

UNIVERSIDADE DO MINHO
DEPARTAMENTO DE INFORMÁTICA

Computação Gráfica - Fase 4
Grupo N°2

João Correia (A84414)

Marco Pereira (A89556)

Pedro António (A58062)

Rúben Cerqueira (A89593)

28 de junho de 2021



João



Marco



Pedro



Rúben

Capítulo 1

Introdução

O presente relatório visa apresentar a quarta e última fase do trabalho proposto no âmbito da Unidade Curricular de Computação Gráfica.

Em relação à fase passada, foram melhoradas ambas as aplicações. O engine foi melhorado de modo a poder interpretar luz, ou seja, é possível ler do XML as várias fontes de luz, bem como aplicar texturas nas respetivas figuras com o parsing do mesmo ficheiro de configuração e definir a componente de luz aplicada a uma figura. Além disso, também lê os pontos referentes às normais e às coordenadas de textura de cada ficheiro .3d a utilizar na aplicação. Quanto ao Generator: permite gerar adicionalmente aos pontos e índices previamente implementados, normais e coordenadas de textura para cada ponto gerado.

Capítulo 2

Luz

Era expectável que nesta fase fosse implementado o Lighting de todas as figuras que o generator era capaz de gerar. Para que isso fosse possível são necessários 2 requisitos: Uma ou mais fontes de luz e um VBO que contém as normais da figura a desenhar.

2.1 Fontes de Luz

Foram implementados 3 tipos de luz: posicional, direcional e foco. Uma luz posicional emite luz em todas as direções a partir desse ponto. A luz direcional emite luz para todos os pontos com uma determinada direção. Por fim, a luz de foco tem um comportamento análogo a uma lanterna, ou seja, tem um ponto, uma direção, e um ângulo de emissão de luz.

O primeiro e segundo tipo de luzes foi conseguido com a rotina `glLightfv`, dando como argumento um array e a flag `GL_POSITION`. Este array teria 4 posições: as 3 primeiras mapeiam as coordenadas x, y e z, respectivamente, e a última afirma se se trata de um ponto ou vetor (1 para ponto e 0 para vetor), dando como resultados luz posicional ou direcional, respectivamente. Quanto à luz do tipo foco, é também utilizado o `glLightfv` para indicar a posição do ponto e a direção do foco, através das flags `GL_POSITION` e `GL_SPOT_DIRECTION`. Para assinalar o ângulo do feixe de luz, foi utilizada a função `glLightf` com a flag `GL_SPOT_CUTOFF`.

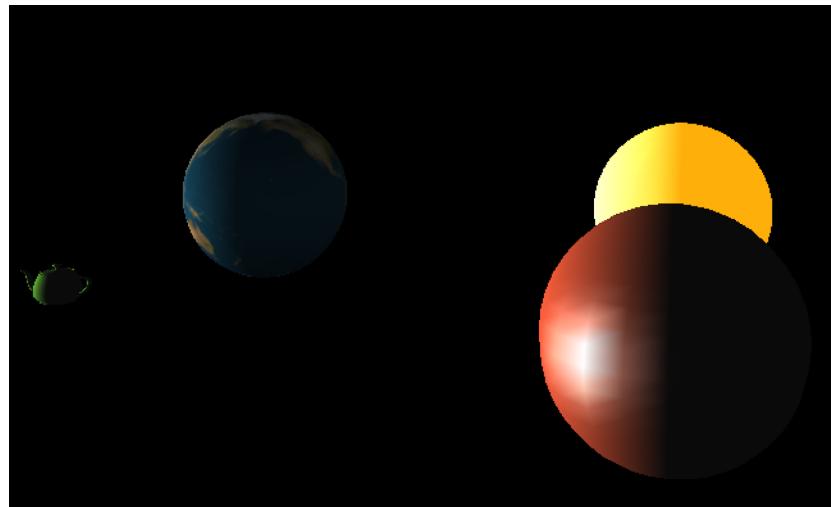


Figura 2.1: Luz direcional com direção em Z positivo

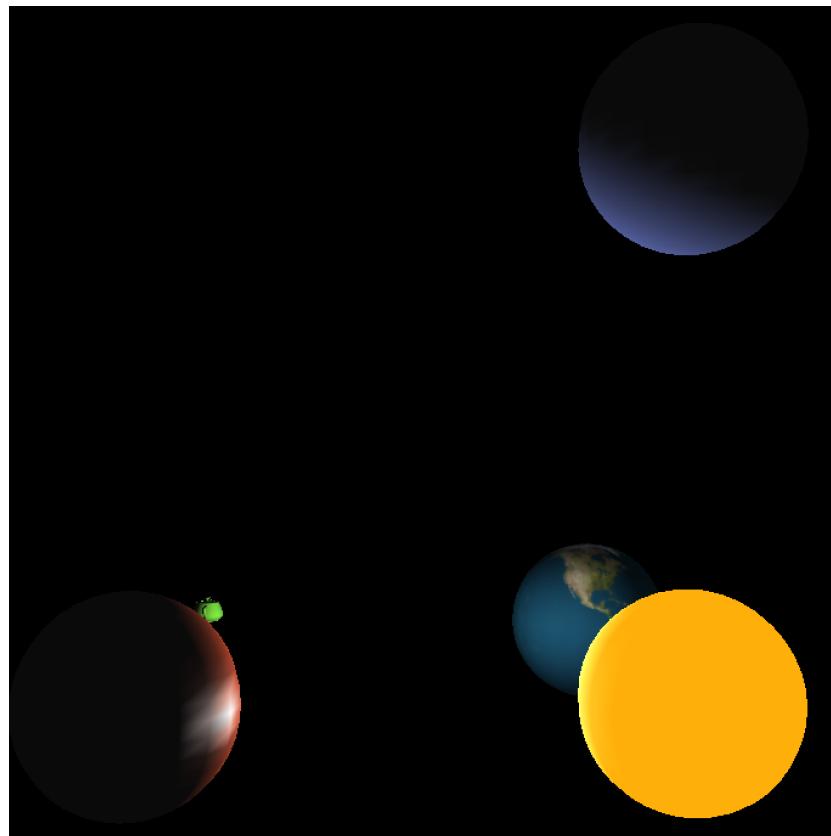


Figura 2.2: Luz posicional posicionada no meio das figuras desenhadas

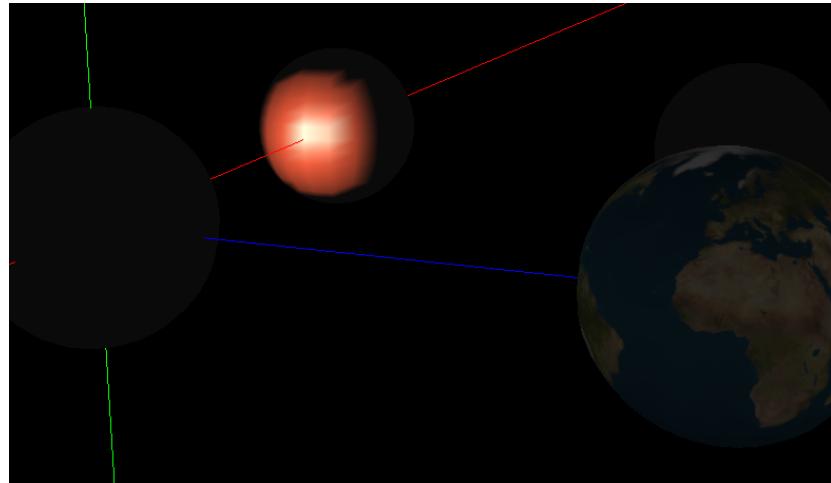


Figura 2.3: Luz foco com direção do X positivo e 18º de ângulo de feixe

2.2 Componentes de Cor

A forma como uma figura reage à luz também é importante, logo, foram implementadas 4 componentes de cor: difusa, especular, ambiente e emissiva. Para se obterem estas componentes de cor recorre-se à função `glMaterialfv`, dando como argumento as flags `GL_DIFFUSE`, `GL_SPECULAR`, `GL_AMBIENT` ou `GL_EMISSION`, dependendo do tipo de componente que se trata, respetivamente, e o array de 4 posições correspondente aos valores `RGBA`. Ainda na luz especular, é possível definir a shininess da luz correspondente à intensidade da luz especular, utilizando a função `glMaterialf` com argumentos `GL_SHININESS` e o valor pretendido.

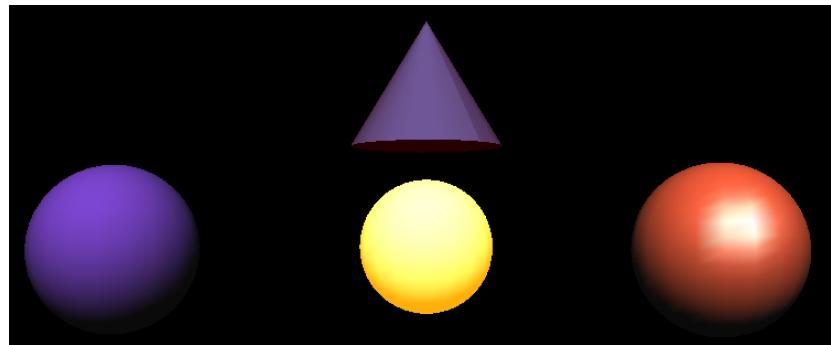


Figura 2.4: Várias componentes de cor

Na Figura 2.4 observam-se 4 figuras com diferentes cores e tipos de cores associados. A figura da esquerda apenas está colorida com componente difusa com tons de roxo. A esfera central incorpora luz emissiva amarela. A esfera à direita contém luz difusa avermelhada com luz especular como se pode observar com uma 'mancha' branca de luz, com uma shininess de 25. O cone é colorido com cor difusa azulada e cor ambiente vermelha, dando um aspecto violeta com a junção das duas componentes.

2.3 Normais

Para que a iluminação desse resultado, as normais teriam que ser bem definidas e normalizadas para cada figura a desenhar.

2.3.1 Plano

Para o plano apenas são necessárias 4 normais com valores $(0,1,0)$ para as normais, uma vez que se tratam de 4 vértices e todo o plano está virado para cima.

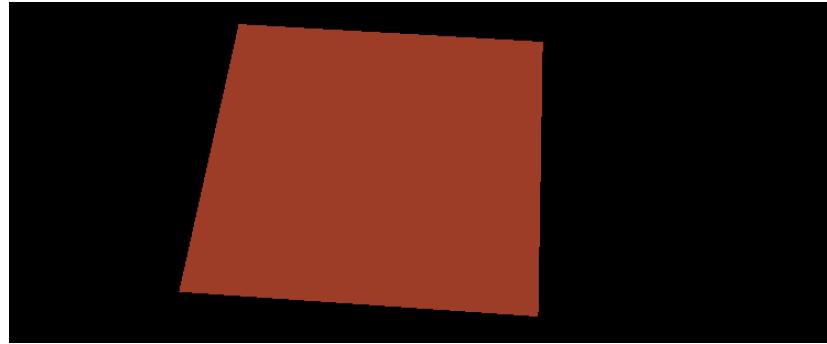


Figura 2.5: Plano iluminado

2.3.2 Cubo

No caso do cubo o processo não facilitou o modo de desenho por índices implementado de modo a poupar pontos, pois cada vértice teve que ser repetido 3 vezes uma vez que contém 3 normais diferentes pertencentes a cada face que integra. Os pontos de cada face mapeiam normais com direção perpendicular ao plano, por exemplo, os vértices da face virada para cima têm todos normais com valores $(0,1,0)$.

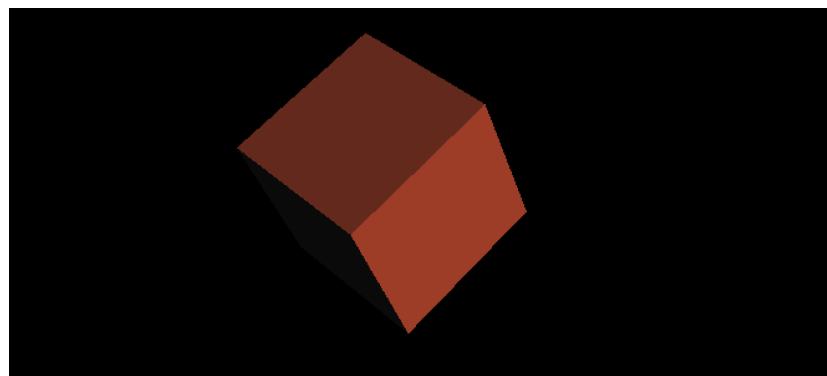


Figura 2.6: Cubo iluminado

2.3.3 Esfera

Para a esfera não foi necessário repetir nenhum ponto, uma vez que em nenhum dos vértices correspondem normais diferentes. Para obter a normal de um vértice basta normalizar as coordenadas do próprio vértice.

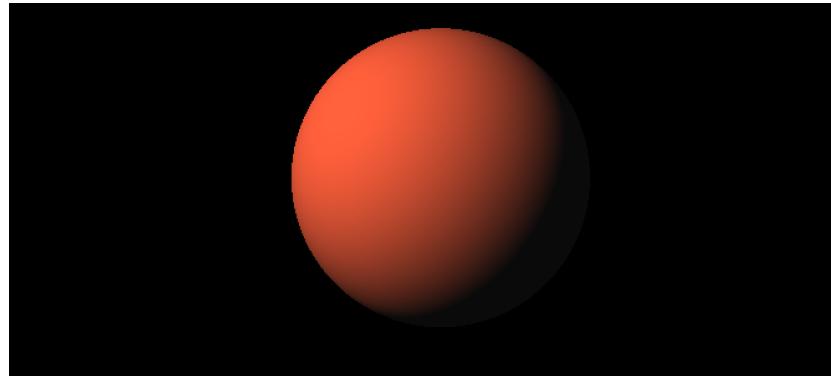


Figura 2.7: Esfera iluminada

2.3.4 Cone

Calcular as normais de um cone já necessitou um cálculo adicional nos vértices da superfície lateral. Dado um vértice são obtidos adicionalmente 3 vértices: o vértice da stack superior, o vértices da slice anterior e o vértice da slice seguinte. É necessário o vértice acima para calcular o vetor vertical do cone, já o anterior e seguinte têm como função calcular o vetor horizontal. Com ambos os vetores obtidos, faz-se o produto externo de forma a calcular a normal do plano definido pelos dois vetores que corresponderá ao plano tangente ao vértice a analisar, obtendo assim a normal pretendida após normalização

São repetidos os pontos da base, uma vez que também pertencem à superfície lateral do cone, tendo duas normais diferentes. As normais da base terão todos os valores $(0, -1, 0)$ uma vez que se encontram voltadas para baixo.

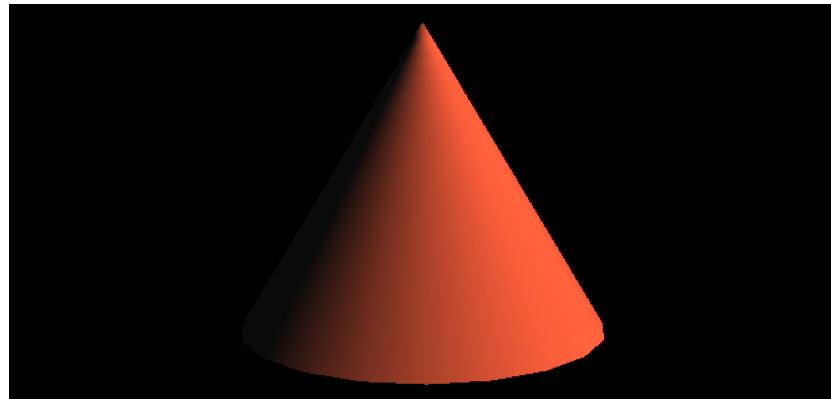


Figura 2.8: Cone iluminado

2.3.5 Torus

De seguida, para o cálculo das normais do torus apenas foi necessário o ponto central de cada anel. Logo, para cada vértice desse anel, subtrai-se ao ponto central, dando origem à normal depois da normalização do resultado.

Teve-se especial atenção ao caso em que o torus tem 2 lados, uma vez que nesse caso se comportaria como um plano, logo as suas normais apontariam para cima e para baixo

dependendo da face. Não se podia optar pelo mesmo método anterior pois caso o fizéssemos, as normais iriam apontar para os lados, não sendo iluminado devidamente.

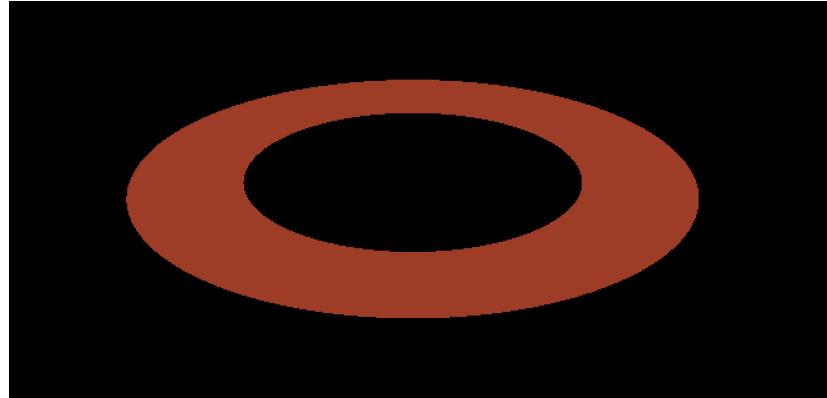


Figura 2.9: Torus com 2 lados iluminado

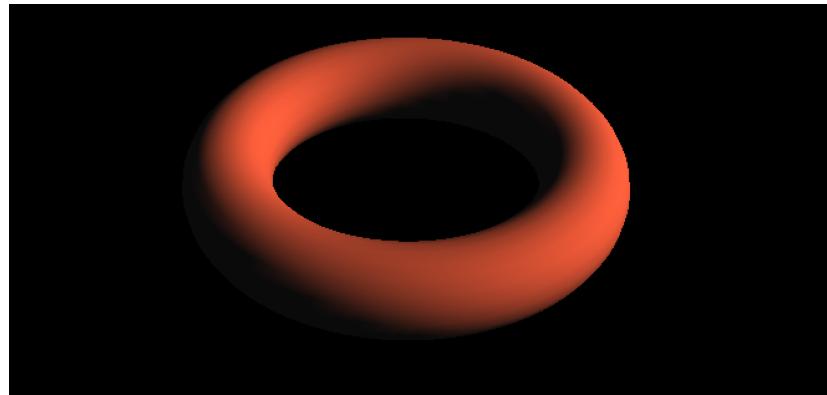


Figura 2.10: Torus iluminado

2.3.6 Patches de Bezier

Por último, nos patches de bezier foi onde se desenvolveu mais cálculo matemático na determinação de normais. Ao mesmo tempo que são calculados os vértices, são calculadas as derivadas parciais sobre as variáveis u e v , que originam os vetores horizontal e vertical, respectivamente, tangentes ao ponto. Depois do cálculo é feito o produto externo destes dois vetores e normalizado, finalizando assim o cálculo da normal para o vértice em questão.



Figura 2.11: Teapot iluminado com base em patches de Bezier

Capítulo 3

Texturas

Nesta fase também foi expectável a implementação de texturas. Tal como no lighting, o generator teve que ser alterado para conseguir conjugar toda a informação necessária para a geração de figuras.

De seguida, são mostrados os algoritmos de mapeamento de texturas para as figuras a desenhar.

3.1 Plano

Para o plano a aplicação da textura é simples, os 4 vértices mapeiam os extremos da textura em questão.



Figura 3.1: Textura aplicada a um plano

3.2 Cubo

Decidiu-se que para aplicar uma determinada textura para o cubo seria aplicá-la por inteiro para cada face do cubo, sendo assim repetida para cada face.

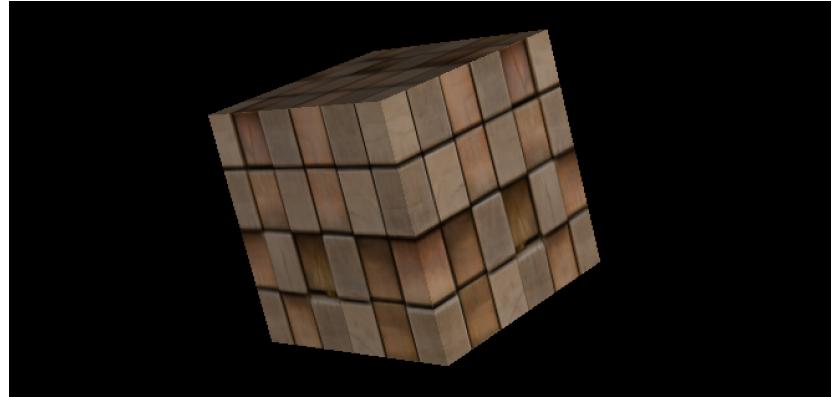


Figura 3.2: Textura aplicada a um cubo

3.3 Esfera

Para a esfera o mapeamento é especial, uma vez que não é possível tirar partido de toda a textura devido à natureza do polígono. A textura é mapeada por inteiro, dividindo as coordenadas de textura pelas várias stacks e slices no intervalo $[0,1]$. No entanto, nos polos isso não é possível uma vez que se tratam de triângulos e não quadrados. Logo, é calculado o ponto mediano entre os pontos entre slices para a definição do terceiro vértice. Este método não usa completamente a textura mas se definirmos um número considerável de stacks e slices, essa perda é pouco significativa e pouco notória.



Figura 3.3: Textura aplicada a uma esfera

3.4 Cone

Quanto ao cone é necessário que a textura contenha uma área retangular e uma área circular abaixo no extremo direito. A área circular terá um raio de 18.75% da textura. A área circular servirá para a base do cone, mapeada da mesma maneira como são calculados os pontos, ou seja, com as funções trigonométricas, apenas diferenciando no raio. A superfície

lateral é mapeada de modo análogo à esfera, tendo o mesmo problema no topo. Este problema é resolvido com a mesma aproximação feita na esfera.



Figura 3.4: Textura aplicada a um cone

3.5 Torus

Ao contrário do que foi feito nas iterações anteriores, a textura do torus é aplicada repetidamente, ou seja, para cada lado e anel é aplicada a textura total, sendo repetida nos próximos anéis e lados.



Figura 3.5: Textura aplicada a um torus

3.6 Patches de Bezier

Por fim, para cada patch de bezier é mapeada a textura completa, sendo repetida para cada patch. Esta foi a abordagem escolhida uma vez que os patches de Bezier podem representar qualquer figura geométrica, e assim é possível generalizar essa abstração.



Figura 3.6: Textura aplicada a um teapot com base em patches de Bezier

Capítulo 4

Modificações ao ficheiro XML

De forma a poder usufruir das modificações e adições feitas ao funcionamento do Engine, foi modificada a estrutura do ficheiro XML.

Nesta fase existe uma nova secção chamada **lights** que conterá a fonte de luz da simulação. A luz poderá ser dos 3 tipos mencionados nos capítulos anteriores, através do campo `type`, dando-lhe os próximos parâmetros adequados para cada tipo.

Além dessa secção, as tags `models` em vez de aceitarem cores como parâmetros, aceitam componentes de cor com os respetivos valores RGB. Shininess também pode ser definida.

Em conjunto com as componentes de cor também é possível dar uma textura como argumento, definindo o nome da textura que se pretende utilizar.

O parsing destas modificações é análogo ao das restantes, usando as funções da biblioteca `tinyxml2`.

Capítulo 5

Cenas relevantes

De forma a demonstrar as novas adições foi modificada a estrutura do xml do sistema solar e foi também adicionada uma cena que apresenta uma mesa de bilhar. Essas imagens são abaixo apresentadas:



Figura 5.1: Sistema Solar

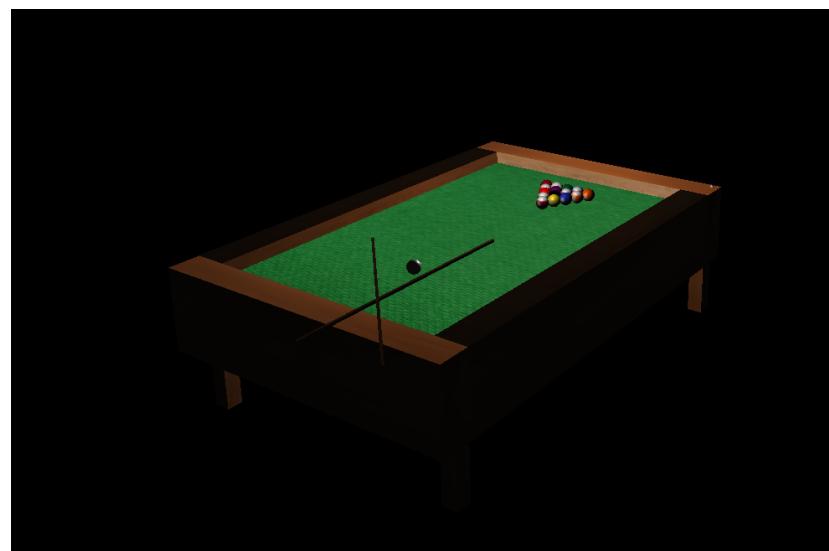


Figura 5.2: Mesa de Bilhar

Capítulo 6

Conclusão

Sendo esta a quarta e última fase do trabalho prático, foi possível desenvolver duas aplicações estáveis que cumprem com os requisitos propostos.

Algumas das funcionalidades extras foram implementadas, tal como a câmara em terceira pessoa, mas não foi possível completar todos os requisitos opcionais por escassez de tempo.

Como trabalho futuro é planeado a adição de frustum culling no trabalho, bem como a adição de várias fontes de luz, uma vez que estamos neste momento limitados a apenas uma.