TRABALHO 2 - COMPUTAÇÃO GRÁFICA

Alunos: Dayane Felix de Freitas - 11511188
Gabriel Belarmino - 11414176

O segundo trabalho da disciplina de computação gráfica do semestre 2017.2 consiste na implementação de um pipeline, conhecendo e desenvolvendo cada etapa .O objetivo principal é transformar vértices que estão no espaço 3D e leva para o espaço da tela que é 2D, através da rasterização de primitivas.
O pipeline gráfico consiste numa sequência de transformações para levar todos os vértices do espaço do objeto para o espaço da tela através de multiplicações de matrizes, a seguir temos todas a etapas que serão explicadas ao longo deste documento:

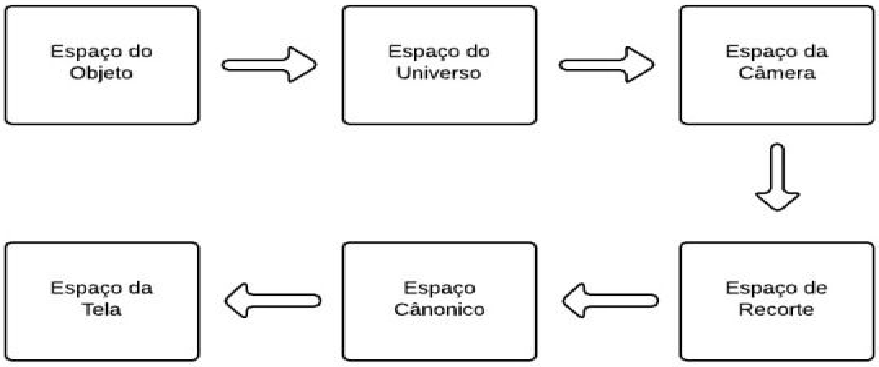


Figura 1 : Pipeline gráfico.

ESPAÇO DO OBJETO → ESPAÇO DO UNIVERSO
Nesta etapa transformamos todos os vértices através de multiplicações pela matriz de modelagem,esta matriz é resultante de uma sequência de transformações geométricas como escala, translação e rotação por exemplo,que ajudam a posicionar o objeto no espaço do universo.Inicialmente temos a matriz identidade que será combinada com as transformações desejadas, caso não queira realizar nenhuma transformação geométrica, basta usar apenas a identidade.

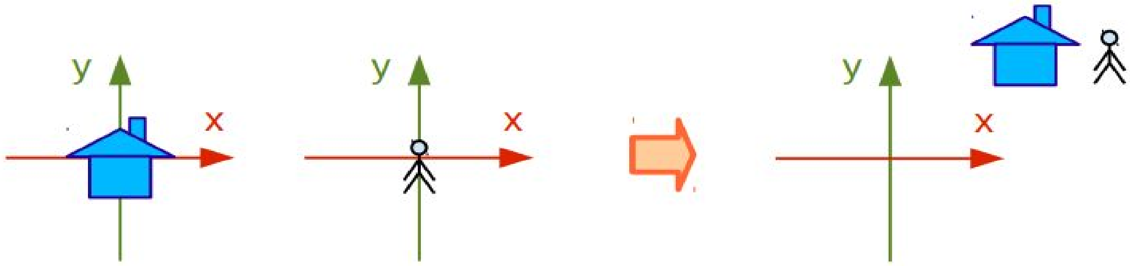


Figura: Espaço Objeto - Espaço do universo

A seguir teremos todas as matrizes de transformações geométricas que podemos aplicar ao nosso objeto:

ESCALA : Responsável pelo aumento do objeto, tanto no eixo X quanto no Y.

ROTAÇÃO: Movimenta o objeto em graus em torno dos eixos.

TRANSLAÇÃO: Desloca o objeto no plano,usando as posições dos eixos.

Matriz para representar a Escala, sendo os componentes Sx, Sy e Sz a variação de tamanho desejada.

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz para a representação da Translação, com componentes Dx, Dy e Dz, que iram representar o valor de deslocamento de cada componente.

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 3 : Escala e Translação.

$$RotX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta x) & -sin(\theta x) & 0 \\ 0 & sin(\theta x) & \cos(\theta x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

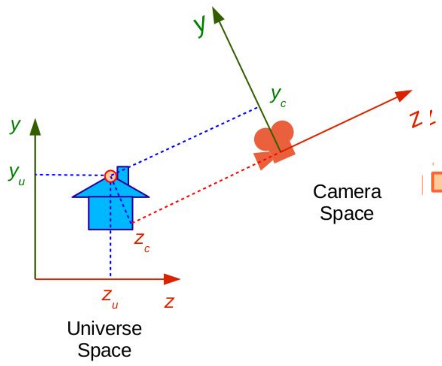
$$RotY = \begin{bmatrix} \cos(\theta y) & 0 & sin(\theta y) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta y) & 0 & \cos(\theta y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$RotZ = \begin{bmatrix} \cos(\theta z) & -sin(\theta z) & 0 & 0 \\ sin(\theta z) & \cos(\theta z) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 4: Rotação nos eixos x, y e z.

ESPAÇO DA UNIVERSO → ESPAÇO DA CÂMERA

Nesta etapa todos os vértices passarão pela matriz View para saírem do espaço do universo e serem representados no espaço da câmera, primeiro definimos a posição da câmera e para onde ela está olhando usando o sistema de coordenadas do universo.



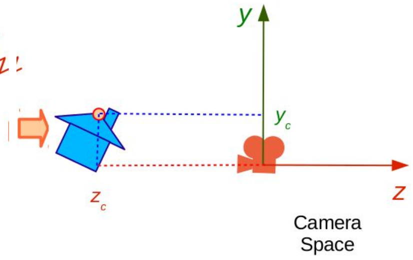


Figura 5: LookAt da Câmera. Figura 6: Espaço da Câmera.

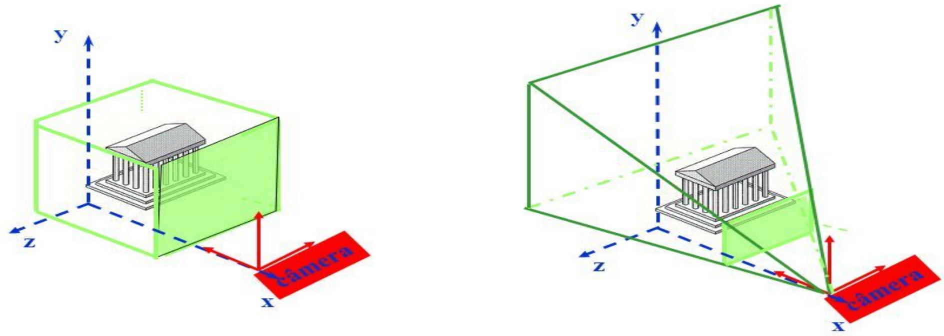
A matriz view é composta por duas transformações geométricas: Translação e Rotação, que devem acontecer nesta mesma ordem.Fazendo a multiplicação dessas duas matrizes conseguimos trazer o objeto para o centro da câmera, esta matriz também é composta por três vetores: Posição_câmera ,Up_câmera e Lookat, devidamente normalizados como está no trecho de código a seguir:

```
myglLookAt(
    0.0f, 0.0f, 5.0f,   // Posição_câmera
    0.0f, 0.0f, 0.0f,   // Lookat
    0.0f, 1.0f, 0.0f    // Up_câmera
);
```

Figura 7: Código correspondente a configuração de câmera.

ESPAÇO DA CÂMERA → ESPAÇO DE RECORTE

Neste passo do pipeline a matriz utilizada será a de Projeção, que será responsável por trazer os pontos para o espaço de recorte, aqui os vértices do objeto serão projetados no viewPlane (plano de visão da câmera). O objeto pode ser projetado de duas formas: Ortogonal ou Perspectiva. Se escolhermos representar de forma ortogonal teremos a certeza de que o paralelismos entre os vértices será preservado, caso a escolha não seja esta, a noção de perspectiva será aplicada, ou seja, teremos a sensação de profundidade da cena ou objeto na tela.



• Projeção ortográfica x projeção perspectiva

A transformação dos vértices é feita através da multiplicação pela matriz de projeção, lembrando que estamos usando coordenadas homogêneas (x,y,z,w),nosso w assumirá valores diferentes de 1 e teremos a projeção do objeto no view plane.

$$\begin{bmatrix} x \\ y \\ z \\ -z/n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/n & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 8: Multiplicação dos vértices (x,y,z,w) pela matriz de projeção.

ESPAÇO DE RECORTE → ESPAÇO CANÔNICO

Nesta etapa do pipeline gráfico temos um gargalo, onde precisamos dividir as nossas coordenadas que estão no espaço homogêneo pela coordenada w, isso fará com que voltemos para o espaço euclidiano e o valor de w passe a assumir novamente o valor igual a 1. Aqui os vértices que não estão no alcance da tela são eliminados, isto causa a sensação de profundidade,ou seja, os vértices mais próximos da câmera são vistos num tamanho maior,e os que estão mais distantes são representados de um tamanho menor.

ESPAÇO CANÔNICO → ESPAÇO DA TELA

Enfim o último passo do nosso pipeline, aqui fizemos a rasterização dos vértices do objeto na tela,isso acontece através da multiplicação pela matriz viewport, que é uma combinação de duas escalas seguida de uma translação.O primeiro passo foi fazer uma inversão de todos os pontos no eixo y (espelhamento),isso faz com que a cena não vá invertida para a tela,segundo passo é realizar duas escalas para que a cena fique em um tamanho de acordo com a tela, utilizamos a largura e altura da tela dividido por 2 ,por último fizemos uma translação para levar o objeto para o espaço da tela, neste passo foi preciso transladar os vértices que tinham valores menores do que zero,tornando-os positivos, assim eles não ficariam fora do processo de rasterização na tela,neste último passo utilizamos o algoritmo de rasterização implementado no trabalho 1.

```
void myglViewport(int x, int y, size_t largura, size_t altura) {
    MyMatrix escala1(4, 4);
    MyMatrix escala2(4, 4);
    MyMatrix traslacao(4, 4);
```

Figura 9: Método ViewPort.

PONTO NO ESPAÇO CANÔNICO → ESPELHAMENTO → ESCALA NOS EIXOS → TRANSLAÇÃO → PONTO NO ESPAÇO CANÔNICO

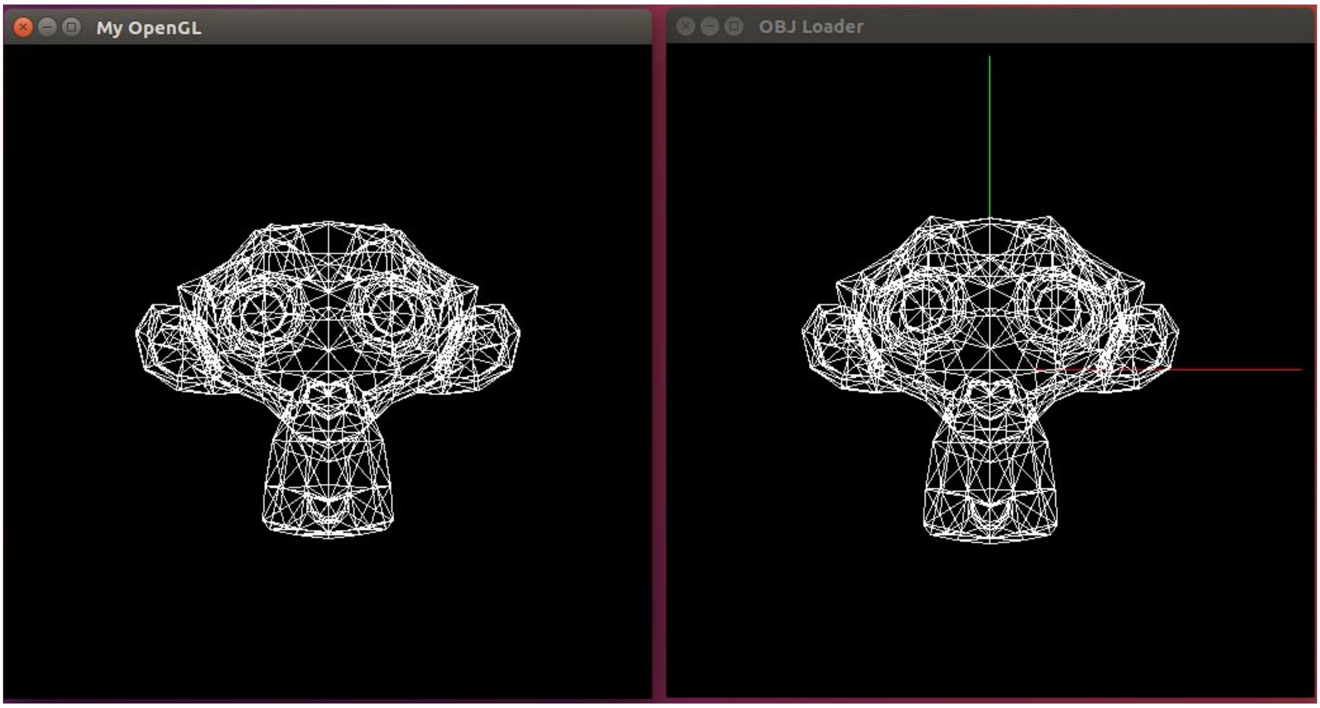


Figura 10: Comparação entre as raterizações desenvolvidas pela dupla ‘My OpenGL’ e o loader do professor ‘OBJ Loader’.

Link para o Vídeo do monkey_head2.obj rotacionado:
https://youtu.be/U3_KuLsOvdI

DIFICULDADES AO LONGO DA IMPLEMENTAÇÃO:

- A parte de passar do espaço canônico para o da tela sem dúvidas foi o mais trabalhoso pois é um processo com etapas especiais.
- Trabalhar a primeira vez com a GLUT ,biblioteca do opengl, gerou algumas pesquisas no google.
- Encontrar o monkey_head2 ,deu um pouco de trabalho no início,mas após encontrar e salvá-lo no devido formato podemos enfim executar o código.

REFERÊNCIAS:

- Slides disponibilizados pelo Professor Christian Pagot.
- https://askubuntu.com/questions/4428/how-can-i-record-my-screen?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa. Acesso em 16 de maio de 2018.
- http://infomitpb.blogspot.com.br/2015/06/pipeline-grafico.html. Acesso em 10 de maio de 2018.