

UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB
DEPARTAMENTO DE INFORMÁTICA
ENGENHARIA DE COMPUTAÇÃO

Integrantes:

Andrea Brito do Nascimento - 11311923

Chaenne Carolina Pessoa - 11406556

Jonas da Silva Antas - 11228383

Disciplina: Introdução a Computação Gráfica
Prof. Christian Pagot

Segunda Atividade

João Pessoa
Maio /2018

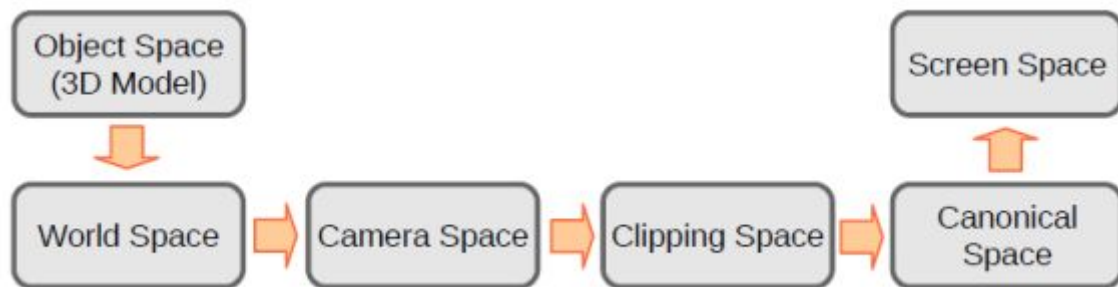
O Pipeline Gráfico:

Objetivo:

O objetivo deste trabalho é implementar o pipeline gráfico proposto na atividade anterior, responsável por levar ao espaço de tela objetos descritos no espaço do objeto. O conteúdo deste trabalho foi ministrado na disciplina de Introdução a Computação Gráfica.

O que é o pipeline gráfico?

Em computação gráfica, o pipeline gráfico é basicamente uma sequência de passos que devem ser seguidos a fim de gerar a representação de um modelo 3D em tela .



Utilizando as funções conceituadas em sala temos como ponto de partida implementar as transformações que darão origem aos vértices descritos no espaço do objeto (object space) até o espaço da tela. Assim aplicamos o conteúdo abordado e usado no primeiro trabalho.

Transformações ao longo do pipeline gráfico:

As transformações realizadas ao longo do pipeline gráfico são implementadas a partir da utilização de matrizes e coordenadas homogêneas.

Espaço do objeto para o espaço do universo:

O espaço do objeto é o espaço onde cada objeto é criado e modelado por meio de suas primitivas geométricas usando o seu sistema de coordenadas próprias.

O primeiro passo é a passagem dos vértices presentes no espaço do objeto para o espaço do universo. A passagem dos vértices se dá a partir da multiplicação das coordenadas de cada vértice por uma matriz de modelagem, onde esta

representa uma combinação de uma ou mais transformações geométricas, que também são matrizes.

As principais transformações dentro do pipeline gráfico são:

Escala:

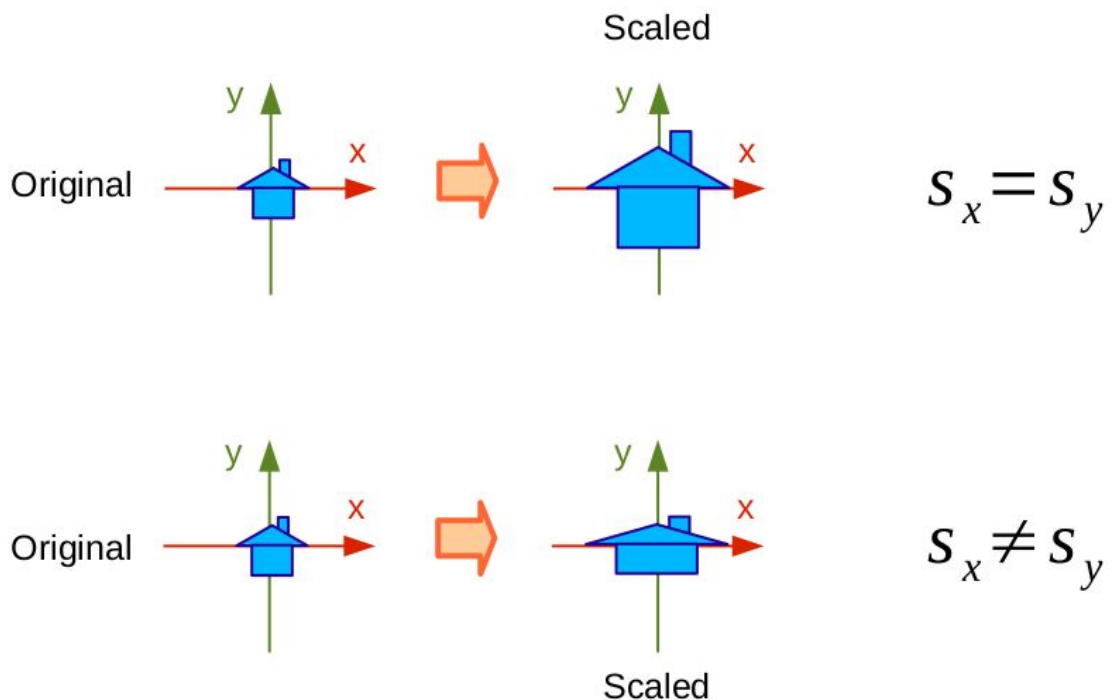
Escalonar um objeto consiste em redimensioná-lo, diminuindo ou aumentando as suas dimensões. Para fazer essa operação, basta multiplicar os valores das coordenadas de um dos vértices do objeto por um fator de escala.

Vale salientar que os vértices possuem valores inteiros, tanto no modelo 3D como no 2D.

Matriz escalar 2D:

$$\begin{aligned}x' &= x \cdot s_x \\ y' &= y \cdot s_y\end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

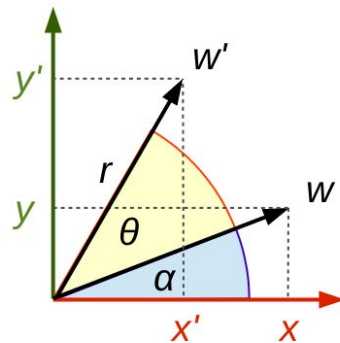
ilustração da mudança de escala:



Rotação:

Equivale a rotacionar os vértices de um objeto a partir de um determinado

ângulo. No espaço 2D, a rotação ocorre em torno da origem do sistema de coordenadas. A matriz de rotação é criada primeiramente definindo um vetor W partindo da origem com posições X e Y , formando assim um ângulo α com o eixo coordenado X . Após esse processo, o vetor W sofre rotação de θ graus no sentido anti-horário definindo assim novas coordenadas X' e Y' .



$$x = r \cos(\alpha)$$

$$y = r \sin(\alpha)$$

$$x' = r \cos(\alpha + \theta)$$

$$y' = r \sin(\alpha + \theta)$$

$$x' = r \cos(\alpha + \theta)$$

$$y' = r \sin(\alpha + \theta)$$

$$x = r \cos(\alpha)$$

$$y = r \sin(\alpha)$$

$$x' = r \cos(\alpha) \cos(\theta) - r \sin(\alpha) \sin(\theta)$$

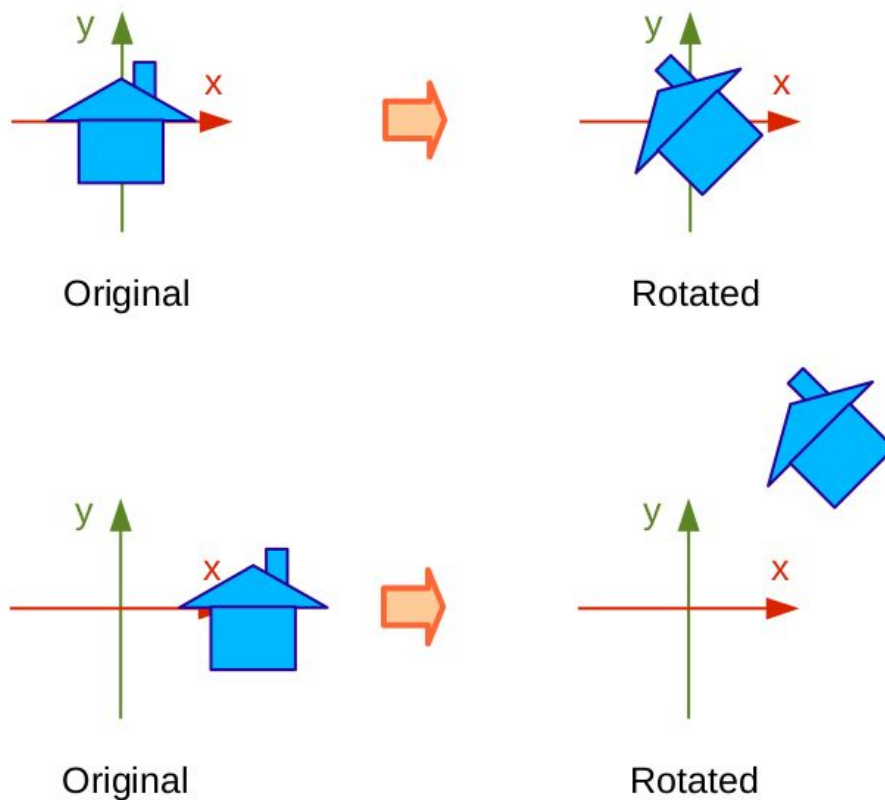
$$y' = r \sin(\alpha) \cos(\theta) + r \cos(\alpha) \sin(\theta)$$

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = y \cos(\theta) + x \sin(\theta)$$



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Matriz de rotação 2D:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Translação:

Transladar um objeto consiste em movimentar os vértices do objeto adicionando unidades ao longo de cada eixo.

$$x' = x + dx$$

$$y' = y + dy$$

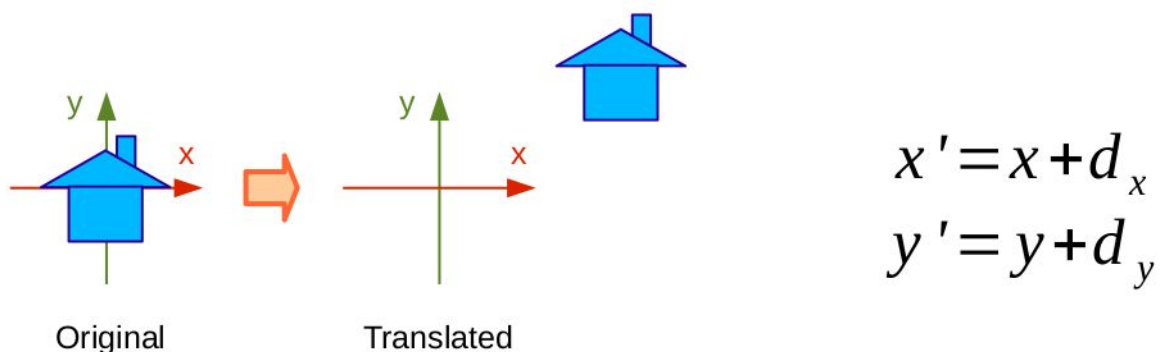
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & dx/y \\ dy/x & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Um problema comum desta abordagem é que X e Y estão inseridos na matriz e com isso criamos uma única matriz para todos os vértices do objeto, onde se por exemplo tivermos um objeto contendo 100 vértices, seria necessário criar 100 matrizes. Com isso os custos computacionais aumentaram muito prejudicando o desempenho da aplicação.

Uma solução seria criar uma matriz de translação utilizando coordenadas homogêneas.

$$\begin{array}{ccccc}
 \text{Euclidean} & & \text{Homogeneous} & & \text{Homogeneous} \\
 \text{space} & & \text{space} & & \text{space} \\
 (x, y) & \Rightarrow & (xw, yw, w) & \xRightarrow{w=1} & (x, y, 1)
 \end{array}$$

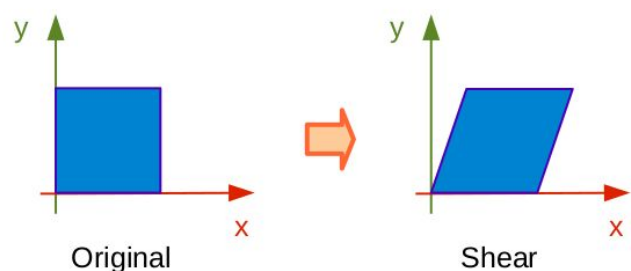
$$\begin{array}{l}
 x' = x + d_x \\
 y' = y + d_y \\
 1 = 1
 \end{array}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Shear:

Uma transformação Shear é outra ferramenta capaz de deformar o objeto dentro do pipeline aplicando transformações. O funcionamento é basicamente manter uma coordenada U parada enquanto os valores presentes na coordenada V mudam ao longo do eixo.

Porém o nosso trabalho não implementa-rá essa transformação.

$$\begin{aligned}
 x' &= x + m_x y \\
 y' &= y
 \end{aligned}$$


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & m_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrix form

Espaço 3D:

Todas as matrizes modeladas até agora foram feitas no espaço 2. Agora veremos essas matrizes no espaço 3D.

Matriz Escala 3D com coordenadas homogêneas :

$$\begin{aligned}
 x' &= x \cdot s_x \\
 y' &= y \cdot s_y \\
 z' &= z \cdot s_z
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matriz Rotação 3D ao longo do eixo X e com coordenadas homogêneas :

$$\begin{aligned}
 x' &= x \\
 y' &= y \cos(\theta) - z \sin(\theta) \\
 z' &= y \sin(\theta) + z \cos(\theta)
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matriz Rotação 3D ao longo do eixo Y e com coordenadas homogêneas :

$$\begin{aligned}
 x' &= x \cos(\theta) + z \sin(\theta) \\
 y' &= y \\
 z' &= -x \sin(\theta) + z \cos(\theta)
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matriz Rotação 3D ao longo do eixo Z e com coordenadas homogêneas :

$$\begin{aligned}
 x' &= x \cos(\theta) - y \sin(\theta) \\
 y' &= x \sin(\theta) + y \cos(\theta) \\
 z' &= z
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matriz Translação 3D e com coordenadas homogêneas :

$$\begin{aligned}
 x' &= x + d_x \\
 y' &= y + d_y \\
 z' &= z + d_z
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matriz Shear 3D com coordenadas homogêneas :

$$\begin{aligned}
 x' &= x + m_x z \\
 y' &= y + m_y z \\
 z' &= z
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & m_x & 0 \\ 0 & 1 & m_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Depois de analisarmos toda a parte teórica do projeto proposto, a implementação das matrizes vistas acima podem ser visualizadas na imagem a seguir:


```

82
83 //*****Espaço Universo para o Espaço da Câmera*****/
84
85 glm::vec3 look_at(0.0f, 0.0f, 0.0f);
86 glm::vec3 camera_position(0.0f, 0.0f, 5.0f);
87
88 //Cálculo do vetor direção
89 glm::vec3 vector_direction = look_at - camera_position;
90
91
92 glm::vec3 camera_up(0.0f, 1.0f, 0.0f);
93
94 //Cálculo dos eixos do novo sistema de coordenadas - Sistema de coordenadas da câmera
95 glm::vec3 z_camera = -normalize(vector_direction);
96 glm::vec3 x_camera = normalize(cross(camera_up, z_camera));
97 glm::vec3 y_camera = normalize(cross(z_camera, x_camera));
98
99 //Cálculo da matriz de rotação
100 glm::vec4 last_line_matrix(0.0f, 0.0f, 0.0f, 1.0f);
101 glm::mat4 B = glm::mat4(1.0f);
102 B[0] = glm::vec4(x_camera, 0.0f);
103 B[1] = glm::vec4(y_camera, 0.0f);
104 B[2] = glm::vec4(z_camera, 0.0f);
105 B[3] = last_line_matrix;
106
107 //Cálculo da matriz de translação
108 glm::mat4 translate = glm::mat4(1.0f);
109 translate[3] = glm::vec4(-camera_position, 1.0f);
110
111 //Matriz view
112 glm::mat4 M_View = transpose(B) * translate;
113
114 //Matriz model view
115 glm::mat4 M_Model_View = M_View * M_Model;
116
117 //*****
118

```

Do espaço do universo para o espaço da câmera

Consiste em levar os objetos do espaço universo para o espaço da câmera. Nessa etapa os vértice devem passar pela **Matriz View**. Para isso criar-se uma sistema de coordenadas da câmera e construir essa matriz. Podemos encontrar a

```

82
83 //*****Espaço Universo para o Espaço da Câmera*****/
84
85 glm::vec3 look_at(0.0f, 0.0f, 0.0f);
86 glm::vec3 camera_position(0.0f, 0.0f, 5.0f);
87
88 //Cálculo do vetor direção
89 glm::vec3 vector_direction = look_at - camera_position;
90
91
92 glm::vec3 camera_up(0.0f, 1.0f, 0.0f);
93
94 //Cálculo dos eixos do novo sistema de coordenadas - Sistema de coordenadas da câmera
95 glm::vec3 z_camera = -normalize(vector_direction);
96 glm::vec3 x_camera = normalize(cross(camera_up, z_camera));
97 glm::vec3 y_camera = normalize(cross(z_camera, x_camera));
98
99 //Cálculo da matriz de rotação
100 glm::vec4 last_line_matrix(0.0f, 0.0f, 0.0f, 1.0f);
101 glm::mat4 B = glm::mat4(1.0f);
102 B[0] = glm::vec4(x_camera, 0.0f);
103 B[1] = glm::vec4(y_camera, 0.0f);
104 B[2] = glm::vec4(z_camera, 0.0f);
105 B[3] = last_line_matrix;
106
107 //Cálculo da matriz de translação
108 glm::mat4 translate = glm::mat4(1.0f);
109 translate[3] = glm::vec4(-camera_position, 1.0f);
110
111 //Matriz view
112 glm::mat4 M_View = transpose(B) * translate;
113
114 //Matriz model view
115 glm::mat4 M_Model_View = M_View * M_Model;
116
117 //*****
118

```

partir de da posição da câmera, o vetor de direção e o vetor up (determina a direção do topo da câmera). Formando uma matriz de rotação.

Do Espaço da câmera para o espaço de recorte:

Neste momento, é feita a conversão dos pontos presentes no espaço da câmera para o espaço de recorte. Esta conversão é feita a partir da multiplicação dos vértices do espaço da câmera pela matriz de projeção. A câmera indica um ponto específico da tela, assim temos 2 perspectivas de visualização sendo elas:

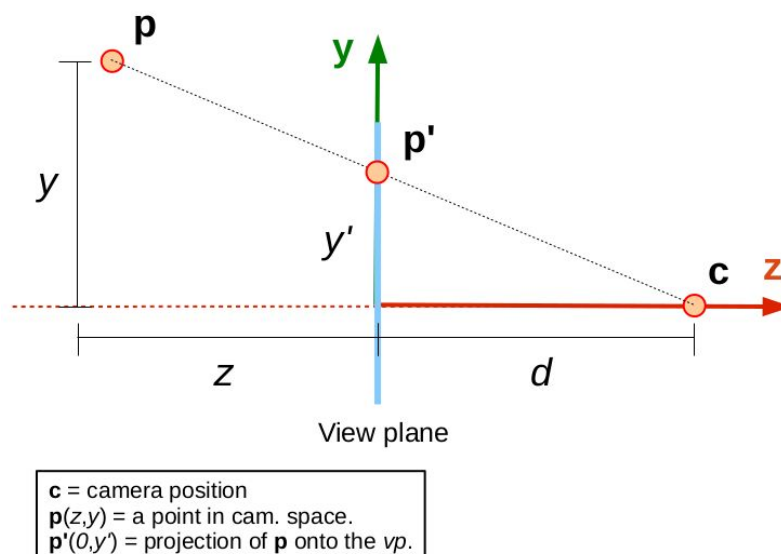
```
118
119  /*Espaço da Câmera para o Espaço de Recorte (CLIPPING)*/
120
121  double d = 1.5;
122
123  glm::mat4 M_Projection = glm::mat4(1.0);
124  M_Projection[2][3] = -1.0/d;
125  M_Projection[3][2] = d;
126  M_Projection[3][3] = 1.0;
127
128  glm::mat4 M_Model_View_Projection = M_Projection * M_Model_View;
129
130  /*****/
```

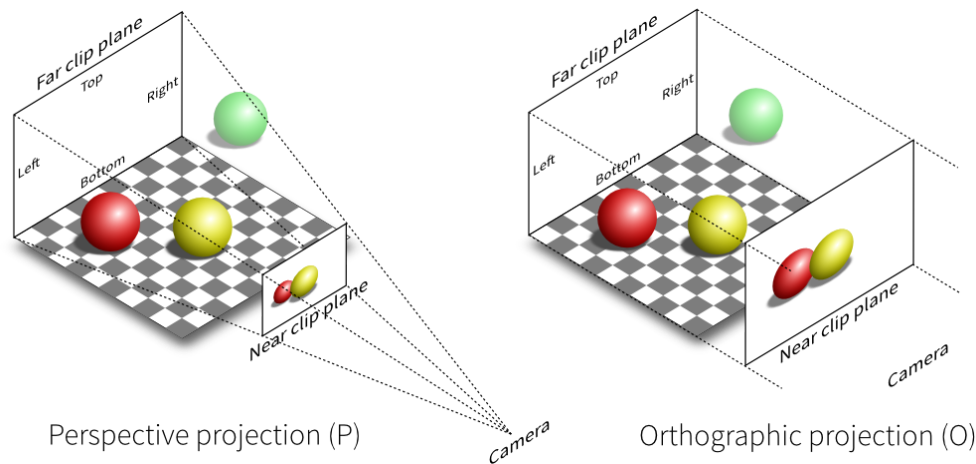
Uma com com distorção perspectiva e outra ortogonal.

Distorção perspectiva: O objeto próximo a câmera tem tamanho maior que objetos distantes da câmera, o que dá uma sensação de profundidade a imagem.

Ortogonal: A sensação de profundidade não existe, o objeto fica “estático” na tela.

A câmera possui um raio de atuação, ou seja, um ângulo de abertura que captura o volume da cena. Apenas objetos contidos na cena são renderizados.





A matriz Model View e a matriz Projection podem ser combinadas dando origem a uma nova matriz chamada $M_Model_ViewProjection$

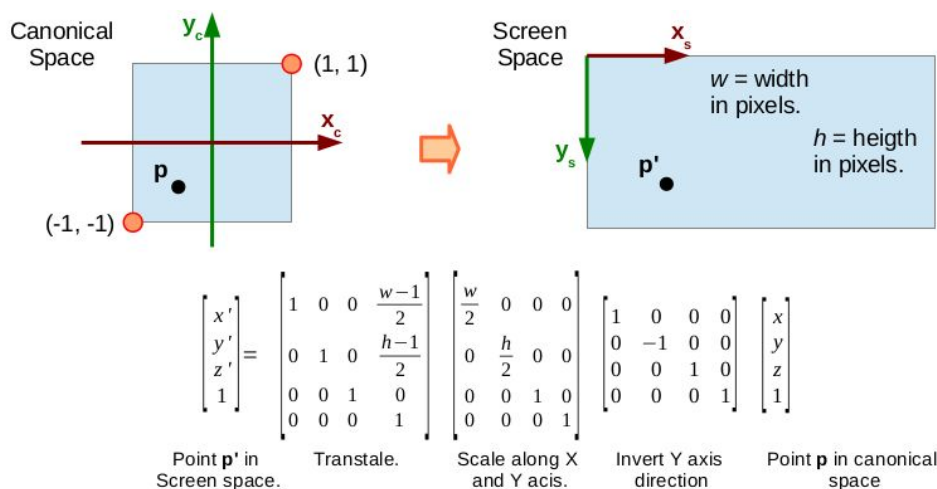
$$M_Model_View_Projection = M_Projection * M_Model_View$$

Espaço de recorte para o espaço canônico

Nesta transformação acontece a divisão por W que é uma coordenada homogênea para que a distorção perspectiva e a sensação de profundidade sejam aplicadas e percebidas na tela.

Espaço Canônico para Espaço de tela

Esta é a última conversão, onde o objeto poderá ser visualizado em tela. Como o centro das coordenadas canônicas é diferente do centro do espaço de tela, teremos que fazer transformações nas matrizes: escalar o objeto (escala anisotrópica), invertendo em y, para que o objeto apareça do lado certo; transladar o objeto para deixar o objeto no centro de tela; escala isotrópica no objeto para que ele fique com largura e altura proporcional ao tamanho da tela. Feito isso, podemos multiplicar os pontos do objeto por essas matrizes de transformação.



```

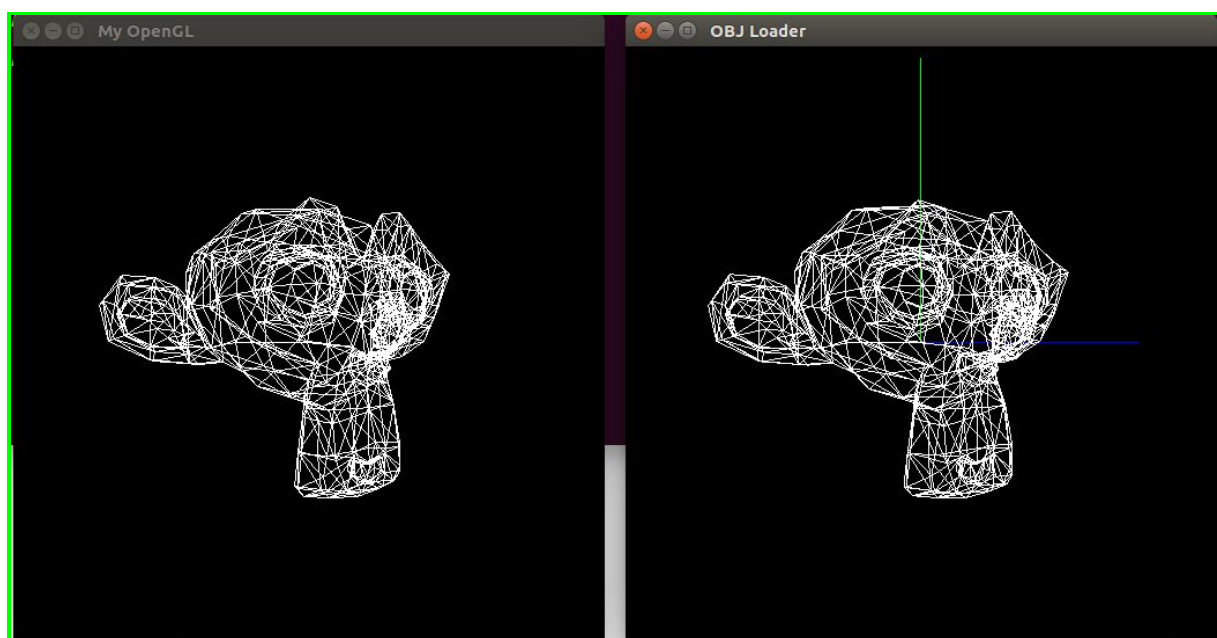
146 /*****Espaço Canônico para o Espaço da Tela*****/
147
148
149 glm::mat4 translation = glm::mat4(1.0);
150 translation[3] = glm::vec4((IMAGE_WIDTH - 1)/2, (IMAGE_HEIGHT - 1)/2, 0.0, 1.0);
151
152 glm::mat4 scale = glm::mat4(1.0);
153 scale[0].x = IMAGE_WIDTH/2;
154 scale[1].y = IMAGE_HEIGHT/2;
155
156 glm::mat4 invertY = glm::mat4(1.0);
157 invertY[1].y = -1.0;
158
159 glm::mat4 Final_Matrix = glm::mat4(1.0);
160 Final_Matrix = translation * scale * invertY;
161

```

Por último fizemos a rasterização dos vértices obtidos ao longo de todo pipeline gráfico. Quando o processo for concluído, teremos um frame ou cena em tela, contendo todo trabalho expresso no pipeline gráfico.

Resultados

O conteúdo visto em sala foi usado em sua totalidade para desempenharmos este trabalho, tanto para objetos bidimensionais quanto tridimensionais. Incluiremos abaixo nossos resultados.



Dificuldades Encontradas

A maior dificuldade foi encontrar os valores compatíveis para o fator de escala, a distância 'd' entre a câmera e o view plane, o ângulo para a matriz de rotação e demais valores para os elementos das matrizes.

Referências

1. Notas de aula do Prof. Christian Azambuja Pagot
2. <http://pt.slideshare.net/thild/computao-grfica-transformaes-geomtricas-no-plano-e-no-espao>
3. http://webserver2.tecgraf.puc-rio.br/~mgattass/LivroCG/07_OpenGLRenderPipeline.pdf