

# Introdução à Computação Gráfica

Emmanuella Faustino Albuquerque

20170002239

## Atividade Prática 2: Implementação do Pipeline Gráfico

11 de outubro de 2021

### VISÃO GERAL

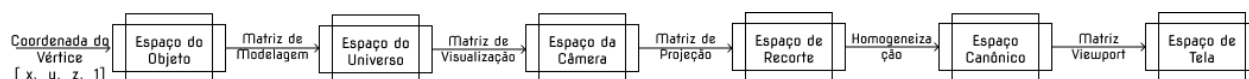
Nesta atividade foram implementados todos os estágios do Pipeline Gráfico, por meio das matrizes de transformação: Matriz de Modelagem(Model), Matriz de Visualização(View), Matriz de Projeção(Projection), Matriz Viewport e a Homogeneização.

### ESTRATÉGIAS

Primeiramente foi preciso derivar os vetores da base da câmera,  $X_{cam}$ ,  $Y_{cam}$ , e o  $Z_{cam}$ , a partir do vetor de posição(**P**) da câmera, do **Look At** e do **Up**. Com eles, foi possível encontrar o vetor direção(**D**) e montar a matriz de visualização a partir das equações abaixo.

$$\vec{D} = \text{LookAt} - \vec{P} \quad Z_{cam} = -\frac{\vec{D}}{|\vec{D}|} \quad X_{cam} = \frac{\vec{U} \times Z_{cam}}{|\vec{U} \times Z_{cam}|} \quad Y_{cam} = \frac{Z_{cam} \times X_{cam}}{|Z_{cam} \times X_{cam}|}$$

### Estrutura do Pipeline



### Matrizes utilizadas e Homogeneização

Para realizar as transformações geométricas do Pipeline Gráfico, foram utilizadas as matrizes listadas abaixo, além da matriz de modelagem carregada com a identidade. A homogeneização foi realizada pela divisão do vetor dos vértices no espaço de recorte pelo escalar W.

$$\vec{t} = \vec{P} - \vec{o}, \text{ onde} \\ \vec{o} = \text{origem esp. universo}$$

$$M_{projection} = M_p \times M_T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & -\frac{1}{d} & 0 \end{bmatrix}$$

$$M_{view} = B^T \times T = \begin{bmatrix} X_{cam}(x) & X_{cam}(y) & X_{cam}(z) & 0 \\ Y_{cam}(x) & Y_{cam}(y) & Y_{cam}(z) & 0 \\ Z_{cam}(x) & Z_{cam}(y) & Z_{cam}(z) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -tx \\ 0 & 1 & 0 & -ty \\ 0 & 0 & 1 & -tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{viewport} = S \times T = \begin{bmatrix} width/2 & 0 & 0 & 0 \\ 0 & height/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### Rasterização

Após construir as matrizes acima, e aplicá-las a cada um dos vértices do objeto, foi realizada a rasterização utilizando o algoritmo do ponto médio desenvolvido na atividade anterior. Com o

---

propósito de organizar melhor o código, a função drawLine(i.e o algoritmo do ponto médio) foi incluída dentro da classe Canvas disponibilizada pelo professor.

Para realizar a rasterização de cada linha, cada aresta do objeto indicado foi analisada, ligando cada vértice.

## RESULTADOS

### Cena Final: Cubo

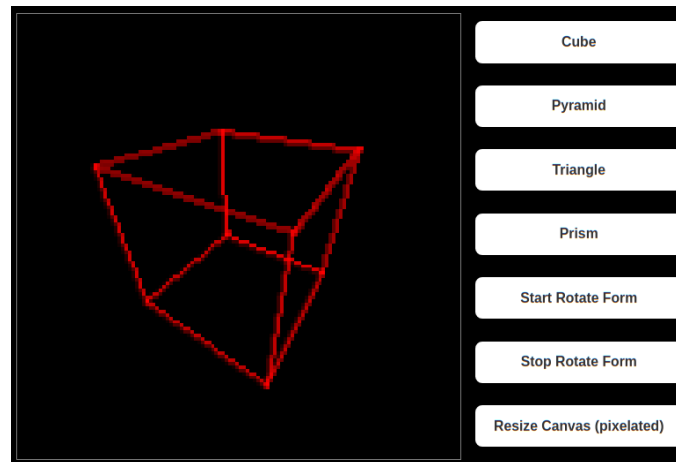


Imagem 1: Cubo Rasterizado

### Cena Final: Outras Formas Geométricas

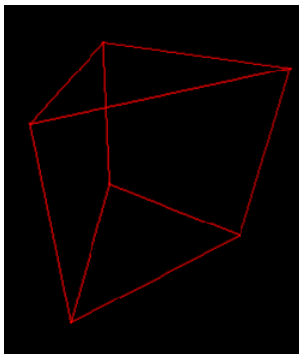


Imagem 2: Prisma com canvas redimensionado 512x512

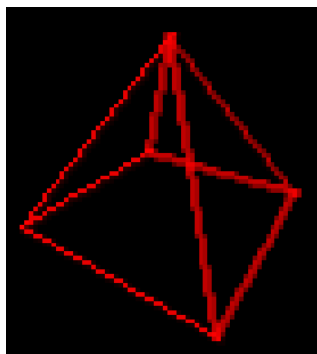


Imagem 3: Pirâmide com canvas redimensionado 128x128

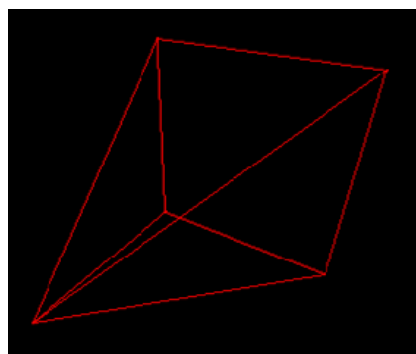


Imagem 4: Triângulo com canvas redimensionado 128x128

### Dificuldades

A maior dificuldade foi conseguir encontrar formas geométricas descritas com vértices e arestas, já que normalmente os modelos .obj são descritos com vértices e faces.

### Possíveis melhorias

Seria interessante reconstruir a matriz de projeção com o view frustum igual ao utilizado pelo opengl, no qual no espaço canônico as coordenadas vão de -1 a 1.

---

## REFERÊNCIAS BIBLIOGRÁFICAS

OpenGL Transformation. Song Ho Ahn. Disponível em:

[http://www.songho.ca/opengl/gl\\_transform.html](http://www.songho.ca/opengl/gl_transform.html). Acesso em: 15 de outubro de 2021.

Vector4. Threejs. Disponível em: <https://threejs.org/docs/#api/en/math/Vector4>. Acesso em: 15 de outubro de 2021.

Prism Reference. HWS. Disponível em: <https://math.hws.edu/eck/cs424/f15/lab7/prism.png>.

Acesso em: 15 de outubro de 2021.

Timeouts e intervalos. MDN Web Docs. Disponível em:

[https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Asynchronous/Timeouts\\_and\\_intervals](https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Asynchronous/Timeouts_and_intervals)  
. Acesso em: 15 de outubro de 2021.