Relatório Aula prática 1

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* 1 da cadeira Visualização de Informação. Com este trabalho, foi possível obter conhecimentos básicos acerca da biblioteca *Three.js*, como por exemplo, câmara, materiais, geometrias, animações, entre outros.

First Example

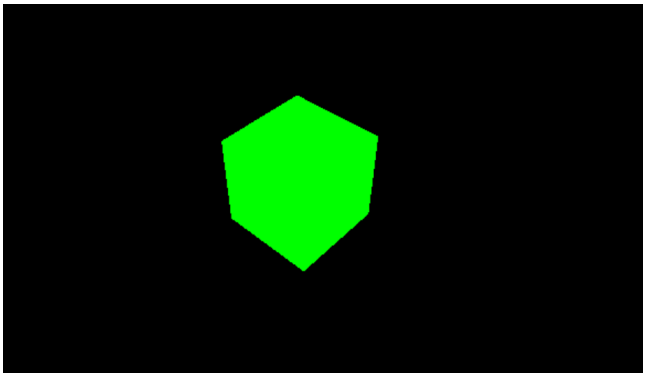

Figure 1: First Example

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

Começamos por definir a cena, a câmara e por fim o *renderer.* De seguida, criamos o cubo, onde tivemos de definir a sua geometria, o seu material. Adicionamos o cubo à cena de maneira a ser possível observá-lo e alteramos a posição da câmara.

Por fim, criamos uma função de render, onde foi adicionada a animação do cubo.

2D Primitives

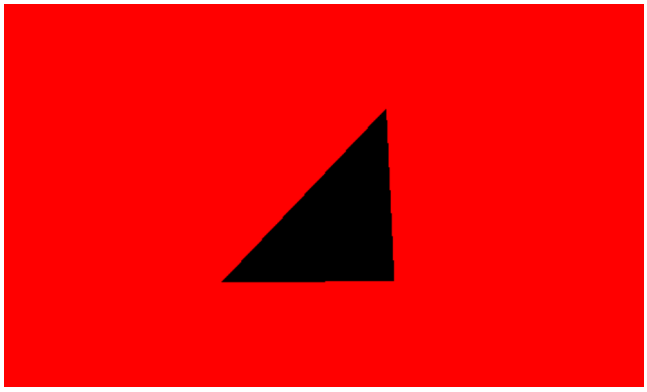

Figure 2: 2D Primitives

Nesta implementação, o exemplo anterior foi modificado para visualizar um triângulo 2D preto sobre um fundo vermelho. Para isso, alteramos a geometria do objeto para um triângulo utilizando as coordenadas dos vértices fornecidas. Foi criado um novo material para o triângulo, colocando a cor preta. Por fim, para alterar a cor de fundo para vermelha usamos a função *setClearColor* do *Renderer.*

Addition of color

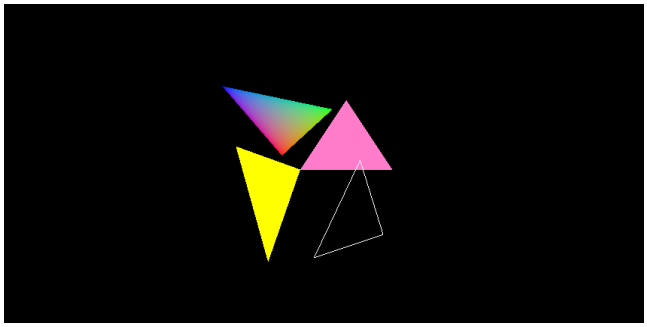

Figure 3: Addition of color

Nesta implementação, alteramos o exemplo *2D Primitives* de forma a ter 4 triângulos em vez de um, para explorar a adição de cores. Começamos então por adicionar os vértices fornecidos de três triângulos e armazená-los na variável *vertices*. De seguida, criamos uma variável *colors* que armazena as cores de cada triângulo. Para um dos triângulos, foi criada uma malha diferente, de maneira a só utilizar o atributo *wireframe neste. Relativamente à malha com os 3 triângulos, utilizamos o argumento THREE.DoubleSide* de maneira a renderizar ambas as faces dos triângulos, mesmo aquelas que têm orientação oposta à câmara.

Viewport Update

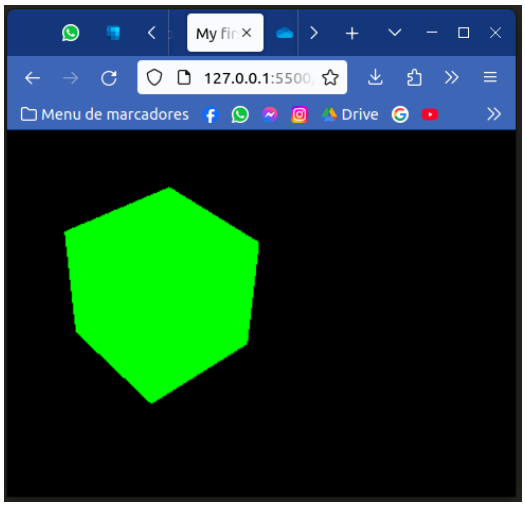

Figure 4: Viewport não implementado

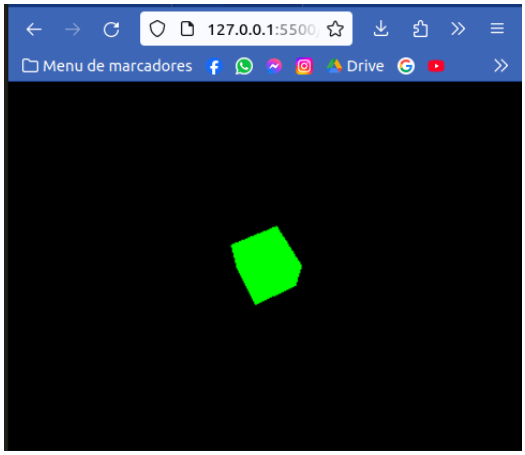

Figure 5: Viewport implementado

Nesta implementação, resolvemos o problema do primeiro exemplo, que consiste na janela de visualização não ser atualizada quando o tamanho da janela do navegador mudava.

Para resolver este problema, usamos *eventListener,* que ao detectar uma alteração no tamanho da janela, ajusta automaticamente o tamanho do *renderer, da câmara e a matriz de projeção da mesma. Assim, garantimos uma visualização adequada da cena.*

Other Primitives

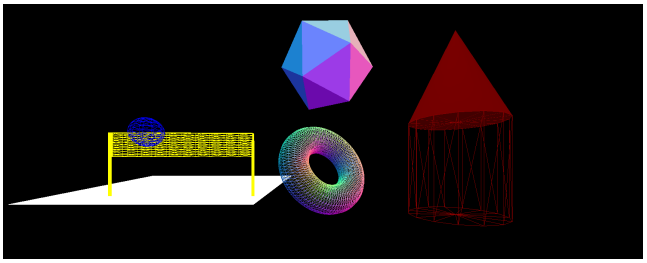

Figure 6: Other primitives

Neste exercício, exploramos diversas funcionalidades do Three.js, abrangendo geometrias, materiais, animações, cores, entre outros. Criamos vários objetos, cada um com as suas características e movimentos. Os objetos implementamos foram:

- Cilindro;
- Cone;
- Esfera;
- *Box*;
- *Icosahedron*;
- *Torus;*

Aplicamos o atributo wireframe ao *Torus, ao cilindro e à rede de vôlei (box).*

Para examinar as diferenças nos tipos de materiais disponíveis no Three.js, utilizamos o *MeshBasicMaterial*, *MeshMatcapMaterial* e *MeshMatcapMaterial*.

Adicionamos animações aos objetos, permitindo rotações em torno dos eixos para o *Torus* e para o *Icosahedron*. Experimentamos diferentes valores para compreender como as rotações influenciam a visualização dos objetos. Por fim, adicionamos uma

translação circular à esfera de maneira a simular o movimento da bola num jogo de vôlei.

References

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

[2] *ua_infovis/Three.js/Lesson_01 at master·pmdjdias/ua_infovis*. (n.d.).