CCI - 36
Trabalho 1º Bimestre
Realidade Aumentada

I. Alunos

- Carlos Renato de Andrade Figueiredo
- Matheus da Silva Martins
- Samara Ribeiro Silva

II. Descrição

A. Composição da página HTML

Na página *html* é possível observar o trabalho desenvolvido com o *three.js* no canvas. No canvas aparece a imagem escolhida, que é uma foto de um cubo mágico. E esse item foi escolhido por possuir arestas bem definidas, o que facilitou no momento de cálculo da matriz de câmera. Como animação foi desenhado um carro que anda em torno de um dos quadrados da parte de cima do cubo mágico.

Como iterações com o canvas é possível ocultar o *Grid Helper*, o *Axes Helper* e o carro através de *checkbox*, ou por teclas específicas definidas. Caso seja necessário, também é possível parar o carro, utilizando a tecla "x" ou clicando no *checkbox*.

Fora disponibilizados na página os *links* contendo o github com o trabalho e o notebook com o script utilizado para calcular a matriz câmera e a posição.

B. Modelagem da animação 3D

A criação da animação 3D começa com a função "makeCar" que começa com a criação de um "Group" da biblioteca *three.js* para agrupamento de objetos. Com a função "Group" foi possível juntar 4 objetos da função "makeWheel" para representar as 4 rodas do carro, a parte do "corpo" do carro guardada na variável "main" criado pela função "Mesh" da mesma biblioteca, a parte da cabine do carro é feita da mesma forma que as outras, sendo representada pela variável "cabin".

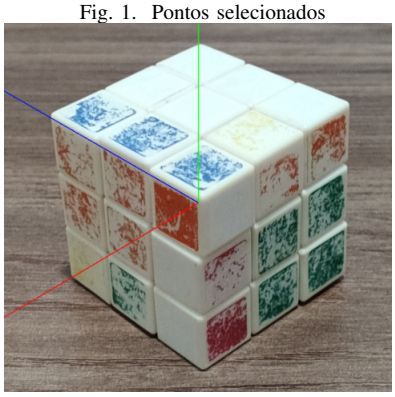
A animação acontece quando chamamos a função *animate()*, que por sua vez chama a função *updatecar()*(responsável por fazer o carro se movimentar no plano XZ) e após renderiza a cena e a câmera.

Para o carro realizar o movimento de curvas para direita ou para esquerda é feito uma rotação do carro no eixo y colocando um ângulo na variável "object_Car.rotation.y".

C. Definição dos eixos X,Y,Z

Os eixos foram escolhidos com base na imagem do cubo de forma que o plano XZ fica na parte superior do cubo mágico, o eixo y fica perpendicular apontando para cima utilizando como referência a aresta do cubo e a origem do plano cartesiano fica em uma das arestas do cubo, conforme pode ser verificado na figura 1.

Fig. 1. Pontos selecionados



D. Cálculo da matriz da câmera

A matriz de projeções P foi calculada conforme:

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = P \begin{bmatrix} X_i^w \\ Y_i^w \\ Z_i^w \\ 1 \end{bmatrix}$$

Efetuando as devidas operações algébricas é possível montar um sistema linear homogêneo para descobrir os elementos da matriz de projeção P.

$$A\mathbf{p} = \mathbf{0}$$

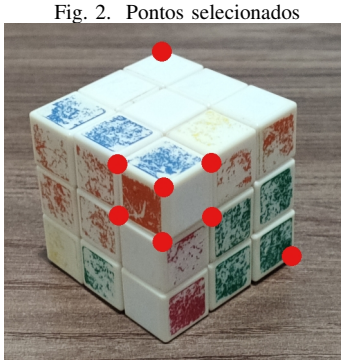
$$A = \begin{matrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 1 & 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 1 & 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \cdots & & & & & & & & & & & - \end{matrix}$$

O vetor **p** corresponde aos elementos da matriz P dispostos em um vetor coluna iniciando com os elementos da primeira linha e terminando com os da última.

O elementos de p foram calculados conforme *script* e Python disponível em encurtador.com.br/iGIL0.

Foi selecionada a imagem de fundo e escolhido um objeto com os eixos bem definidos para selecionar os pontos para os cálculos. Na figura 2 é possível observar os pontos selecionados.

Fig. 2. Pontos selecionados



Para definir os pontos (u_i, w_i) utilizou-se o *GIMP* e as coordenadas do *Real World* é conhecida.

Foram calculados também o centro da câmera C, a matriz intrínseca K e extrínseca $[R \mid -RC]$ através da decomposição de P, conforme [2].

$$P = [M \mid -MC] \tag{1}$$

$$C = M^{-1}[-MC] \tag{2}$$

$$P = K[R \mid -RC] \tag{3}$$

Após os cálculos, encontrou-se a seguinte matriz [R — RC]:

$$[R|-RC] = \begin{bmatrix} 0.69 & -0.71 & -0.01 & 0.05 \\ -0.44 & -0.42 & -0.79 & 0.34 \\ 0.56 & 0.55 & -0.61 & -6.98 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

O centro C encontrado foi:

$$C = \begin{bmatrix} 4.04 & 4.07 & -3.99 \end{bmatrix}$$

A matriz [R — RC] foi aplicada a camera da animação por meio do comando ”camera.applyMatrix4()”. Após isso foi inserido os dados de C na camera através do comando ”camera.position.set()”. Depois, com esses dados inseridos, atualizou-se a matrix world através do comando ”camera.updateMatrixWorld()”.

E. Controle de visualização

Para o controle de visualização foram criadas *checkboxes* que fornecem opções de ocultar a animação do carro, ocultar o grid, ocultar o axes ou fazer o caro parar de andar. Também é possível executar essas mesmas ações apertando algumas teclas definidas.

III. Modo de execução e interação com a animação

Para executar o projeto deve-se descomprimir a pasta o arquivo compactado disponibilizado, ou fazer o download do projeto do github. O projeto contém na raíz o arquivo html ”projeto.html” e um README.md com instruções. As imagens utilizadas para o trabalho foram armazenadas na pasta ”img” enquanto os javascripts do three.js e o script com o código da animação ficam na pasta ”js”. A folha de estilos permaneceu fundida junto com a página html, escrita na parte do head .

Para execução utilizou-se uma extensão do *Visual Studio Code* chamada ”Live Server”, podendo ser encontrada em https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer. Após instalado, abra a página ”projeto.html” e basta clicar em ”Go Live”, no canto inferior direito da tela, conforme pode ser observado na imagem 3. Automaticamente a página do browser se abrirá com a animação.

Fig. 3. Instruções para executar o Live Server

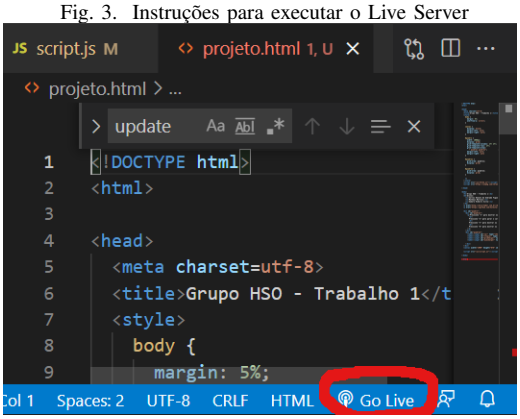
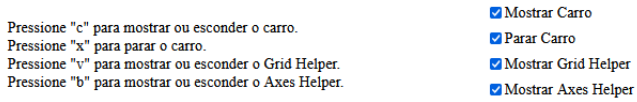


Fig. 4. Instruções para ocultar objetos



Como pode ser observado na figura 4, pressionar a tecla ”c” oculta a animação do carro , pressionar a tecla ”x” faz com que o carro pare de andar, pressionar a tecla ”v” oculta o GridHelper e pressionar a tecla ”b” oculta o AxeHelper. Pressionando novamente qualquer uma das teclas citadas ou apertando no checkbox, a animação correspondente volta a aparecer.Assim a interação permite avaliar os objetos presentes na animação individualmente com a ocultação dos outros.

References

[1] Departamento de Computação – ITA, Prof. Carlos Henrique Quartucci Forster , Fundamentos de Computação Gráfica - Vídeo Aula e Slides de CCI-36

[2] K. Simek, Dissecting the Camera Matrix, Part 1: Extrinsic/Intrinsic Decomposition, [online] Available: http://ksimek.github.io/2012/08/14/decompose/.