Universidade de Coimbra

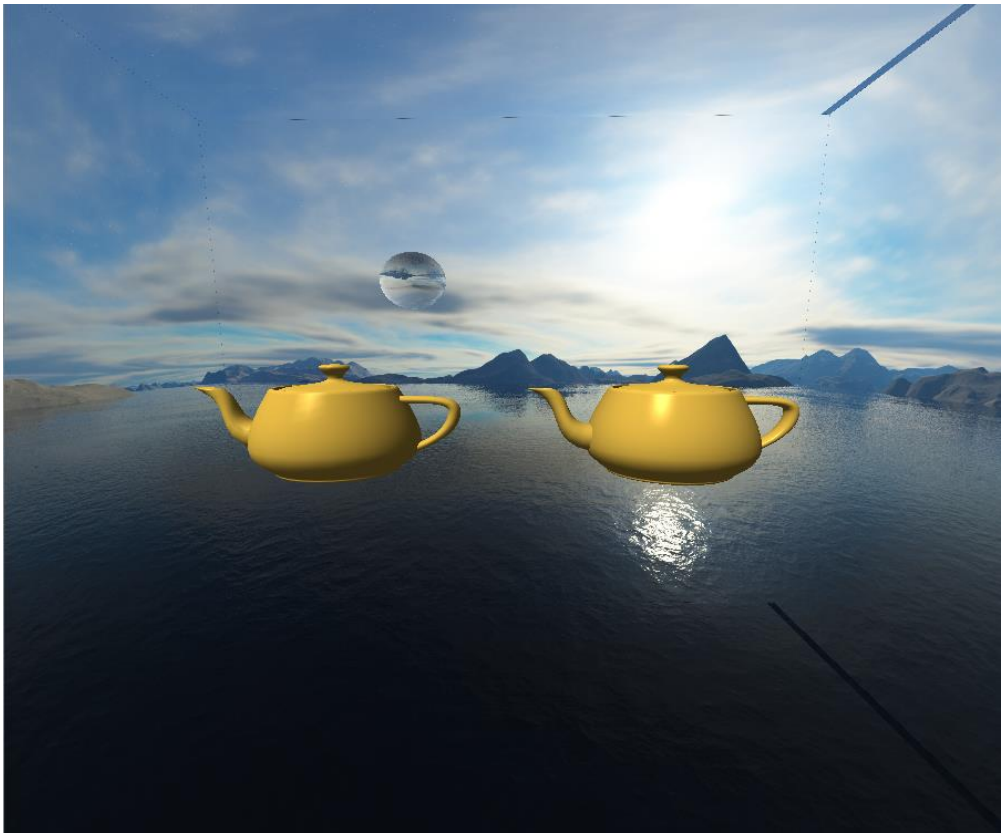
Faculdade de Ciências e Tecnologia

Licenciatura em Engenharia Informática

Computação Gráfica – 2020/2021

3º Ano – 1º Semestre

Relatório Meta 3



Coimbra, 23 de janeiro de 2021

2018298731 João Pedro Pacheco Silva joaopedro@student.dei.uc.pt

Iluminação:

O tema pelo qual optei foi a alternativa 1, onde era suposto implementar o modelo de iluminação de phong usando a interpolação de Gouraud num objeto e a interpolação de Phong em outro.

Como era suposto implementar dois tipos luz, havia a necessidade de calcular o vetor de iluminação de duas maneiras diferentes.

No caso da luz ser direcional, o vetor de iluminação era obtido normalizando o vetor "Direcao" passado para os shaders como uma variável Uniform. Caso fosse pontual "Direcao" passava a ser usado como a localização da luz e o vetor de iluminação era obtido pela diferença entre a posição da luz e do vértice.

```
if(Luz==1){   //pontual
        luz = normalize(Direcao-Position);
}
else{   // direcional
        luz = normalize(Direcao);
}
```

A posição dos vértices nas coordenadas mundo era guardada na variável varying com o nome "Position".

Em ambos os objetos, a determinação da cor envolvia o cálculo das intensidades especular (intensityS) e difusa (intensityS). Esse cálculo foi feito da seguinte forma:

```
vec3 r = 2*dot( nNormal, luz)*nNormal-luz;
vec3 obs = normalize(cameraPosition - Position);
float intensityD = max( dot( nNormal, luz), 0.0);
float intensityS = pow(max( dot( obs, r), 0.0), coef);
```

Para o vetor de visualização (obs) foi necessaria a posição do observador (cameraPosition) que foi passada como variavel Uniform e para o vetor de reflexão especular (r) foi usou-se a normal normalizada (nNormal).

Para o objeto com a interpolação de Gouraud a cor é calculada no vertex shader e guardada numa variável varying com o nome "cor". A cor do fragmento era depois definida usando essa mesma varying, havendo, portanto, interpolação da cor em cada fragmento.

Vertex Shader:

```
varying vec4  cor;
…
void main(void) {
…
(cálculo das intensidades)
cor = corA + intensityD*corD + intensityS*corS;
}
```

Fragment Shader:

```
varying vec4  cor;
…
void main(void) {
…
gl_FragColor = cor;
}
```

Para o objeto com a interpolação de Gouraud a cor é calculada no fragment shader, sendo os valores da posição e vetor normal passados para o fragment shader através de variáveis varying. Fazendo isso passa haver interpolação das posições e normais dos vértices.

```
Verteex Shader:
varying vec3  vNormal;
varying vec3  Position;
…
void main(void) {
…
vNormal = gl_Normal ;
Position =  gl_Vertex.xyz;
}
```

```
Fragment Shader:
varying vec3  vNormal;
varying vec3  Position;
…
void main(void) {
…
(cálculo das intensidades)
gl_FragColor= corA + intensityD*corD + intensityS*corS;
}
```

Para simular os efeitos de reflexão e refração foi usado o conceito de superfície envolvente para fazer o mapeamento de uma textura sobre os objetos. Os raios foram calculados através da reflexão/refração do vetor de visualização sobre a normal interpolada e a textura usada no mapeamento foi a mesma que foi usada para o senário.

```
vec3 reflected = normalize(cameraPosition - Position);
reflected =  2*dot( nNormal, reflected)*nNormal-reflected;
gl_FragColor  = texture( text7,  reflected );
```

```
float ratio = 1.00 / 1.33;
vec3 refracted = normalize(cameraPosition - Position);
refracted = -refracted;
float angi = acos(dot(-nNormal, refracted));
float angr = asin(ratio*sin(angi));
refracted = cos(angr)*(-nNormal) + sin(angr)*(refracted);
refracted = normalize(refracted);
gl_FragColor  = 0.3*gl_FragColor +  0.7*texture( text7,  refracted );
```

Animação:

No programa principal existe uma função timer que vai atualizando constantemente uma variável contendo o instante temporal. O valor dessa variável é passado ao vertex shader através de uma variável Uniform (Time). A posição de cada vértice no eixo x vai sendo incrementado pela variável enquanto que a posição no eixo y é descrita por várias parábolas em função de "Time".

De cada vez que uma parábola interceta o eixo das abcissas é iniciado outra mas com uma altura menor.

```
float time = Time/20;
gl_Position.x = gl_Position.x - time+1;
float y = 6 + 0.5*(-9.81)*time*time;
float i = 6;
float x = 0;
while(y<=0 && i>0.0001){
        i = i/1.8;
        x= sqrt((i)/(0.5*9.81));
        x = 2*x;
        time = time -x;
        y = i + 0.5*(-9.81)*time*time;
}
if (y<0) { y=0; }
gl_Position.y = gl_Position.y + y;
```