Modelagem dos objetos, sistema de coordenadas do mundo, da câmera e projeção em duas dimensões

Equipe: Mateus Machado Costa e
João Carlos Nepomuceno Fernandes

Disciplina: Computação Gráfica

Semestre: 2023.1

Professor: Ajalmar Rego da Rocha Neto

13 de Junho de 2023

Sumário

1 - Introdução:

Este relatório apresenta o desenvolvimento e aplicação de técnicas de modelagem e transformações de sólidos em um sistema de coordenadas tridimensional utilizando a linguagem Python. O objetivo principal é criar representações em arame de diversos sólidos geométricos, como cilindro, cone, esfera, tronco de pirâmide e cubo, e posteriormente posicionar esses sólidos em um cenário coerente, evitando sobreposições e intersecções entre eles.

Na primeira parte do relatório, foram utilizadas funções para criar os sólidos a partir de parâmetros fornecidos. Cada sólido foi modelado de forma indireta, levando em consideração medidas como altura, raios das bases, número de fatias e circunferências intermediárias. Foram criadas funções que retornam a matriz de vértices e arestas de cada sólido, permitindo sua posterior manipulação no sistema de coordenadas tridimensional.

Em seguida, uma cena foi composta, onde os sólidos modelados anteriormente foram posicionados no sistema de coordenadas do mundo. Foram aplicadas transformações de escala, rotação e translação para evitar sobreposições e intersecções entre os sólidos. O cilindro e o cone foram posicionados em um octante específico, enquanto a esfera, o tronco de pirâmide e o cubo foram posicionados em outro octante.

Foi estabelecido um limite máximo para as componentes dos vértices no sistema de coordenadas do mundo, definindo-o como 10. Quando necessário, foram aplicadas transformações adicionais para garantir que os sólidos estivessem dentro desses limites.

Em seguida, foi selecionado um octante vazio e um ponto de origem para o sistema de coordenadas da câmera. A base vetorial (n, u e v) do novo sistema de coordenadas foi calculada considerando o volume de visão, usando o ponto médio entre os centros de massa de cada um dos sólidos. Os objetos no sistema de coordenadas do mundo foram então transformados para o sistema de coordenadas da câmera.

A origem do sistema de coordenadas do mundo foi mostrada como um ponto no sistema de coordenadas da câmera. Os diversos sólidos também foram apresentados nesse sistema de coordenadas tridimensionais.

Por fim, foi realizada uma transformação de projeção em perspectiva dos sólidos contidos no volume de visão. As arestas dos sólidos foram projetadas em duas dimensões na janela de projeção. Cada sólido recebeu uma cor única para suas arestas, permitindo uma melhor visualização e distinção entre os objetos.

2 - Desenvolvimento:

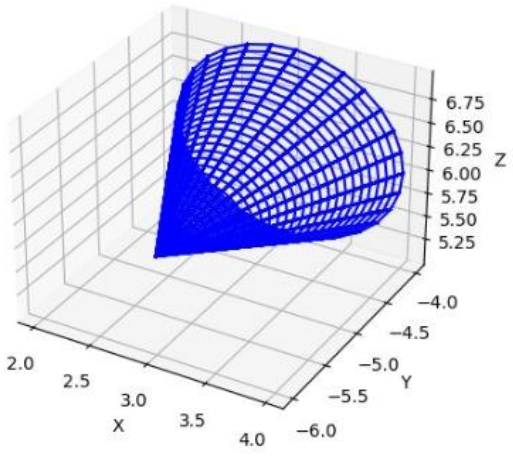
2.1 - Modelo em representação Aramada:

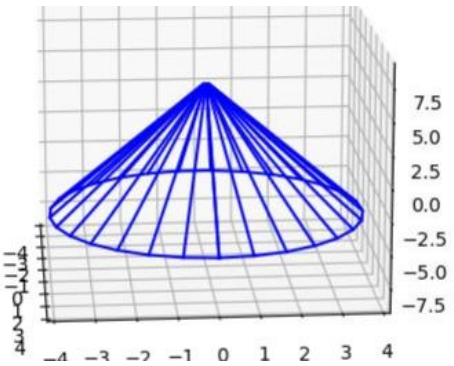
Nesta seção, apresentaremos a implementação dos sólidos/objetos solicitados e discutiremos as funções criadas para modelar cada um deles. Utilizamos uma representação aramada, onde os sólidos são definidos por meio de uma matriz de vértices e uma matriz de arestas.

I) Cone:

A lógica adotada para criar o cone consiste em definir os parâmetros do cone, como raio, altura e número de fatias. Em seguida, são geradas as coordenadas dos pontos que compõem o cone, levando em consideração esses parâmetros. As coordenadas x e y são calculadas em relação ao raio e à posição ao longo da circunferência, enquanto a coordenada z é distribuída uniformemente ao longo da altura do cone.

As coordenadas dos pontos são armazenadas em listas correspondentes aos eixos x, y e z. Além disso, são criadas as arestas que conectam os vértices para formar a estrutura do cone. A classe Cone também possui métodos para realizar transformações no cone, como rotação, deslocamento e escala. Por fim, o cone pode ser visualizado através do método “plota_cone”, que exibe o cone em um gráfico tridimensional.





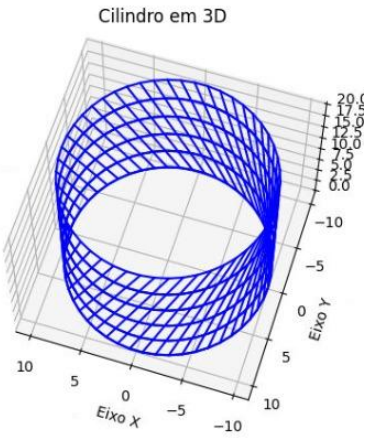
II) Cilindro:

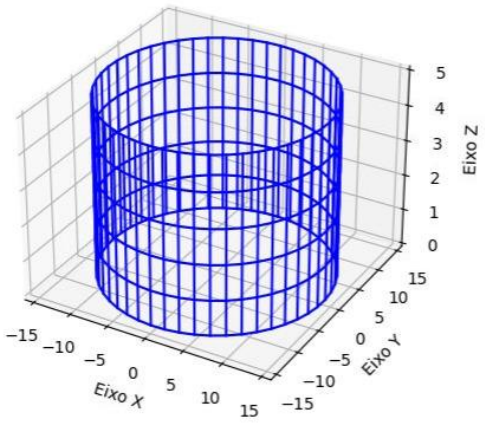
A lógica adotada nessa implementação segue uma abordagem que se baseia nos parâmetros essenciais do cilindro: raio, altura e ponto inicial. Para construir o cilindro, é utilizado um processo de geração dos pontos que compõem sua superfície. Esses pontos são determinados considerando uma resolução específica, o que permite controlar a suavidade e a precisão da representação do cilindro. Através de cálculos trigonométricos, as coordenadas dos pontos são calculadas com base no raio, na posição ao longo da circunferência e na altura do cilindro.

Os vértices do cilindro são então armazenados em listas separadas para cada eixo: x, y e z. Essa estrutura de dados facilita a manipulação posterior dos pontos e a definição das arestas que conectam os vértices. As arestas são estabelecidas entre vértices adjacentes, formando a estrutura sólida do cilindro. Essas conexões são armazenadas em uma lista de arestas, que permite uma representação adequada do objeto.

Além disso, a classe Cilindro oferece métodos adicionais para realizar transformações no cilindro, como escala, deslocamento e rotação. Essas transformações permitem modificar as coordenadas dos vértices de acordo com os parâmetros fornecidos, possibilitando a personalização do cilindro conforme a necessidade do usuário.

Dessa forma, a implementação do cilindro na classe Cilindro proporciona uma solução versátil e intuitiva para modelar e visualizar cilindros tridimensionais. Através dessa lógica, é possível criar cilindros com diferentes tamanhos, posições e orientações, ampliando as possibilidades de aplicação e explorando a representação gráfica desse sólido geométrico.





III) Cubo:

A lógica adotada nessa implementação baseia-se nos parâmetros essenciais do cubo: raio e ponto inicial.

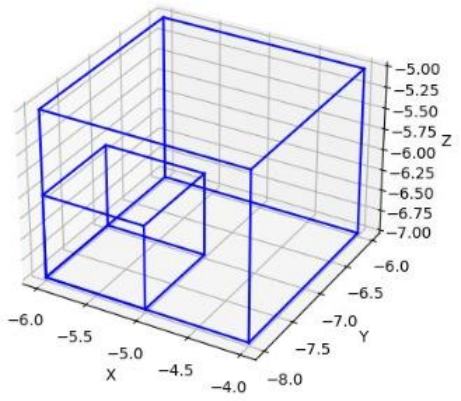
O processo de construção do cubo consiste em formar as bases e as arestas verticais do objeto. As bases são definidas como um quadrado formado pelos vértices do cubo, e as arestas verticais conectam os vértices das bases superiores e inferiores. Essa estrutura sólida é armazenada nos arrays x, y e z, que representam as coordenadas dos vértices.

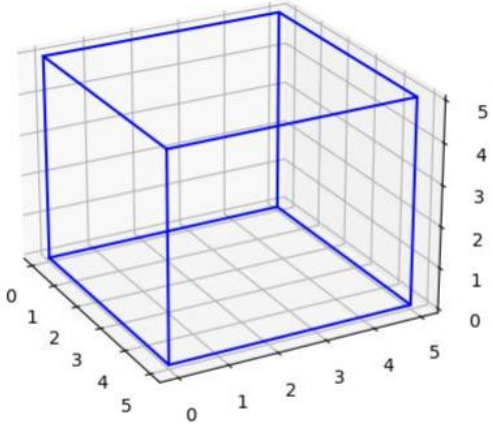
Os vértices das bases são calculados com base no ponto inicial e no raio do cubo. Através de manipulações simples, é possível obter as coordenadas dos vértices que compõem as bases. Esses vértices são adicionados aos arrays x, y e z, considerando a altura do cubo. Em seguida, as arestas que conectam os vértices das bases são formadas, permitindo uma representação adequada do objeto.

Além disso, a classe Cubo fornece métodos para realizar transformações no cubo, como rotação, deslocamento e escala. Essas transformações permitem modificar as coordenadas dos vértices de acordo com os parâmetros fornecidos, possibilitando a personalização do cubo de acordo com as necessidades do usuário.

Ao chamar o método “gera_cubo()”, todo o processo de geração das bases, formação das arestas e armazenamento dos vértices é realizado. Por fim, o cubo pode ser visualizado por meio do método “plota_cubo()”, que utiliza as coordenadas dos vértices e as arestas para renderizar o sólido tridimensional.

A implementação da classe Cubo oferece uma solução versátil e intuitiva para modelar e visualizar cubos tridimensionais. Através dessa lógica, é possível criar cubos com diferentes tamanhos, posições e orientações, ampliando as possibilidades de aplicação e explorando a representação gráfica desse sólido geométrico.



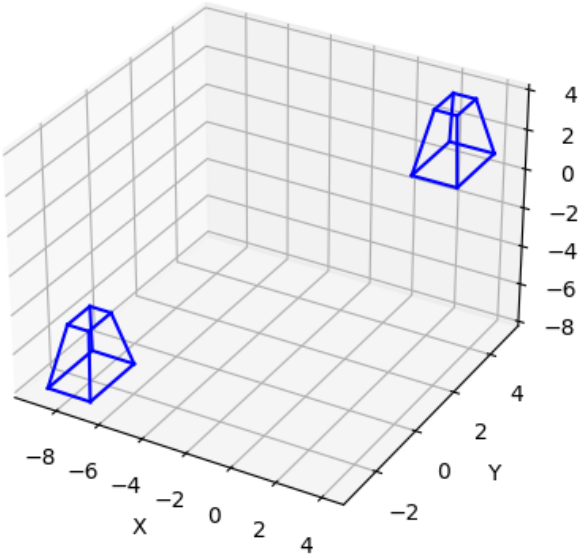


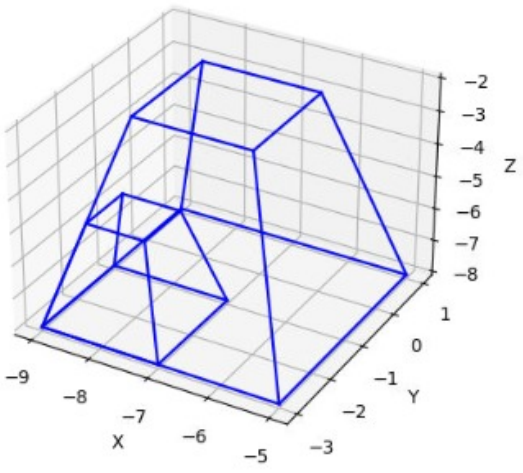
IV) Tronco de Pirâmide:

O processo de construção do tronco de pirâmide envolve a formação das bases e das arestas verticais. As bases são polígonos formados pelos vértices do tronco, enquanto as arestas verticais conectam os vértices das bases inferior e superior. Os vértices das bases são calculados com base nas dimensões fornecidas e no ponto inicial. As coordenadas x, y e z dos vértices são armazenadas em arrays. Após a formação dos vértices, as arestas que conectam os vértices das bases são definidas. Essas arestas são adicionadas a uma lista de arestas.

A classe “Tronco_piramide” também fornece métodos para aplicar transformações ao tronco, como deslocamento, rotação e escala. Ao chamar o método “gera_tronco()”, o tronco de pirâmide é gerado, formando as bases e as arestas. Em seguida, o método “plota_tronco()” pode ser usado para visualizar o objeto em um ambiente tridimensional.

Essa implementação oferece uma maneira conveniente e flexível de criar e visualizar troncos de pirâmides em três dimensões, permitindo ajustar as dimensões, a posição e a orientação do tronco de forma intuitiva.

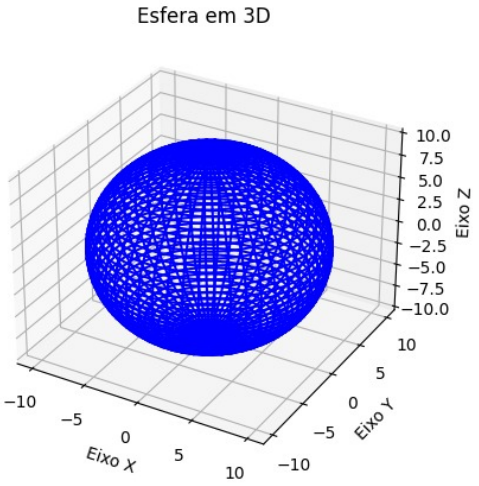


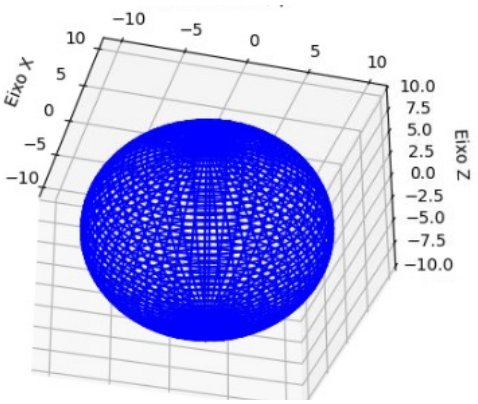


V) Esfera:

A geração da esfera é feita usando um número fixo de pontos distribuídos uniformemente em torno dos ângulos theta e phi. Com base nesses pontos, o raio e o ponto inicial fornecidos, são calculadas as coordenadas dos vértices da esfera.

As arestas da esfera são definidas conectando vértices adjacentes. Essas arestas são armazenadas em uma lista. Os métodos de transformação permitem alterar a escala, posição e orientação da esfera, ajustando as coordenadas dos vértices.Ao chamar o método “plota_esfera()”, a esfera é exibida em um ambiente tridimensional usando as coordenadas dos vértices e informações das arestas. Essa implementação oferece uma maneira conveniente de criar, modificar e visualizar esferas, permitindo ajustar facilmente seu tamanho, posição e orientação.





2.2 - Transformações em escala, rotação e translação:

I) Escala:

A escala é um processo que permite aumentar ou diminuir o tamanho de um objeto em relação aos eixos x, y e z. Para isso, são utilizados os fatores de escala sx, sy e sz, que determinam a proporção pela qual o objeto será escalado em cada um desses eixos.

No caso do cubo e do tronco de pirâmide, o processo de escala é aplicado após a geração das coordenadas dos vértices. Para cada vértice, as coordenadas originais (x, y, z) são multiplicadas pelos fatores de escala correspondentes sx, sy e sz. Dessa forma, as coordenadas são alteradas proporcionalmente em relação aos eixos, resultando em um cubo ou tronco de pirâmide escalado.

Já no caso da esfera, a escala é aplicada diretamente durante a geração das coordenadas dos vértices. Durante o processo de geração, para cada ponto da esfera, as coordenadas esféricas (r, theta, phi) são convertidas em coordenadas cartesianas (x, y, z). Nesse momento, os fatores de escala sx, sy e sz são aplicados diretamente às coordenadas (x, y, z). Assim, as coordenadas dos vértices da esfera são geradas considerando a escala desejada.

Em resumo, nos sólidos do tipo cubo e tronco de pirâmide, a escala é aplicada às coordenadas dos vértices após a geração das mesmas. Já na esfera, os fatores de escala são aplicados diretamente durante o processo de geração das coordenadas dos vértices. Essa abordagem permite obter os sólidos geométricos escalados, ajustando suas dimensões em relação aos eixos x, y e z de acordo com os fatores de escala especificados.

II) Rotação:

A rotação é um processo que permite girar um objeto em torno dos eixos x, y e z. Para realizar a rotação, são utilizados ângulos de rotação, geralmente denotados como” angle_x”, “angle_y” e” angle_z”, que determinam a quantidade de rotação em cada um dos eixos.Nos sólidos geométricos apresentados, a rotação é aplicada após a geração das coordenadas dos vértices.

Para realizar a rotação nos sólidos, são utilizadas as coordenadas cartesianas (x, y, z) dos vértices. A aplicação da rotação ocorre por meio da utilização de funções trigonométricas, como seno e cosseno, para calcular as novas coordenadas após a rotação. No caso do cubo e do tronco de pirâmide, a rotação é aplicada aos valores das coordenadas (x, y, z) de cada vértice. As coordenadas são atualizadas utilizando as fórmulas de rotação em torno dos eixos x, y e z, levando em consideração os ângulos de rotação especificados. Essas fórmulas permitem calcular as novas coordenadas após a rotação.

Para a esfera, o processo de rotação é similar. Durante a geração das coordenadas dos vértices, as coordenadas cartesianas (x, y, z) são calculadas a partir das coordenadas esféricas (r, theta, phi). Após a conversão para coordenadas cartesianas, a rotação é aplicada utilizando as mesmas fórmulas trigonométricas mencionadas anteriormente.

Em resumo, nos sólidos do tipo cubo, tronco de pirâmide e esfera, a rotação é aplicada às coordenadas dos vértices após a sua geração. As coordenadas são atualizadas utilizando fórmulas trigonométricas que levam em consideração os ângulos de rotação especificados. Dessa forma, é possível obter os sólidos geométricos com a rotação desejada em torno dos eixos x, y e z.

III) Translação:

A translação é um processo que permite mover um objeto geometricamente em um espaço tridimensional. Ela é aplicada alterando as coordenadas dos vértices do objeto para deslocá-lo em uma determinada direção e distância. Nos sólidos geométricos mencionados, a translação é realizada através do método de deslocamento dos vértices. Para aplicar a translação nos sólidos geométricos, são utilizados os parâmetros dx, dy e dz, que representam as distâncias de deslocamento nas direções x, y e z, respectivamente.

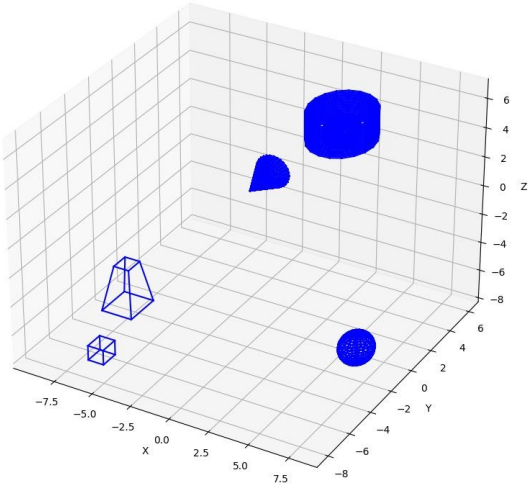
O processo de translação é simples. Cada coordenada (x, y, z) de cada vértice do sólido é atualizada somando-se as distâncias de deslocamento dx, dy e dz às coordenadas originais.Essa operação é realizada para todos os vértices do sólido, alterando suas coordenadas e movendo o sólido no espaço tridimensional.

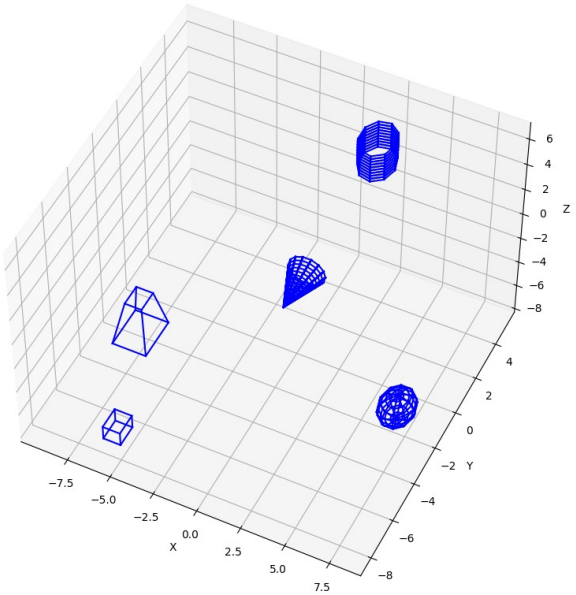
Em resumo, a translação é aplicada nos sólidos geométricos através da alteração das coordenadas dos vértices. As coordenadas de cada vértice são deslocadas somando-se as distâncias de deslocamento dx, dy e dz às suas coordenadas originais. Dessa forma, é possível mover os sólidos geométricos em uma determinada direção e distância no espaço tridimensional.

2.3 - Cena contendo os diversos sólidos:

No problema em questão, foi implementada uma divisão em octantes para lidar com diferentes regiões do espaço tridimensional. Essa divisão é uma abordagem comum em problemas que envolvem a representação e manipulação de objetos tridimensionais.

Em um sistema de coordenadas tridimensionais, os octantes são as oito partes em que o espaço é dividido. Cada octante representa uma região específica do espaço. Essa divisão permite tratar cada octante de forma distinta, aplicando as transformações necessárias para renderizar corretamente os objetos presentes em cada região. Cada octante possui características particulares, como o posicionamento relativo das coordenadas x, y e z.





2.4 - Sistema de Coordenadas da Câmera:

No contexto dessa questão, uma escolha foi feita para utilizar um octante diferente dos anteriores para a representação dos objetos do mundo. Essa decisão foi tomada com o intuito de explorar uma perspectiva única. Agora, o próximo passo consiste em calcular os vetores (n, u e v) a partir do ponto 'eye' no Sistema de Coordenadas da Câmera.

Para alcançar esse objetivo, é necessário realizar uma série de procedimentos. Primeiramente, é preciso calcular a média dos pontos de cada sólido de forma individual, considerando todos os objetos presentes. Essas médias individuais são então somadas e divididas pela quantidade total de objetos, resultando na média total.

A partir da média total, obtemos as coordenadas do ponto 'at', que representa o centro da região que desejamos observar. O próximo passo é determinar o vetor "n" através da subtração do vetor 'eye' do vetor 'at'. No entanto, ainda faltam dois vetores que devem ser ortogonais entre si.

Para obter o vetor "u", recorremos a uma projeção de um vetor auxiliar sobre o vetor "n", seguida da subtração dos vetores correspondentes. Isso nos dá um dos vetores ortogonais necessários. Por fim, para encontrar o vetor "v", recorremos ao produto vetorial entre "n" e "u", obtendo assim um vetor que é ortogonal tanto a "n" quanto a "u".

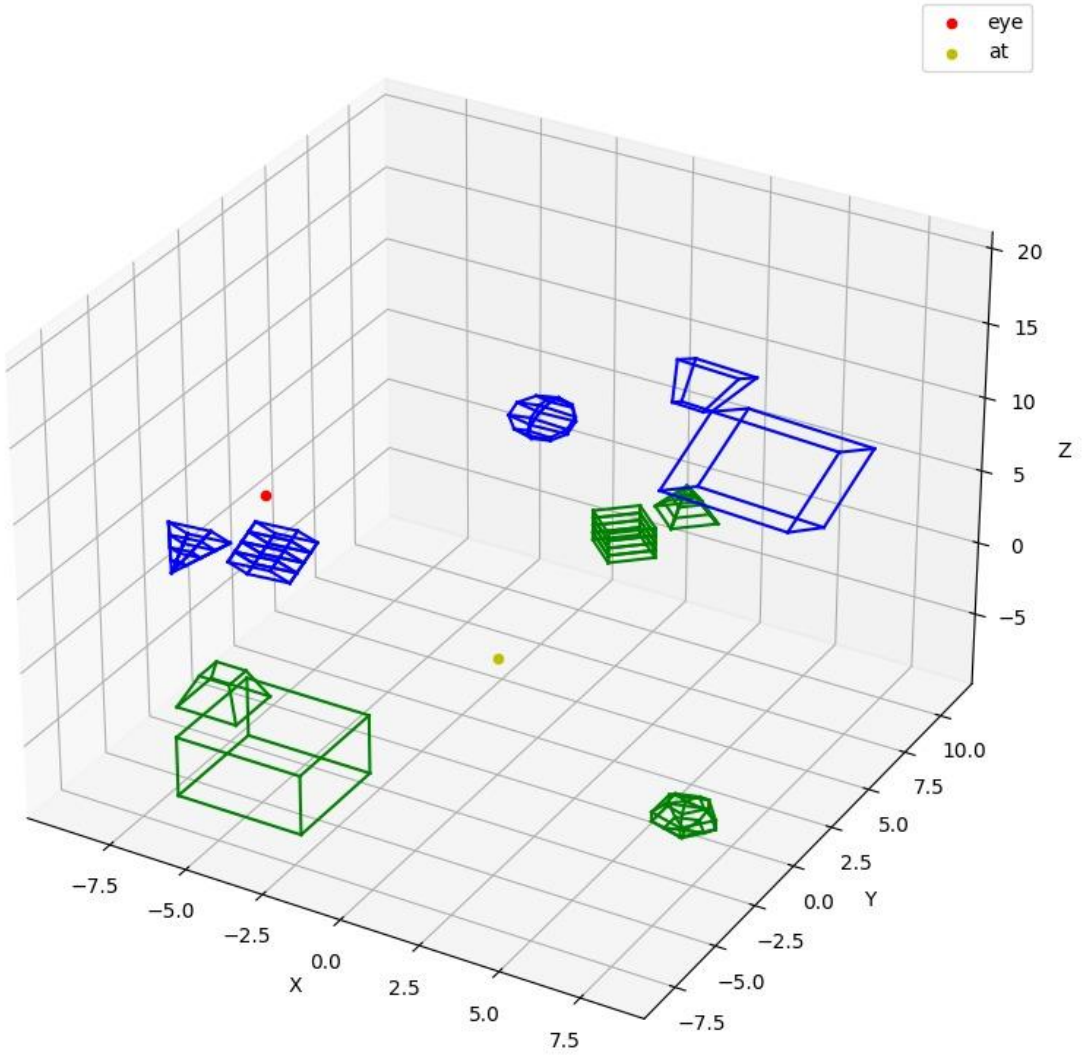
Esses passos cuidadosamente planejados nos permitem estabelecer os vetores (n, u e v) a partir do ponto 'eye', garantindo uma configuração adequada e coerente para a representação visual dos objetos no Sistema de Coordenadas da Câmera

Falando um pouco sobre converter as coordenadas de um objeto ou pontos no sistema de coordenadas mundial para o sistema de coordenadas da câmera. A lógica por trás da função é baseada em transformações lineares. Ela usa uma matriz homogênea para realizar a transformação dos pontos do sistema de coordenadas mundial para o sistema de coordenadas da câmera. Essa matriz é construída a partir dos vetores U, V, N e da posição da câmera eye.

Os vetores U, V, N representam os eixos da câmera no sistema de coordenadas mundial. Esses vetores são usados para definir uma base no sistema de coordenadas da câmera. A posição da câmera eye é usada para determinar a origem do sistema de coordenadas da câmera.

Ao multiplicar a matriz homogênea pelos pontos da vertexList (juntamente com uma coluna de "1" adicionada a cada ponto para torná-lo um ponto homogêneo), a função realiza uma combinação linear dos vetores de base da câmera com as coordenadas dos pontos. Isso resulta em uma transformação dos pontos do sistema de coordenadas mundial para o sistema de coordenadas da câmera.

A função retorna os pontos transformados, agora expressos no sistema de coordenadas da câmera, na variável "cameraVertexList". Confira a saída abaixo.

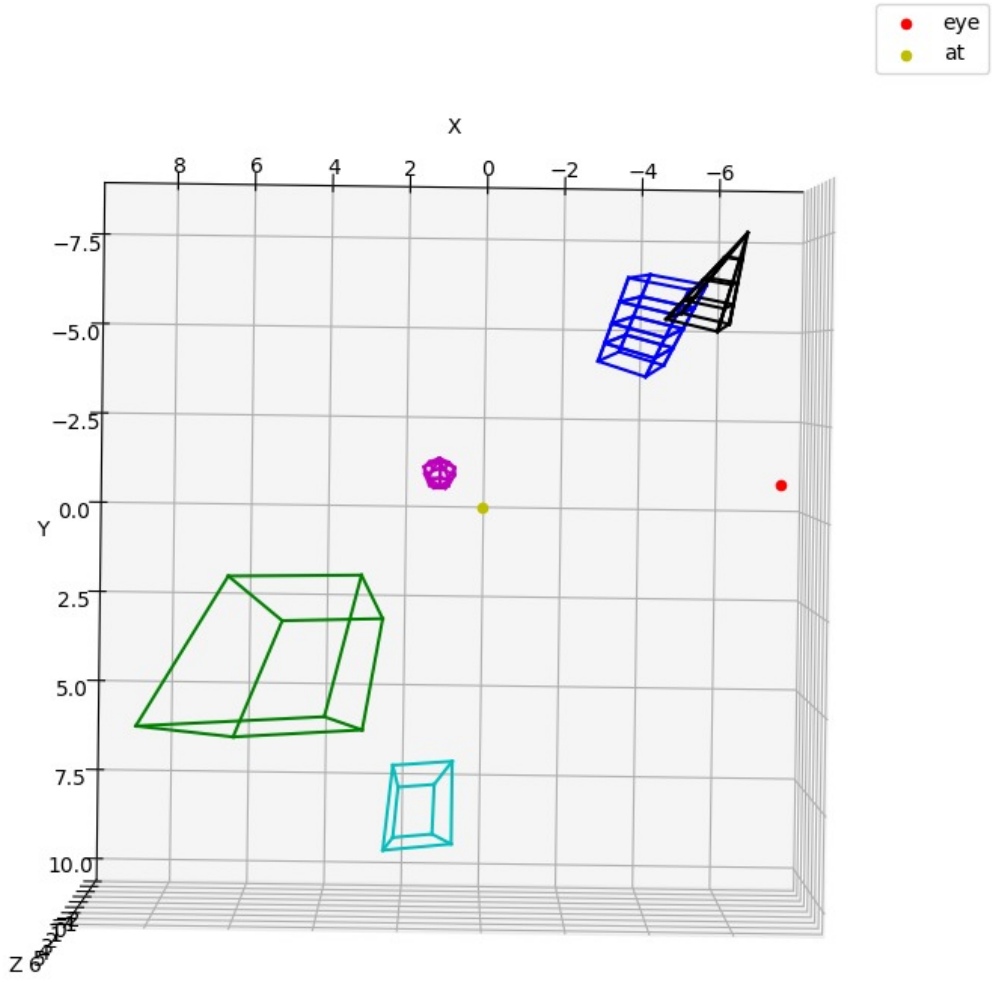


2.5 - Projeção em Perspectiva:

A função TransformacaoEmPerspectiva é responsável por aplicar uma transformação em perspectiva em um conjunto de vértices tridimensionais. Ela recebe dois parâmetros: vertices, que é uma matriz contendo os vértices tridimensionais representados como colunas, e alpha, que é o ângulo de visão desejado.

Primeiro, a função cria uma matriz homogênea ao empilhar os vértices com uma linha adicional de uns. Essa matriz homogênea é uma representação estendida dos vértices, permitindo operações matriciais mais eficientes.
Em seguida, a função inicializa uma matriz vazia chamada verticesEmPerspectiva, que terá o mesmo formato que a matriz homogênea.A função então itera sobre cada coluna da matriz homogênea usando um loop for. Para cada vértice, é criada uma matriz de transformação em perspectiva usando o valor de alpha. Essa matriz de transformação é uma matriz 4x4 que leva em consideração a distância do vértice em relação à câmera e o ângulo de visão. Utilizando a multiplicação de matrizes (@), a função multiplica a matriz de transformação pelo vértice correspondente da matriz homogênea. Isso resulta em um novo vértice transformado em perspectiva.

A matriz verticesEmPerspectiva é atualizada com os valores transformados do vértice atual, repetindo o processo para todos os vértices. Após o loop, a última linha da matriz verticesEmPerspectiva é removida, pois não é mais necessária. Além disso, os valores Z dos vértices são definidos como zero. Por fim, a função retorna a matriz verticesEmPerspectiva, que contém os vértices transformados em perspectiva.



3 - Conclusão:

Ao concluir este relatório, é imprescindível ressaltar a árdua e complexa natureza deste trabalho. A compreensão dos resultados de cada transformação revelou-se mais desafiadora do que a própria implementação em si, exigindo um esforço considerável. No entanto, essa experiência foi extremamente enriquecedora para o aprendizado do grupo como um todo, impulsionando o crescimento de nossos conhecimentos sobre a ferramenta utilizada no desenvolvimento do projeto.

Através da realização de inúmeros testes e da minuciosa observação dos resultados, pudemos explorar amplamente os diversos sistemas de coordenadas apresentados e as transformações aplicadas às figuras. Essa abordagem meticulosa nos permitiu alcançar o resultado esperado e adquirir um domínio mais aprofundado desses conceitos complexos.

O grupo encontra-se extremamente satisfeito com os resultados alcançados e com o conhecimento adquirido durante toda essa experiência desafiadora. Essa experiência não apenas consolidou nossas habilidades técnicas, mas também fortaleceu nossa capacidade de enfrentar problemas complexos e encontrar soluções criativas. Estamos confiantes de que esses aprendizados serão inestimáveis em nossos futuros projetos e contribuirão significativamente para o nosso crescimento profissional e acadêmico.

4 - Referências Bibliográficas:

AZEVEDO, Eduardo; CONSI, Aura. Computação Gráfica, Teoria e Prática : Geração de imagens. 2° edição. Rio de Janeiro: GEN LTC, 24 julho de 2018