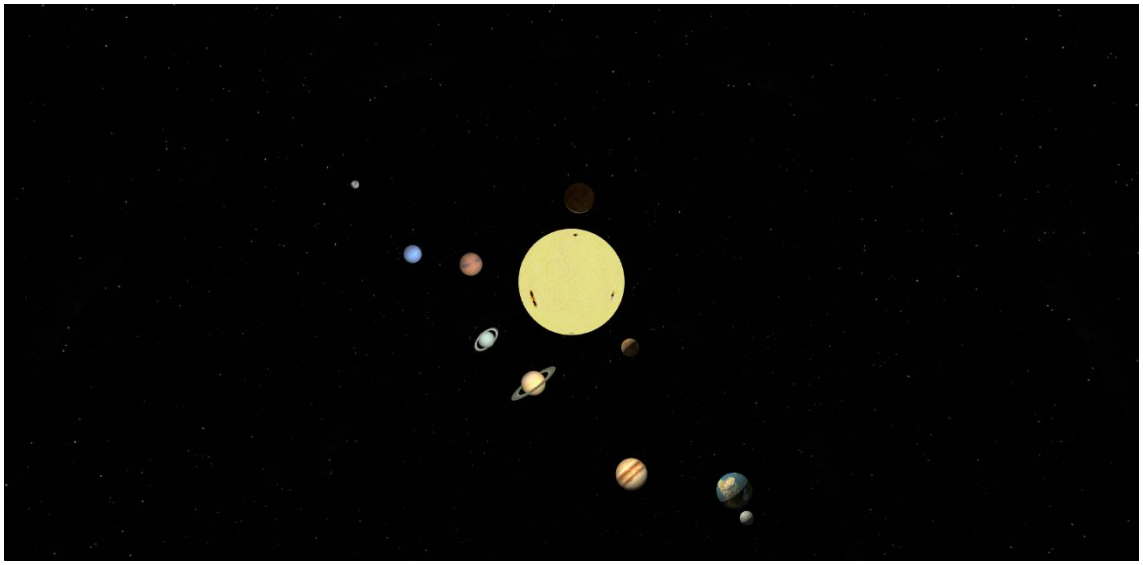

Figura 4 - Shading de Phong, com componente especular

Na figura acima vemos novamente o algoritmo de *Phong*, desta vez completo com a componente especular. Apesar de subtil, é possível ver algum brilho nos planetas, na direção do sol, resultante desta componente adicional não presente na imagem da figura 3.

Figura 5 - Componente especular

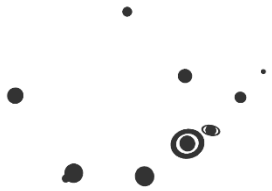

Figura 6 - Componente Ambiente

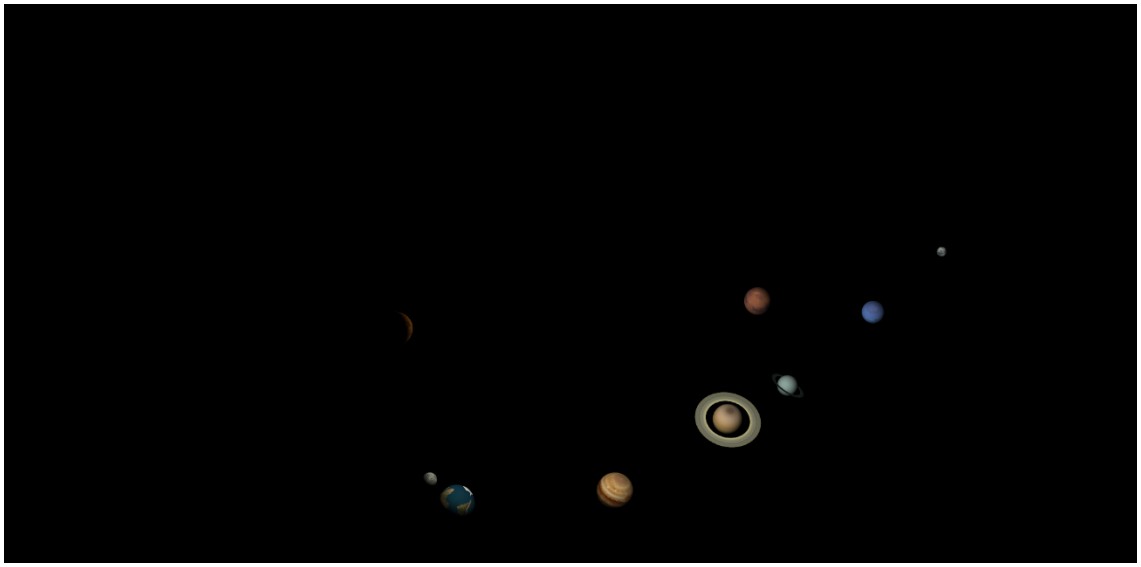

Figura 7 - Componente difusa

Nas figuras 5 a 7 podemos ver o algoritmo de *Phong*, com as componentes especular, ambiente e difusa habilitadas individualmente. Para visualizar apenas a componente ambiente foi aplicada uma textura branca a cada objeto, de forma que esta componente de luz se destacasse.

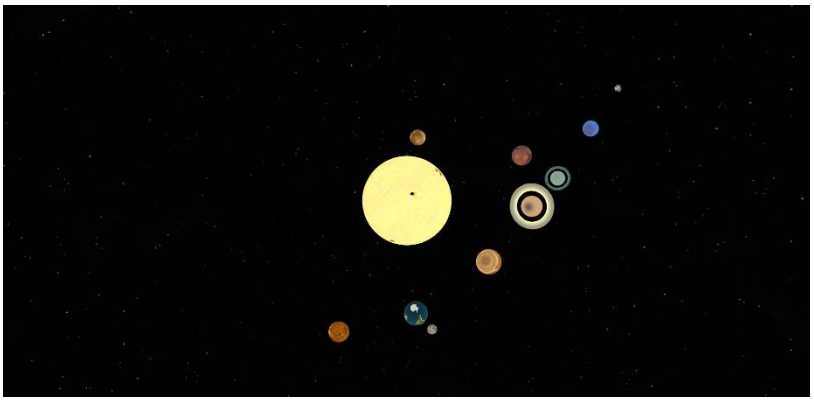

Figura 8 - Fator de especularidade 0

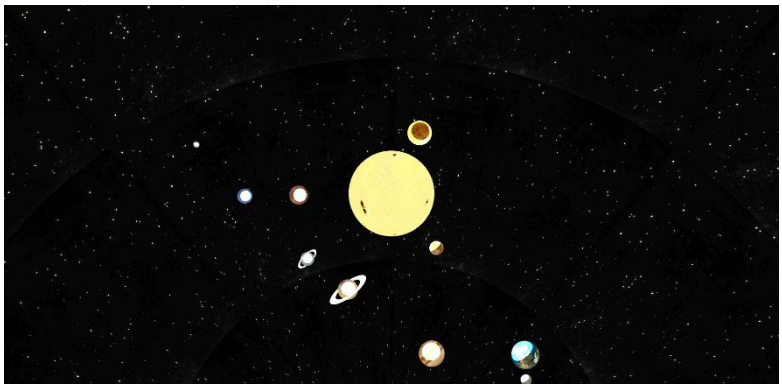

Figura 9 - Fator de especularidade 5

Por fim, na figura 8 vemos o algoritmo de *Gouraud*, com coeficiente de especularidade 0 e na figura 9, o mesmo algoritmo, desta vez com coeficiente de especularidade 5. As diferenças são imediatamente observáveis, na luz que incide diretamente nos planetas.

v) Extras

Implementámos uma funcionalidade de movimento da câmara com o botão direito do rato e rotação da mesma, com o botão esquerdo, bem como zoom, com a roda.

Os planetas Saturno e Úrano têm também os seus anéis representados, que orbitam sobre si.

Adicionámos ainda um fundo com estrelas a toda a cena, que pode ser facilmente desabilitado ao ativar a opção *gl_CULL_FACE* na função *render*.