Relatório Aula prática 2

José santos 98279, Ricardo Antunes 98275
Information Visualization, 2023 (Msc MRSI, Msc MEI , University of Aveiro)

Abstract

Neste relatório apresentaremos uma explicação do trabalho realizado na *Lesson* 2 da cadeira Visualização de Informação. Com este trabalho, foi possível obter conhecimentos sobre as projeções,iluminação e transformações utilizando a biblioteca Three.js.

Camera models and Orbit control

Nesta implementação seguimos o primeiro exercício que é baseado num tutorial disponível no site do *three.js*.

Com este tutorial, implementamos uma configuração de uma câmera ortográfica, com proporções baseadas no tamanho da janela do navegador, de modo a manter as proporções reais do cubo, sendo isto visível comparando a vista usando uma câmera com perspectiva .

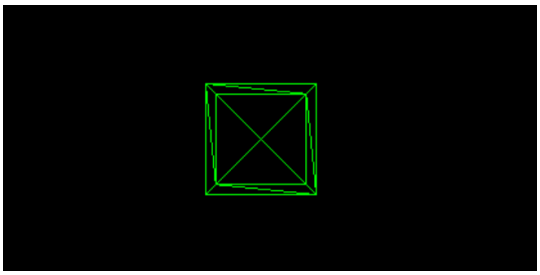

Figure 1: default perspective camera

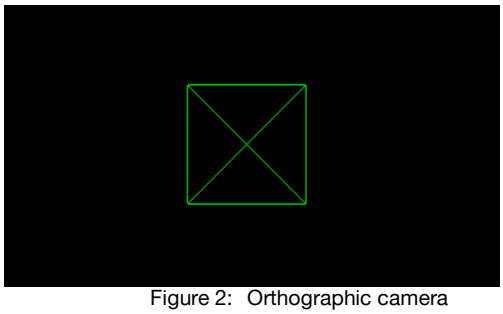

Figure 2: Orthographic camera

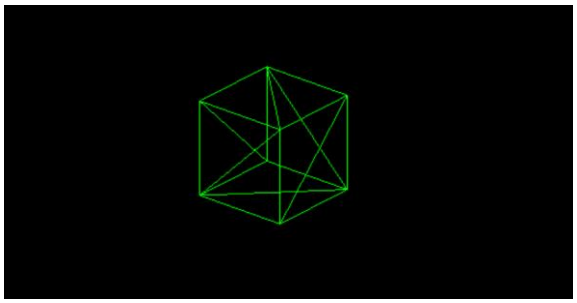

Figure 3: example orbit control

Outra função implementada foi o uso do *OrbitControls* permitindo a interação do utilizador com a cena 3D. Isto facilita a manipulação da vista câmera, sendo possível rodar, aproximar e afastar a cena, proporcionando uma experiência interativa. Para a atualização repetida das ações do utilizador, é necessário fazer uma chamada na função animate, que é responsável por atualizar os objetos e a posição da câmera na cena.

Lighting and materials

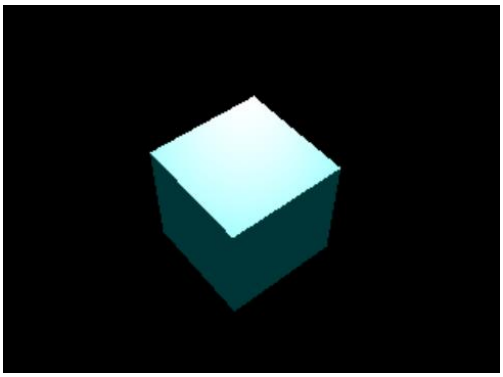

Figure 4: lighting and materials example

Nesta implementação, exploramos as luzes e os materiais para realçar a cena. Começamos adicionando uma luz direcional. Para isso, definimos vários atributos: a cor da luz (branca), a intensidade (1.0) que determina o brilho dela, entre outros. A direção da luz será da sua posição (0,5,0) para o centro da cena.

Inicialmente, não observamos mudanças na cena, o que se deveu ao tipo de material que estávamos a utilizar. Ao substituir o *MeshBasicMaterial* por um *MeshPhongMaterial*, percebemos que o cubo agora é iluminado pela *DirectionalLight*, e a intensidade da luz afeta o brilho do objeto.

Além disso, para melhorar a iluminação geral da cena, adicionamos uma luz ambiente. Essa luz é utilizada para adicionar uma iluminação difusa geral, simulando o efeito da luz sendo espalhada em todas as direções pelas superfícies da cena, criando um nível uniforme de brilho. A luz ambiente é normalmente utilizada em conjunto com outros tipos de iluminação como *PointLight* e *DirectionalLight* de maneira a criar efeitos de iluminação mais realistas.

Shading

Neste exercício, partimos da adaptação do exemplo original do cubo rotativo para renderizar uma esfera.

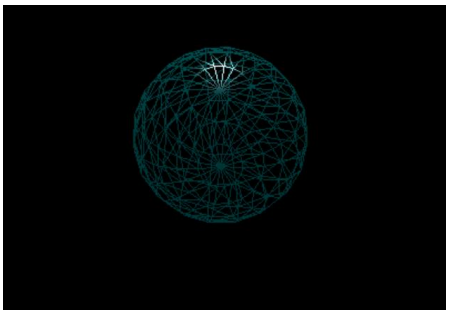

Figure 5: mesh sphere wireframe

Os parâmetros *widthSegments* e *heightSegments* foram ajustados para controlar o nível de detalhe da esfera, determinando o número de segmentos horizontais e verticais que compõem a sua geometria. O aumento desses valores resulta em uma geometria esférica mais detalhada, com um número maior de faces. Por outro lado, a redução desses valores resulta em uma esfera com menos faces, diminuindo o nível de detalhe e assim, simplificar a representação geométrica.

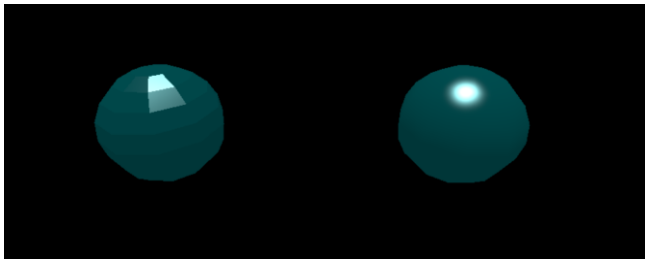

Figure 6: FlatShading vs not FlatShading

Após isso, adicionamos uma nova esfera e outros elementos à cena. Neste ponto, experimentamos a propriedade *flatShading* presente nos materiais dos objetos na cena. Ao alterar entre os valores true e false, pudemos observar diferenças na aparência das superfícies das esferas. Com *flatShading*, as superfícies parecem ser compostas de polígonos planos com bordas claramente definidas, enquanto que sem *flatShading*, as superfícies dos objetos adquirem uma aparência mais suave, parecendo serem lisas e curvas.

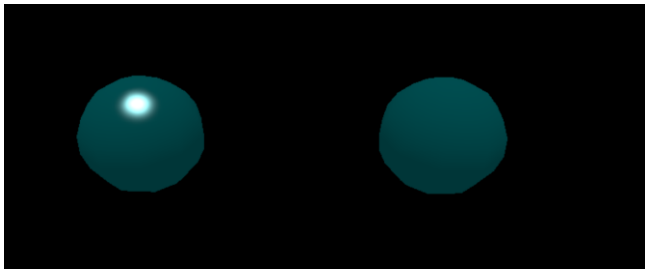

Figure 7: with and without specular and shiny

Aplicamos o *MeshLambertMaterial* à esfera inicial, removendo as componentes *specular* e *shininess*. Notamos a ausência de realces ou reflexos na superfície. A superfície adquiriu uma cor uniforme, sem variação de brilho em resposta à iluminação. Essa mudança tem um impacto significativo na aparência do objeto, uma vez que os realces e reflexos desempenham um papel muito importante na criação da sensação de profundidade tridimensional e na transmissão das propriedades materiais do objeto.

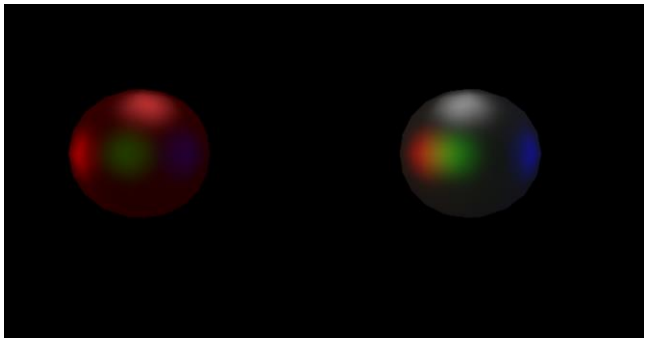

Figure 8: Addition of color

Finalmente, adicionamos o material de *red rubber* a uma esfera e *black rubber* a outra. Adicionamos também as diferentes luzes direcionais conforme pedido. Notamos que as luzes com cor azul e vermelha eram visíveis com bastante clareza na superfície esférica, enquanto que a luz de cor verde só é perceptível numa das esferas ao mover a câmera para visualizar a parte de trás das esferas, devido à sua posição fixa em (0,0,-5). De forma a poder observar-se a luz verde em ambas as esferas, foi necessário aumentar o valor do ângulo.

Transparency

Neste exercício, começamos com as duas esferas criadas no exemplo anterior e adicionamos duas esferas maiores ao redor delas. Em seguida, aplicamos o material *glass* fornecido para esses novos modelos. O resultado obtido está representado na figura abaixo.

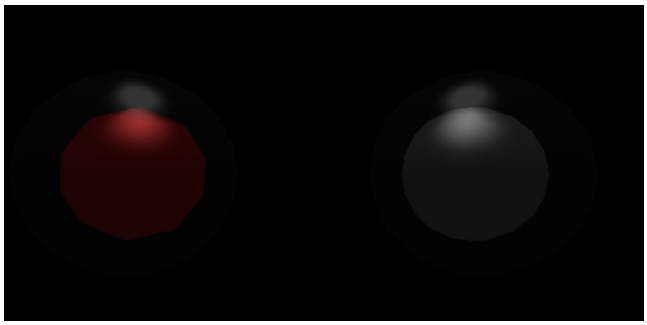

Figure 9: glass spheres example

Transformations- scale and rotation

Nesta implementação, criamos um objeto novo usando o *THREE.Object3D* para ter um objeto único com várias meshes, sendo estas as quatro rodas e o corpo do carro. Devido a não ser necessário materiais e formatos diferentes para as rodas, apenas criamos uma geometria e um material para as quatro rodas, e modificamos a posição das rodas para a localização correta relativa ao corpo do carro, usando a função *position.set()*.

A matriz de transformação dos objetos está de acordo com o esperado, onde o corpo do carro se encontra na posição 0,0,0(a esquerda. Uma das rodas, tem o seu centro num dos vértices inferiores do corpo do carro, como é possível observar na matriz de transformação(a direita).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 10: Matrizes de transformação do corpo e da roda frontal esquerda

Transformations- rotations

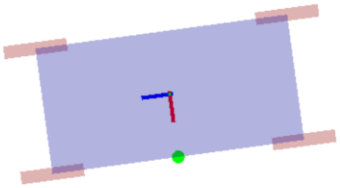

Figure 11: Carro em rotação em torno do ponto pedido

Para a rotação foi necessário implementar um sistema de eixos, usando cilindros com várias cores, para diferenciar os eixos, sendo o cilindro vermelho o eixo X, o cilindro

verde o eixo Y e o cilindro azul o eixo Z para facilitar a visualização e compreensão do carro no espaço 3D.

De modo a criar um carro mais realista, criamos rodas cilíndricas com um raio de 0.5 e altura de 0.2, nas mesmas posições que foram utilizadas no exemplo anterior.

Para mostrar a rotação correta do carro ao longo de um ponto, sendo este (0,0,-1), criamos uma pequena esfera verde para ser mais fácil visualizar se o carro está a fazer a rotação corretamente.

A rotação do carro em redor deste ponto específico é calculada passo a passo na função *animate*, que modifica as posições X e Z baseando-se no seno e coseno do ângulo que evolui ao longo do tempo. Com este método é possível garantir que o carro se mova a uma velocidade constante e de maneira circular em torno do ponto definido.

Outra modificação é a orientação do carro. Originalmente o veículo ao fazer a rotação, tinha o eixo Z, sendo este a frente do carro, voltado diretamente para o centro da esfera. No entanto, ao adicionar $\pi/2$ ao ângulo calculado, foi possível corrigir a animação, criando assim uma animação que simula um carro a fazer uma trajetória circular em torno de um ponto central.

References

[1] *three.js – Javascript 3D library*. (2019). Threejs.org.

[2] *ua_infovis/Three.js/Lesson_02 at master·pmdjdias/ua_infovis*. (n.d.).