UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE INFORMÁTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
DEPARTAMENTO DE MATEMÁTICA COMPUTACIONAL

CARLOS MAGNO DA SILVA – Matrícula: 20160143331

EMMANUELLA FAUSTINO ALBUQUERQUE SILVA – Matrícula: 20170002239

INTRODUÇÃO A COMPUTAÇÃO GRÁFICA

Trabalho 2 – Implementação de um *Pipeline Gráfico Completo*

JOÃO PESSOA

2019

UFPB – Universidade Federal da Paraíba
Centro de Informática – CI

Disciplina: Introdução a Computação Gráfica – ICG

Professor: Christian Azambuja Pagot

Aluno(s): Carlos Magno da Silva – **Matrícula:** 20160143331
Emmanuella Faustino Albuquerque Silva – **Matrícula:** 20170002239 –

Curso: Ciência da Computação

Semestre: 2018.2

Implementação de um *Pipeline Gráfico Completo*

1. INTRODUÇÃO

O presente projeto consiste na segunda atividade da disciplina de ICG, no presente período, tendo como finalidade familiarizar os alunos com a **estrutura e o funcionamento** do **pipeline gráfico,** através da **implementação de um *pipeline gráfico completo,*** capaz de **transformar vértices** descritos no **espaço do objeto** em ***primitivas rasterizadas*** no ***espaço de tela****.*

2. ATIVIDADE

O trabalho desenvolvido no **trabalho individual I**, foi a implementação de **Algoritmos de Rasterização utilizados em Computação Gráfica**, sendo este um dos últimos estágios do ***pipeline gráfico de renderização***, responsável pela rasterização das primitivas no espaço de tela, que significa a descrição dos modelos através de **coordenadas inteiras** definidas sobre o **espaço bidimensional discretizado da tela**. Acontece que esta forma de descrever primitivas, **não** é prática, pois, em vários casos, os modelos geométricos são tridimensionais e se estendem para além dos limites definidos de tela. Visando facilitar o processo de descrição de modelos, opta-se por fazê-lo em um espaço mais adequado, tanto em termos de extensão quanto de dimensões, que é o ***espaço do objeto***. Após a descrição neste espaço (***espaço do objeto***), o modelo passa por **uma série de transformações** que tem como **resultado o seu mapeamento final** para o ***espaço de tela***. Tendo como resultado, a **imagem final** obtida através da ***rasterização das primitivas*** projetadas na tela.

Esta atividade, tem por finalidade a **implementação das transformações** que **levarão os vértices** descritos **originalmente no espaço do objeto** para **o espaço de tela**.

3. DESENVOLVIMENTO

Neste trabalho foram implementados todas as transformações do pipeline em forma de Matriz e utilizando coordenadas homogêneas.

A seguir, estão enumeradas as etapas normalmente encontradas ao longo de um ***pipeline gráfico***, conforme mostra a **Figura 01**, que exemplifica as diversas fases encontradas em um ***pipeline gráfico***:

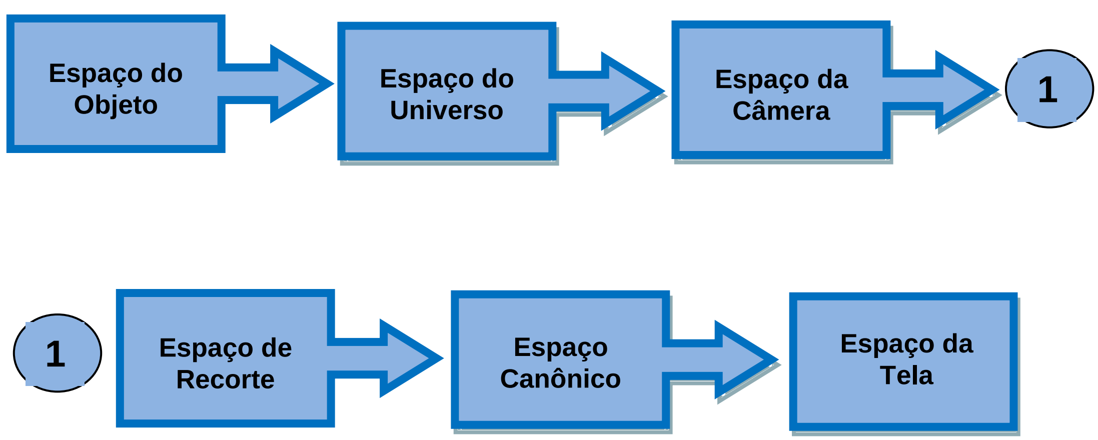

Figura 01 – Pipeline Gráfico de Renderização - Etapas

1 - Continuação do Pipeline Gráfico

4. TRANSFORMAÇÕES DE ESPAÇOS

4.1 – Espaço do Objeto para Espaço do Universo

É no Espaço do Objeto onde cada objeto é inicialmente criado. Cada objeto tem seu espaço individual, caracterizado pelo seu sistema de coordenadas. A **Figura 02**, a seguir, ilustra esta transformação.

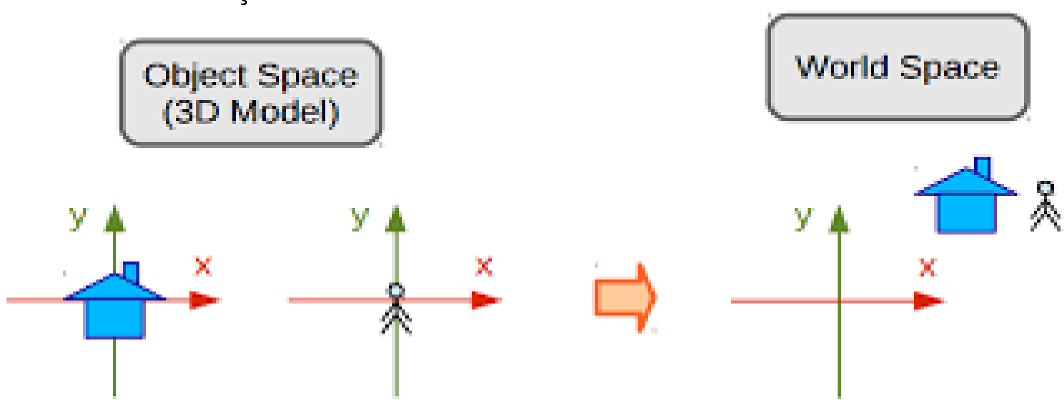

Figura 02 – Espaço do Objeto para Espaço do Universo

A transformação neste espaço visa transferir os objetos deste estado para o **Estado do Universo**. É neste novo espaço que os objetos advindos do espaço do objeto, passam por algumas transformações, a saber:

Transformações Geométricas de:

1-) ***Escala:*** É a mudança de medida de uma ou mais dimensões; essas mudanças são feitas multiplicando as coordenadas por um fator de escala, um valor constante. Esta transformação consiste na ***alteração das dimensões do objeto como: largura, altura*** *e etc.* A **Figura 03**, mostra a matriz utilizada para transformação em escala.

$$\begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & Sz \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Figura 03 – Matriz utilizada na Transformação de Escala

.Onde:

$x' = x*S_x$	$y' = y*S_y$	$z' = z*S_z$

Tipos de Escalas:

Escala Isotrópica: Uma escala é dita isotrópica, quando os fatores de escalas são iguais, ou seja: **Sx = Sy = Sz**.

A **Figura 04**, mostra o tipo de **Escala Isotrópica** nas transformações em escala.

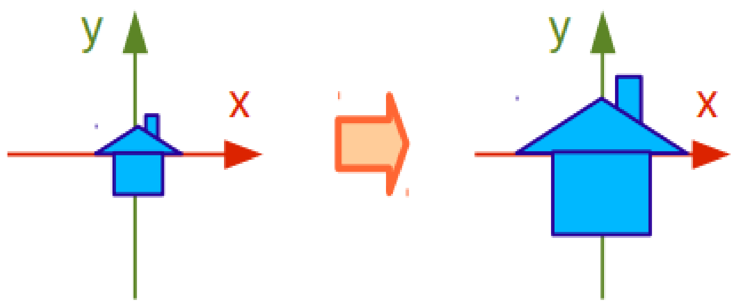

Figura 04 – Escala Isotrópica na Transformação em Escala

Escala Anisotrópica: Uma escala é dita anisotrópica, quando os fatores de escalas são diferentes, ou seja, **Sx ≠ Sy ≠ Sz**.

A **Figura 05**, mostra o tipo de **Escala Anisotrópica** nas transformações em escala.

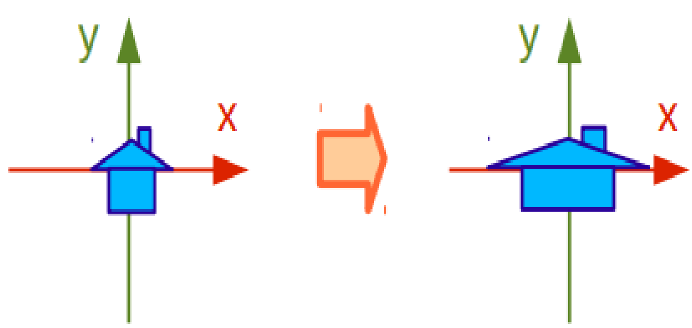

Figura 05 – Escala Anisotrópica na Transformação em Escala

Escala Espalhamento: Uma escala é dita de espelhamento quando o objeto tem **um ou mais eixos invertidos**.

A **Figura 06**, mostra o tipo de **Escala de Espelhamento** nas transformações em escala.

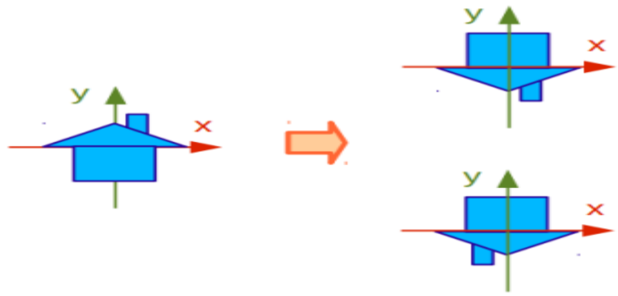

Figura 06 – Escala de Espelhamento na Transformação em Escala

2-) ***Rotação:*** É um movimento giratório em torno de um eixo referencial, nesse caso o objeto pode rotacionar em torno de cada um dos eixos sendo necessário uma matriz para cada tipo de rotação.

A **Figura 07**, a seguir, ilustra como seria a **rotação** em **relação aos 03(três) eixos**.

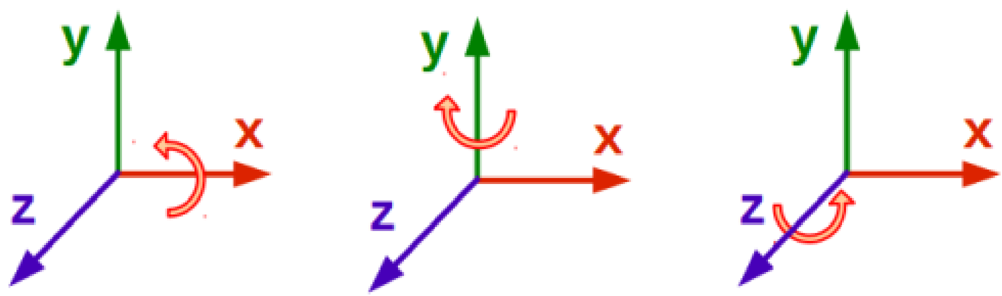

Figura 07 – Rotações em Relação aos Eixos X Y e Z

Observação: Para que a rotação seja aplicada corretamente é necessário que o objeto esteja na origem do sistema de coordenadas, caso contrário, o objeta irá transladar. A **Figura 08**, ilustra esta situação.

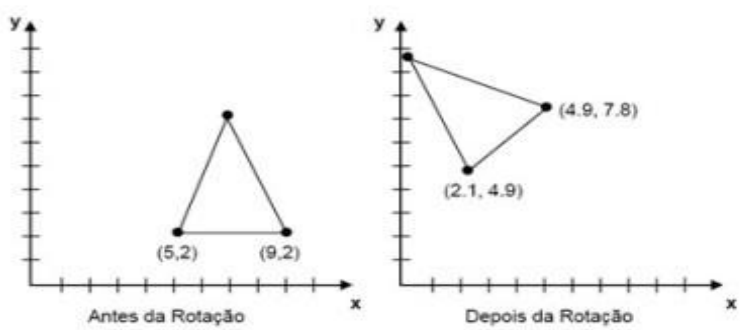

Figura 08 – Rotação quando o objeto não está na origem

As **Figuras 09, 10 e 11**, a seguir, ilustram como seriam as matrizes aplicadas na **rotação** em **relação a cada um dos 03(três) eixos**.

Matrizes de Rotação

Rotação em X:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -\sin\theta \\ 0 & \sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Figura 09 – Matrizes de Rotação em Relação ao eixo X

Rotação em Y:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & cos\theta \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Figura 10 – Matrizes de Rotação em Relação aos eixo Y

Rotação em Z:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Figura 11 – Matrizes de Rotação em Relação aos eixo Z

3-) Translação: É a movimentação do objeto ao longo do sistema de coordenadas, para fazer isso basta somar valores Tx, Ty e Tz as coordenadas do objeto podendo ser descrito da seguinte forma:

$$x' = x + Tx$$

$$y' = y + Ty$$

$$z' = z + Tz$$

A **Figura 12** mostra a transformação de translação.

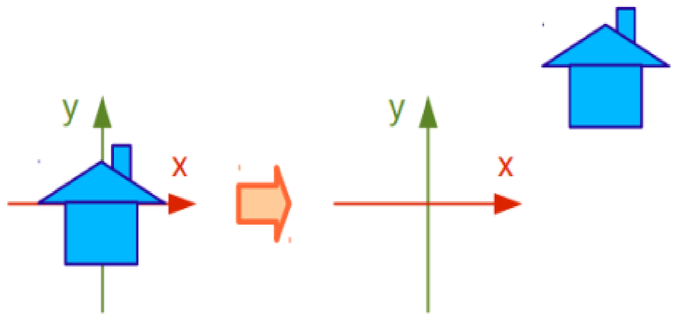

Figura 12 – Transformação de Translação

Como se pode notar a **translação** diferente das outras, **<u>não</u> pode ser expressa em uma matriz** e isso gera um problema, uma vez que, quando forem feitas as transformações no objeto, **as translações ficariam a parte**, isso é muito custoso. Para **resolver esse problema** se utiliza **as coordenadas homogêneas**.

4.1.1 - COORDENADAS HOMOGÊNEAS

A utilização de matrizes é importante, pois permite unir todas as transformações em uma única matriz, porém como foi visto anteriormente a **translação <u>não</u>** pode ser descrita como **uma expressão matricial** para resolver esse problema **utilizamos da coordenada homogênea**. Ao utilizá-la **teremos que adicionar uma dimensão a mais a todas as matrizes de transformação**. Isso significa que os vetores que antes eram expressos em **(x,y,z)**, agora será expresso da seguinte forma: **(x,y,z,w)**.

Para **sair do espaço euclidiano** para **o espaço homogêneo** basta **multiplicar x, y** e **z** pela **coordenada homogênea** e **acrescentar uma dimensão a mais** que **é a própria coordenada homogênea**, isto é:

(x, y, z) => (xw, yw, zw, w), para w = 1 => (x,y,z,1)

Para **retornar do espaço homogêneo** para o **espaço euclidiano** basta **dividir x, y, z** por **w** e **retirar a coordenada homogênea**, isto é:

(x, y, z, w) => (x/w, y/w, z/w)

Feito isso, podemos reescrever **as matrizes de escala e rotação** e **formar a matriz de translação**.

Escala:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 13 – Matriz utilizada na Transformação de Escala

Rotação em X:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\Theta) & -sin(\Theta) & 0 \\ 0 & sin(\Theta) & cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 14 – Matrizes de Rotação em Relação ao eixo X

Rotação em Y:

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

Figura 15 – Matrizes de Rotação em Relação ao eixo Y

Rotação em Z:

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

Figura 16 – Matrizes de Rotação em Relação ao eixo Z

Translação:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 17 – Matrizes de Translação

4.2 – Espaço do Universo para Espaço da Câmera

Da mesma forma que acontece com os objetos, a câmera (que é o observador) necessita também, ter seus sistemas de coordenadas configurados. A câmera é definida através da configuração dos seguintes parâmetros:

1. **Sua posição;**
2. **O seu vetor de direção;**
3. **Seu vetor de UP(cima).**

Os valores dos parâmetros supracitados serão necessários para formar as bases do espaço da câmera.

A **Figura 18**, a seguir, ilustra este tipo de transformação.

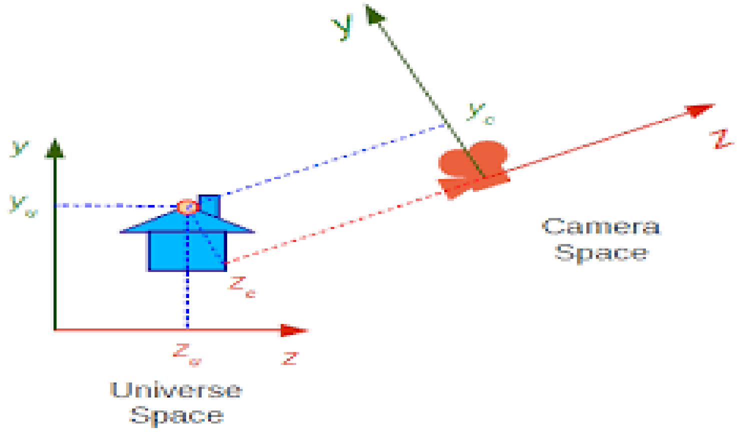

Figura 18 – Espaço do Universo para Espaço da Câmera

Quando uma câmera é adicionada à cena, os **pontos do espaço do universo** irão passar para um **novo sistema de coordenadas**. Esse sistema será construído por meio das informações supracitadas referentes á câmera, que refere-se de como a câmera foi posicionada na cena, ou seja, a ***posição***, a ***direção*** para onde **a câmera aponta** e o **vetor *UP***, que **informa para onde a cabeça da câmera aponta**. Esses valores serão necessários para formar as bases do espaço da câmera.

Para se obter os valores de configuração da câmera, deve-se começar pelo **cálculo do Eixo Z**, pois **a câmera aponta para a mesma direção do Eixo Z**, porém com **sentido oposto**. Logo, o **calculo do valor do Eixo Z** é obtido bastando para isso normalizar (**base ortonormal**) o **vetor direção** e **inverter o sentido**.

O **cálculo do Eixo X** se dá por meio do **produto vetorial entre o vetor *up*** e o **novo valor do Eixo Z**, tendo em vista que o **Eixo X é perpendicular** a **estes outros dois vetores**. O **resultado do produto vetorial**, **normalizado**, será o **valor do eixo X**.

De forma semelhante, para o **cálculo do Eixo Y**, basta **aplicar uma multiplicação vetorial entre os Eixos X e Z**, seguida de **uma normalização do vetor resultante**.

De posse dos valores de parâmetros da câmera supracitados, pode-se criar a **Matriz *View***, que **transformará pontos no espaço do universo**, para o **espaço de câmera**, após a multiplicação. A **matriz *view*** é **composta por uma rotação** (para alinhar **as bases do espaço do universo** com **as bases do espaço de câmera**) e **uma**

translação (para **igualar as origens**). A **Figura 19**, a seguir, mostra a **Matriz View** de **Transformação do Espaço do Universo** para o **Espaço da Câmera**.

$$\mathbf{B}^{\mathrm{T}} = \begin{bmatrix} x_{cx} & x_{cy} & x_{cz} & 0 \\ y_{cx} & y_{cy} & y_{cz} & 0 \\ z_{cx} & z_{cy} & z_{cz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boxed{\mathbf{M}_{view} = \mathbf{B}^{\mathrm{T}}\mathbf{T}}$$

Figura 19 – Matriz View de Transformação do Espaço do Universo para Espaço da Câmera

A primeira, segunda e terceira linha da **matriz transposta de B** (**B^T**) serão preenchidas respectivamente, pelos **valores encontrados** para os **eixos X, Y** e **Z** (**Xcam, Ycam** e **Zcam**).

O **vetor posição da câmera** é representado por: **P(Px, Py, Pz, 1).**

A **Figura 20**, a seguir, ilustra as etapas de transformações ocorridas no ***pipeline gráfico*** deste o **espaço do objeto** até o **espação da câmera**:

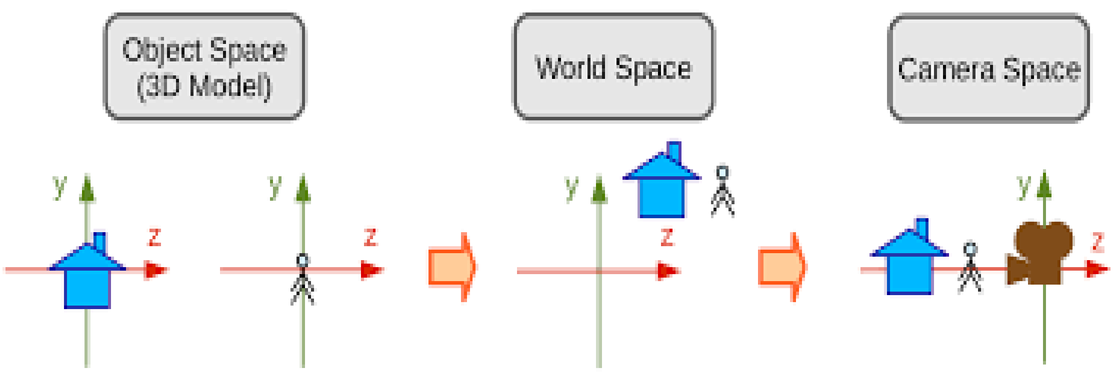

Figura 20 – Espaço do Objeto - Espaço do Universo - Espaço da Câmera

4.3 – Espaço da Câmera para Espaço de Recorte(Clipping Space)

De posse dos pontos no espaço da câmera, o próximo passo do pipeline gráfico é projetar esses pontos em um **plano de projeção (*view plane*)**, para em seguida, passar esses pontos, pelo procedimento de recorte. Ao realizar isso, podemos escolher como queremos que tais primitivas sejam projetadas: com **projeção em perspectiva** ou **projeção em paralelo**.

4.3.1 - Projeção Ortogonal

Na **projeção ortogonal**, a **forma e tamanho da imagem original são mantidos após serem projetadas** no ***view plane***. A **projeção ortogonal** se caracteriza pela **desconsideração da coordenada Z**, **transformando um objeto tridimensional em um objeto bidimensional**.

Sendo assim, a **matriz de uma projeção ortogonal** é simplesmente **a matriz identidade**, tendo em vista que nesse tipo de projeção, **<u>não</u>** há alteração do valor da coordenada homogênea, nem nas demais coordenadas. A **Figura 21**, a seguir, mostra um exemplo de **Projeção Ortogonal**.

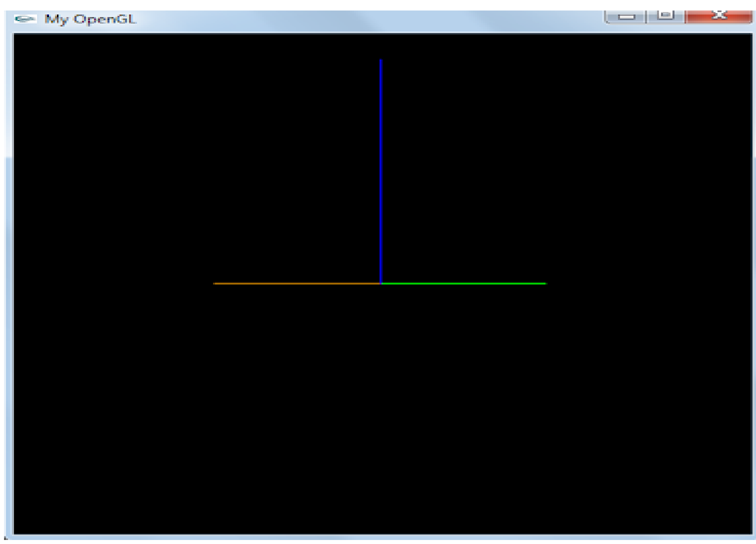

Figura 21 – Exemplo de uma Projeção Ortogonal
Eixo X (cor verde), Eixo Y (cor azul) e
Eixo Z (laranja) projetados sobre o *view plane* de forma ortogonal.

4.3.2 - Projeção Perspectiva

Neste **tipo de projeção**, o **tamanho e a forma de um objeto** são modificados de acordo com a **sua posição do espaço do universo**, dando **a sensação de distancia** ou **proximidade** quando **projetado na tela**.

Para fazer a projeção, **precisamos trabalhar com as coordenadas homogêneas**, o que **não** acontecia com **a projeção ortogonal**. Sendo assim **altera-se o valor de *w***. Esse **é o único momento do pipeline** em que a **coordenada homogênea** pode **ter valor diferente de 1**. A matriz que altera o valor da **coordenada *w*** está representada pela variável **perspectiva.**

Após isso, todas as outras quatro coordenadas do ponto serão divididas pelo novo valor de *w*, ou seja, passaram pelo processo de homogeneização. Esse processo é feito para que ocorra uma distorção na imagem, comprimindo toda a cena em um cubo, que quando diminuído as dimensões, irá representar o espaço canônico.

Por fim, o ponto homogeneizado será multiplicado pela matriz de projeção, como é mostrado na **Figura 22**, abaixo:

$$\mathbf{B}^{\mathrm{T}} = \begin{bmatrix} x_{cx} & x_{cy} & x_{cz} & 0 \\ y_{cx} & y_{cy} & y_{cz} & 0 \\ z_{cx} & z_{cy} & z_{cz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boxed{\mathbf{M}_{view} = \mathbf{B}^{\mathrm{T}}\mathbf{T}}$$

Figura 22 – Exemplo de uma Projeção Ortogonal

Multiplicando **os pontos do espaço do universo** pela **matriz *view,*** teremos os mesmos pontos representados agora no **espaço de recorte** ou **projeção**.

4.4 – Espaço de Recorte para o Espação Canônico

Após a aplicação da projeção perspectiva devemos homogeneizar os vértices dividindo eles pela **coordenada w**, isto irá provocar uma mudança na cena fazendo com que os objetos próximos da câmera fiquem maiores e os mais afastados fiquem menores, tudo isso em um espaço que varia de -1 até 1. A **Figura 22**, abaixo, ilustra esta situação:

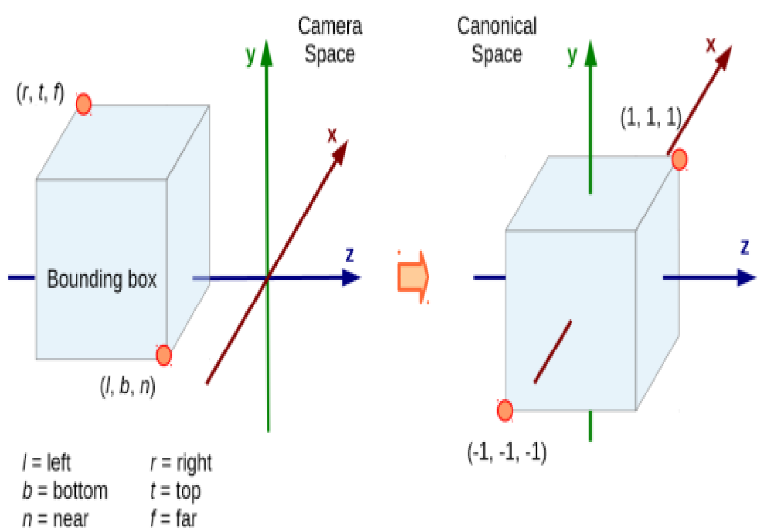

Figura 23 – Espaço de Recorte para o Espaço Canônico

4.5 – Espaço Canônico para Espaço de Tela

No espaço canônico é garantido que todos os vértices da cena visível possuam os valores de suas coordenadas entre -1 e 1. Isto acontece depois da homogeneização dos vértices, conforme mostrado anteriormente. Em seguida se faz necessário preparar os vértices para **serem rasteirados na tela**. Este processo é feito multiplicando os vértices por **uma matriz** chamada **viewport**. Essa matriz **leva os vértices do espaço canônico** para o **espaço da tela** e é formada pela multiplicação das matrizes mostradas na **Figura 24**, abaixo:

$$\underbrace{\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}}_{\text{Point } \mathbf{p'} \text{ in Screen space.}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \frac{w-1}{2} \\ 0 & 1 & 0 & \frac{h-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Transtale.}} \underbrace{\begin{bmatrix} \frac{w}{2} & 0 & 0 & 0 \\ 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Scale along X and Y acis.}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Invert Y axis direction}} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\text{Point } \mathbf{p} \text{ in canonical space}}$$

Figura 24 – Multiplicação de Matrizes – Vértices do Espaço Canônico para Espaço de Tela

A **Figura 25**, a seguir, ilustra a etapa final do ***pipeline gráfico***.

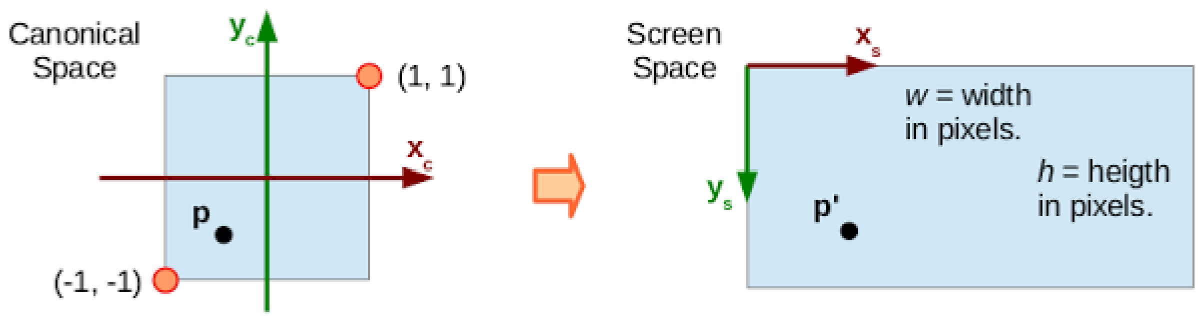

Figura 25 – Espaço Canônico para Espaço de Tela

5. RESULTADOS DAS IMPLEMENTAÇÕES - CÓDIGOS

Neste projeto foi solicitado a implementação das transformações que levam vértices descritos originalmente no espaço do objeto para o espaço de tela. Para esta implementação foi fornecido pelo professor da disciplina um **arquivo do tipo .OBJ**, contendo os dados de vértices de um objeto, a saber, o ***arquivo "monkey_head2.obj*"**. Foi fornecida, também, uma biblioteca simples para carga de modelos descritos em **.OBJ (Wavefront .obj)**.

De posse destes arquivos, foi desenvolvido na **IDE do Code::Blocks,** utilizando a **biblioteca OpenGL** e suas **bibliotecas auxiliares**, um sistema capaz de ler os dados do arquivo fornecido e, através dos processos de renderização do ***Pipeline Gráfico,*** exibir o resultado na tela do computador.

O resultado obtido na renderização pode ser verificado na **Figura 26**, a seguir:

Objeto: Monkey Head (Cabeça de Macaco)

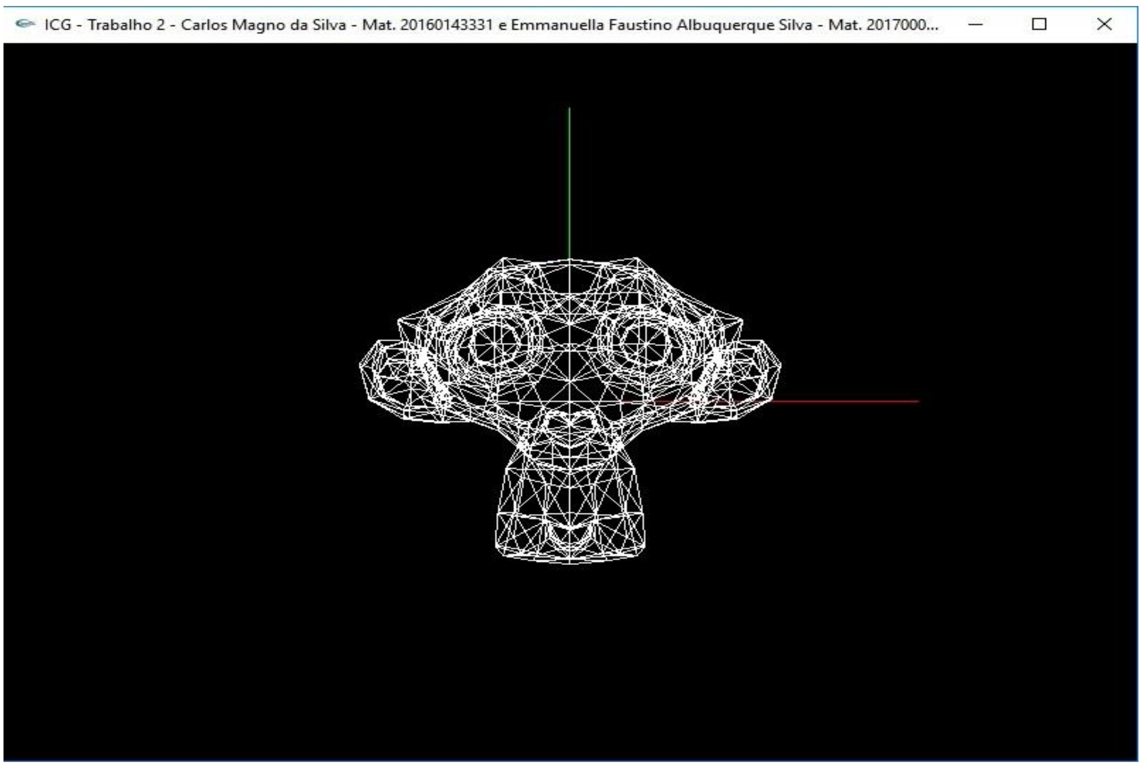

Figura 26 – Imagem – Monkey Head (Cabeça de Macaco)

6. ATIVIDADE EXTRA

Embora neste projeto tenha sido solicitado, apenas, a renderização através do ***Pipeline Gráfico*** do objeto, referente ao **arquivo .OBJ (*arquivo "monkey_head2.obj*"),** fornecido pelo professor; o sistema implementado neste projeto foi utilizado, também, para leitura de **arquivos .OBJ** e **renderização** de **outros objetos**, visando verificar a consistência de seu funcionamento..

Para isto, no sistema, foi implementado e incluído um **módulo de Menus Popup's**, na tela principal, conforme mostra a **Figura 27,** de modo a organizar o processamento dos diversos objetos renderizados e exibidos na tela do computador. Os **arquivos .OBJ's** dos diversos objetos foram obtidos através de downloads de um site na internet, cujo **link** está referenciado no final deste projeto.

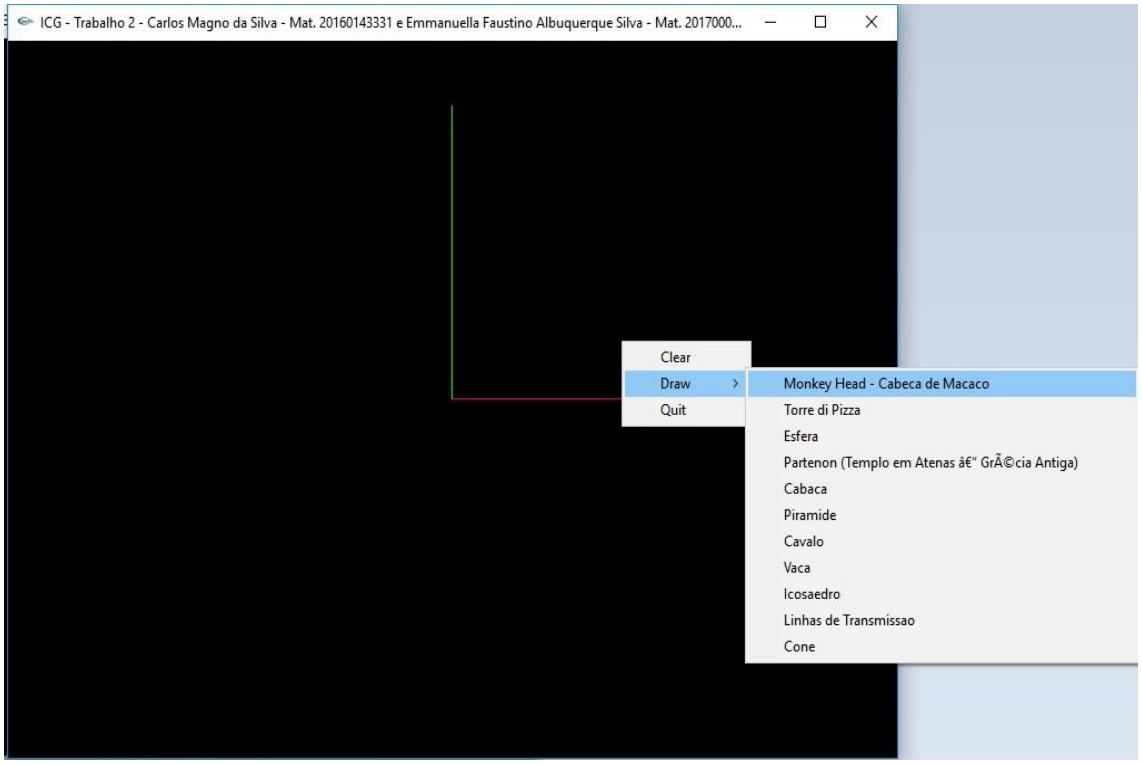

Figura 27 – Tela Inicial – Menu de Opções do Sistema

7. RESULTADOS DE RENDERIZAÇÃO DE OUTROS OBJETOS

Os resultados obtidos são os mostrados nas figuras seguintes.

Objeto: TorreDiPizza (Torre de Pizza)

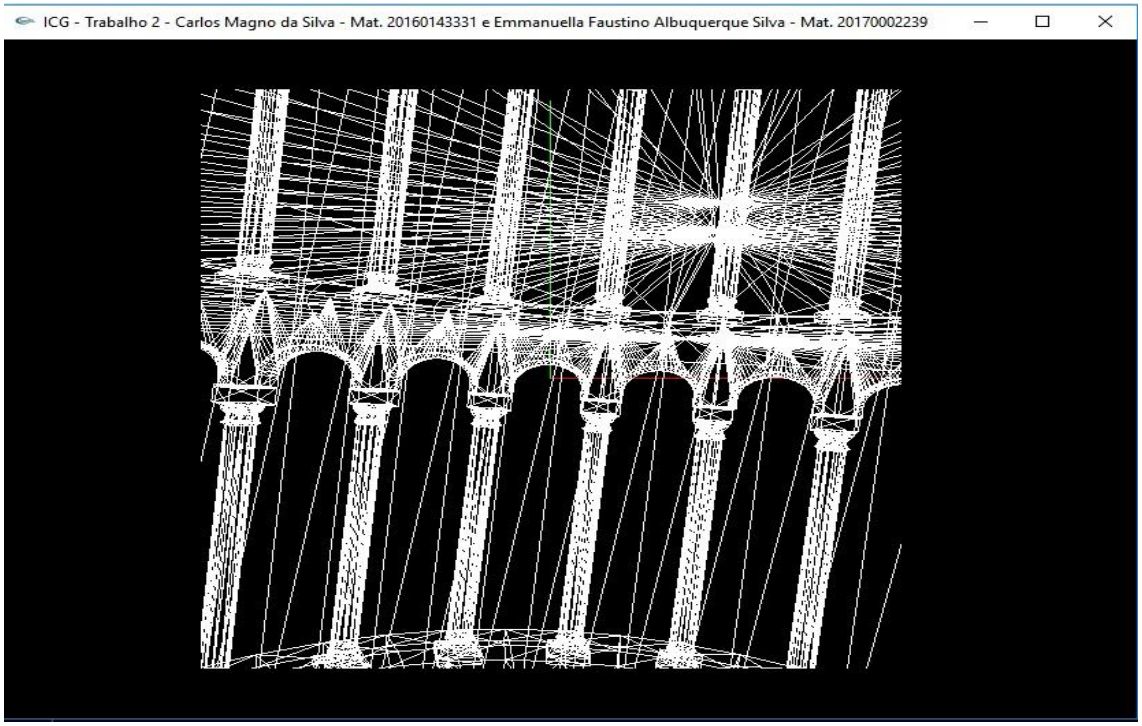

Figura 28 – Imagem – TorreDiPizza (Torre de Pizza)

Objeto: Sphere (Esfera)

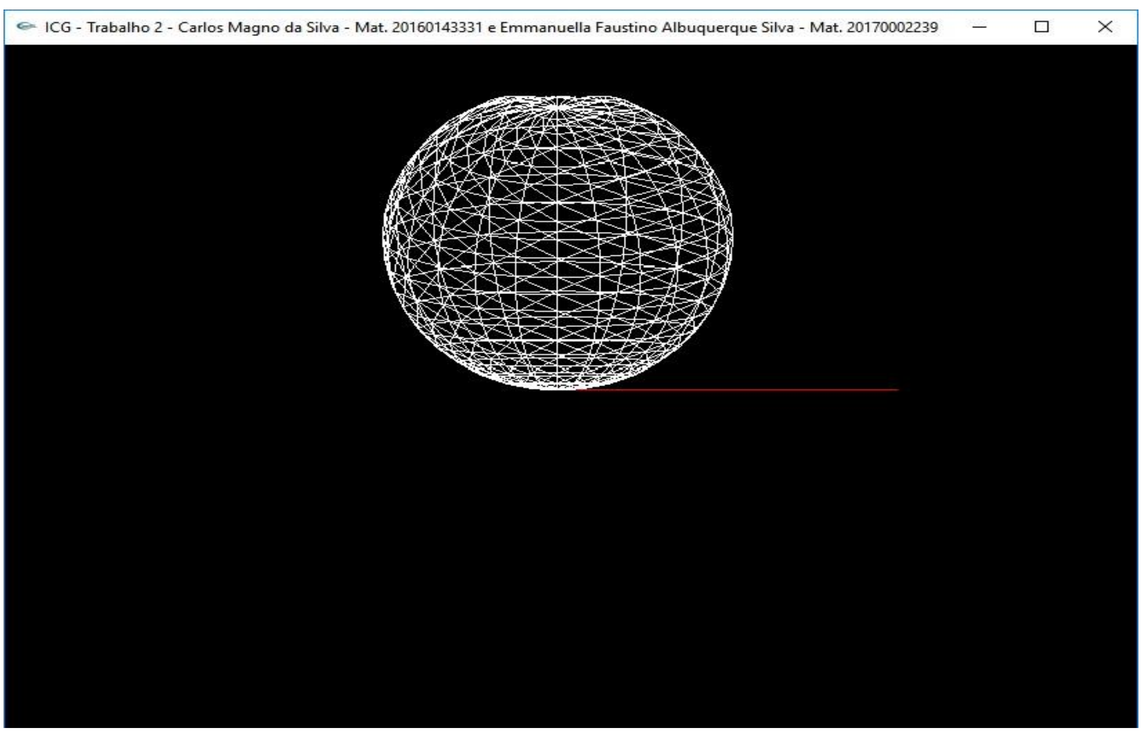

Figura 29 – Imagem – Esfera

Objeto: Partenon (Templo em Atenas – Grécia Antiga)

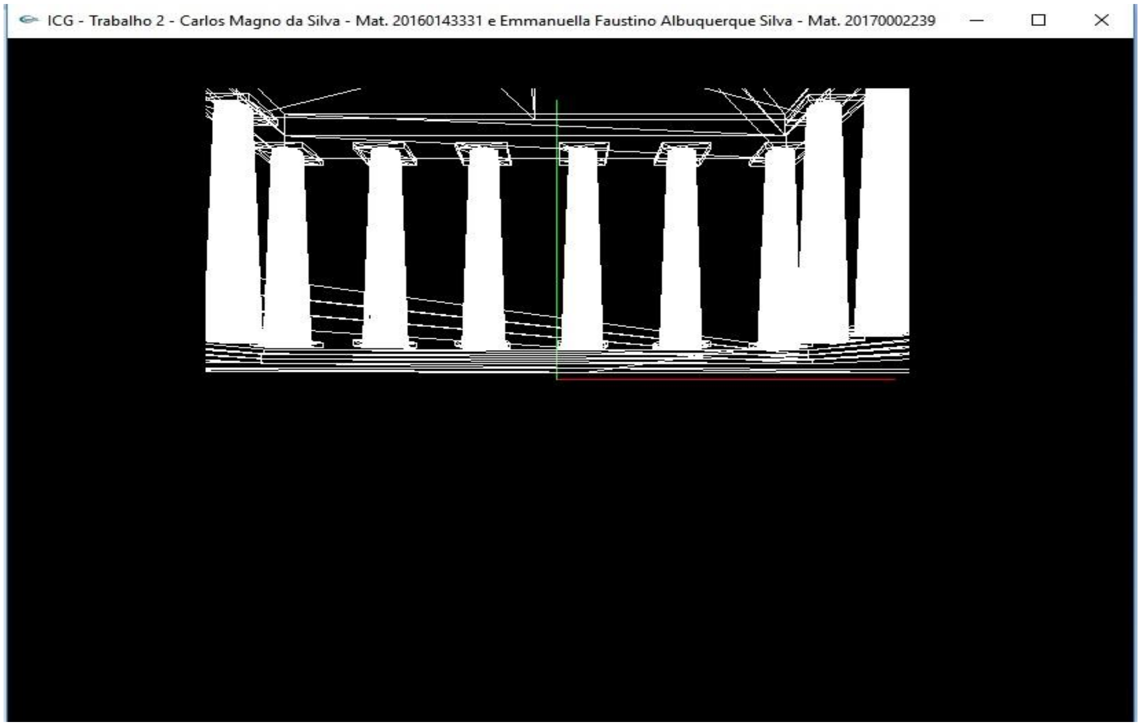

Figura 30 – Imagem – Partenon - Templo em Atenas - Grécia

Objeto: Gourd (cabaça)

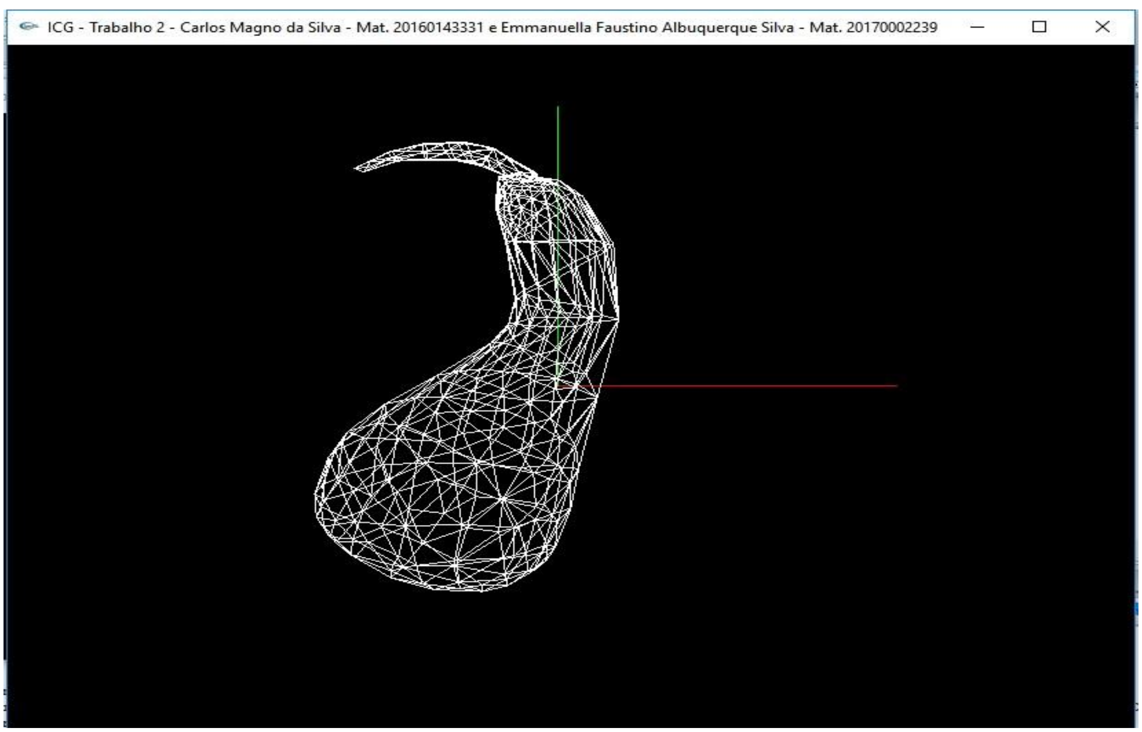

Figura 31 – Imagem - Cabaça

Objeto: Pyramid (Pirâmide)

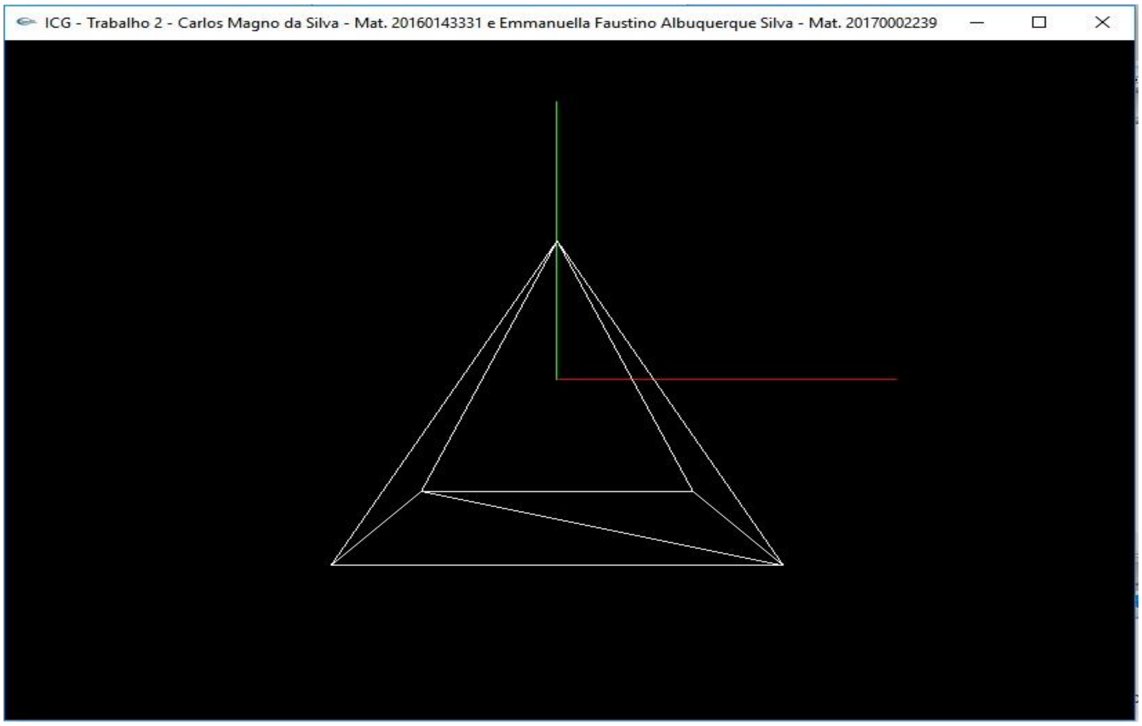

Figura 32 – Imagem – Pirâmide

Objeto: Horse (Cavalo)

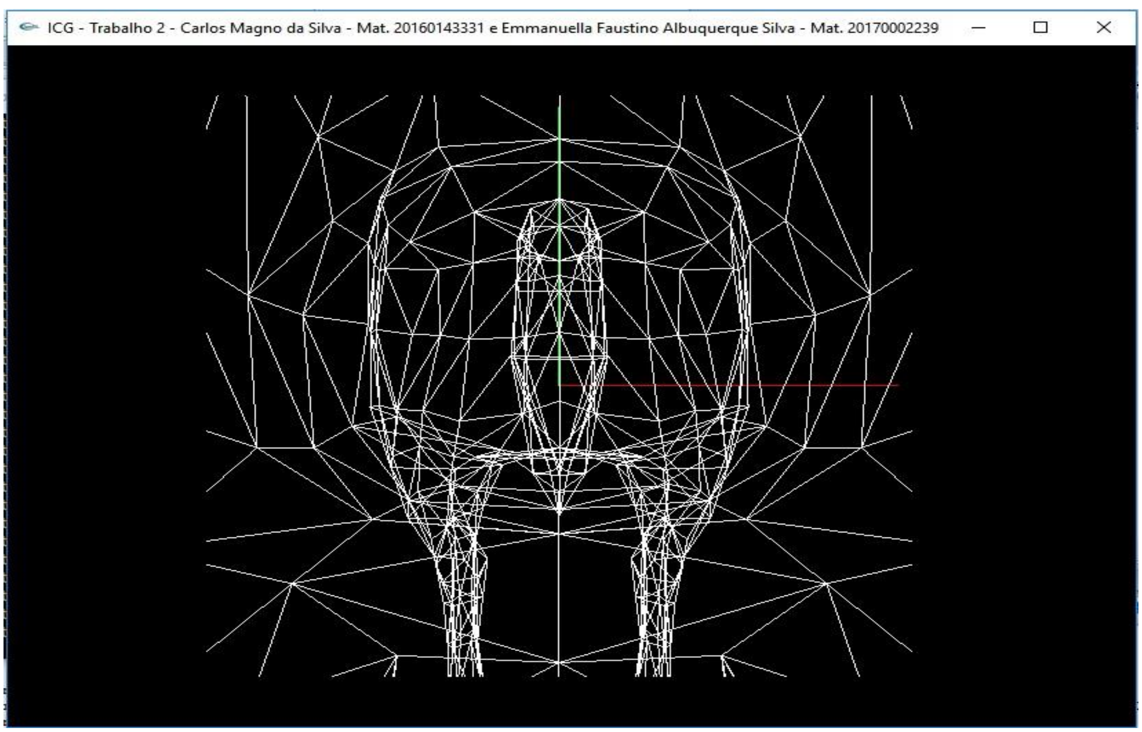

Figura 33 – Imagem – Cavalo

Objeto: Cow (Vaca)

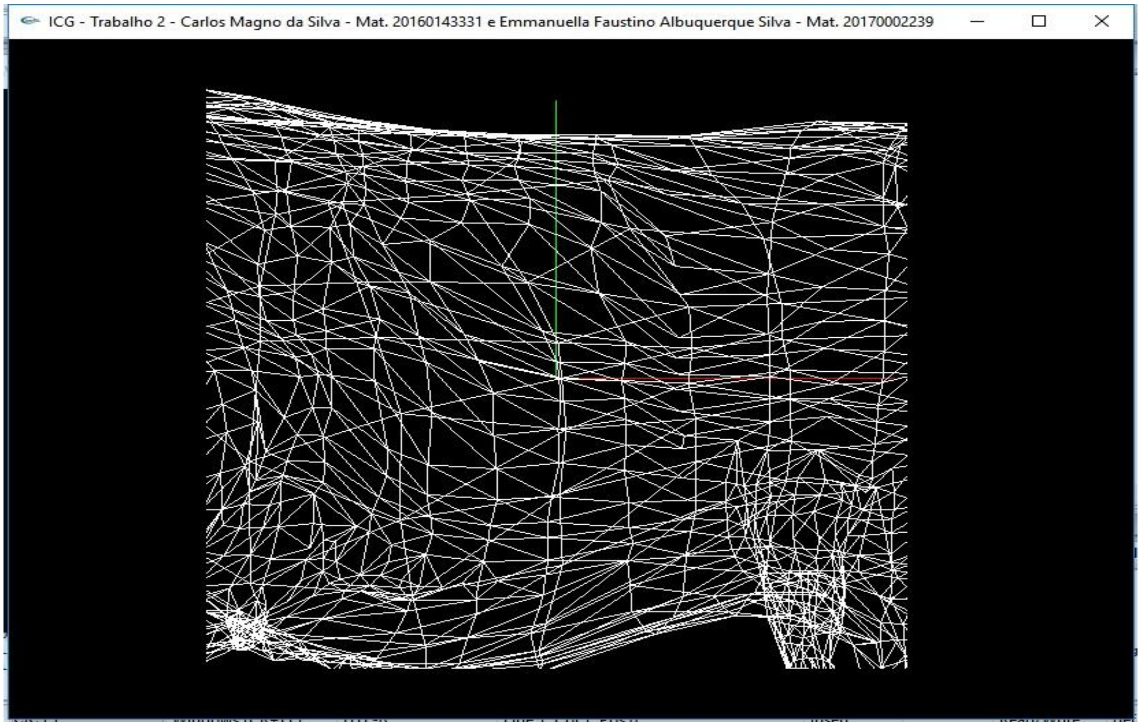

Figura 34 – Imagem – Vaca

Objeto: icosahedron (icosaedro)

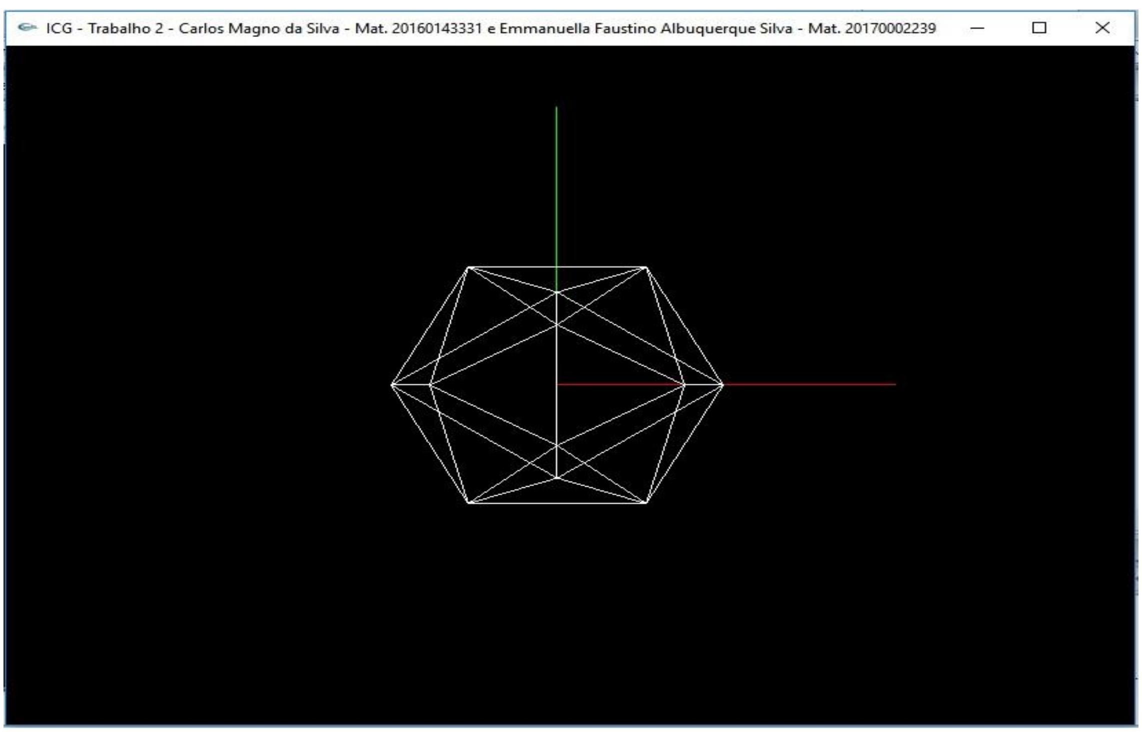

Figura 35 – Imagem – icosaedro

Objeto: Power_lines (linhas de Energia)

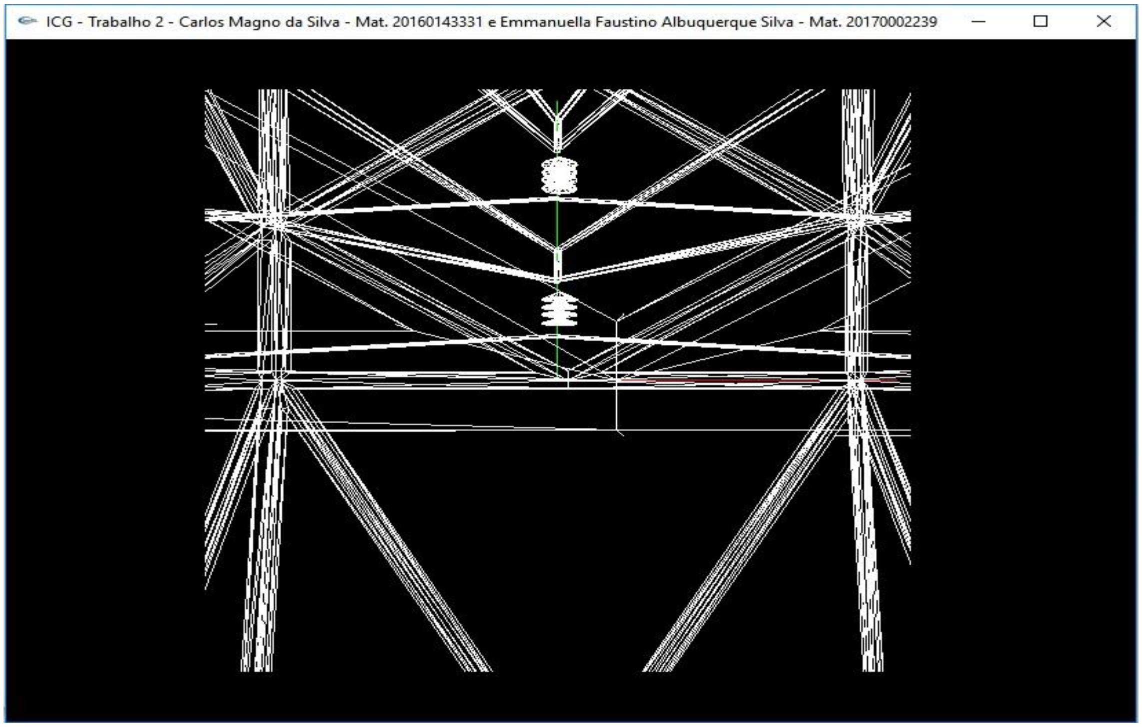

Figura 36 – Imagem - Linhas de Transmissão

8. REFERÊNCIAS

1 – Tutorial de Utilização de OpenGl – Marcionílio Barbosa Sobrinho – Belo Horizonte – MG – 2003

2 – Introdução à OpenGL - Professora Isabel Harb Manssour – https://www.inf.pucrs.br/~manssour/OpenGL/Desenhando.html. Acesso em: 18/04/2019

3 – Downloads de Imagens utilizadas no Projeto – Arquivos do Tipo OBJ - Retiradas do seguinte Site: OBJ Files - A 3D Object Format https://people.sc.fsu.edu/~jburkardt/data/obj/obj.html. Acesso em: 18/04/2019

Links:

https://www.google.com/search?tbm=isch&q=transforma%C3%A7%C3%B5es+geom%C3%A9tricas+no+Pipeline+Gr%C3%A1fico&chips=q:transforma%C3%A7%C3%B5es+geom%C3%A9tricas+no+pipeline+gr%C3%A1fico,online_chips:opengl&sa=X&ved=0ahUKEwj0mp7NlPHaAhWGipAKHSclAlAQ4lYIJigA&biw=1034&bih=615&dpr=1. Acesso em: 18/04/2019.