UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO
COMP0431 - Computação Gráfica
Prof.: Dra. Beatriz Trinchão Andrade de Carvalho
Alunos: Résmony Silva Muniz
João Marcelo dos Santos Nascimento

Mosca Fantasma

1. Instruções de uso

Compilação (Linux):

```
g++ -Wall main.cpp CameraView.cpp SceneManager.cpp Utils/utils.cpp Utils/objloader.cpp Utils/shader.cpp Utils/texture.cpp -o fly -lGLEW -lGL -lGLU -lglut -lglfw - lSOIL

./fly Files/input.txt
```

A solução proposta utiliza as bibliotecas especiais `glm`, `soil`.

Comandos teclado:
Up - Translada a câmara para cima em seu sistema de coordenadas.
Left - Translada a câmera para a esquerda em seu sistema de coordenadas.
Down - Translada a câmera para baixo em seu sistema de coordenadas.
Right - Translada a câmera para a esquerda em seu sistema de coordenadas.
F1 - Retorna à posição inicial da câmera no espaço. Posição (5, 3.5, 15), LookAt (30, 2.5, 15).

Comandos mouse:
Scroll up - Move a câmera na direção -z de seu sistema de coordenadas.
Scroll down - Move a câmera na direção +z de seu sistema de coordenadas.
Rotação - Movimentando o mouse com qualquer botão pressionado.

2. Câmera

A implementação da câmera foi baseada na abordagem disponível no site LearnOpenGL (https://learnopengl.com/Getting-started/Camera).

A Figura 1 a seguir ilustra como a câmera é representada na cena.

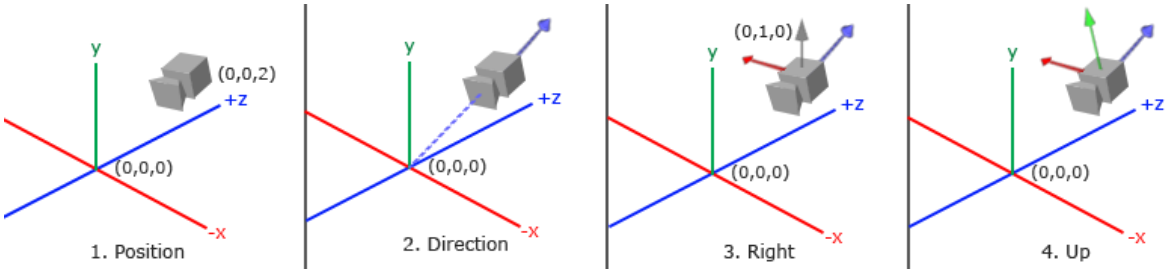

Figura 1 - Câmera

Quando é inicializada, é dada a posição inicial em que a câmera ficará no espaço e a direção que ela aponta. Os comandos dados pelo teclado/mouse para movimentação de câmera alteram sua direção ou posição. Ao pressionar as teclas up, left, right, down no teclado e srcoll up, scroll down no mouse, alterará somente a posição da câmera. Já quando o mouse é movimentado com algum de seus botões pressionado, somente a direção que a câmera aponta sofrerá alteração.

A rotação da câmera utiliza os ângulos de Euler, para representar a rotação 3D.

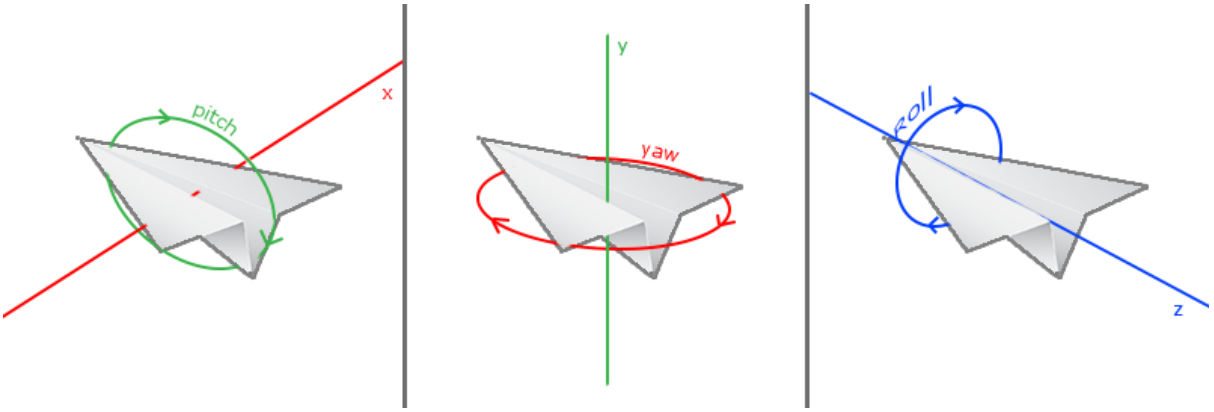

Figura 2 - Ângulos de Euler

3. Cena

3.1 Disposição no espaço

Uma vez que a geração de salas é dada de forma dinâmica no programa de acordo com a entrada do arquivo de texto, foi pensado então em gerar os limitantes da sala iterativamente sobre o plano XZ.

As salas possuem tamanho fixo de 30x30 no plano XZ e paredes com altura 5 no eixo Y. Todas as salas estão dispostas no octante positivo do sistema de coordenadas.

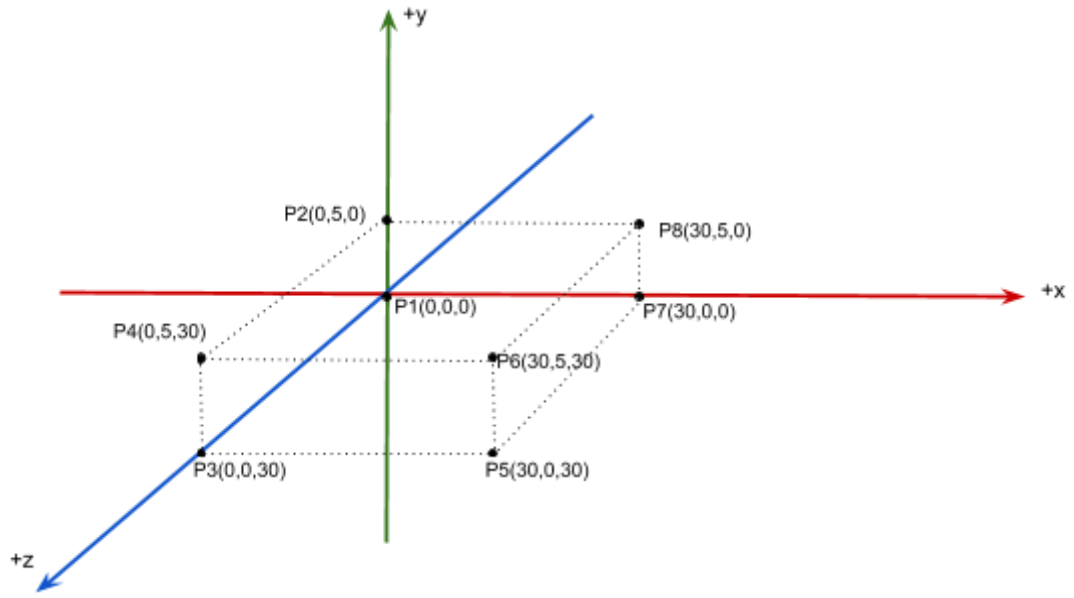

Figura 3 - Distribuição das salas no espaço

A Figura 3 acima representa como está disposta a primeira sala no sistema de coordenadas do espaço. As demais salas são dispostas seguindo a mesma estrutura no sentido positivo do eixo x.

Os pontos seguem a ordem de passagem de vértices em `GL_QUAD_STRIP`.

3.2 Disposição dos objetos

A quantidade de objetos de uma sala é dada através de uma distribuição normal. Para calcular a quantidade de cada objeto, o método `normal_distribution<>` disponível na biblioteca `random` do C++ foi utilizada para gerar a distribuição gaussiana e logo após um número aleatório dentro dos limitantes e com tendência foi sorteado.

Depois de obtida a quantidade de cada tipo de objeto, números aleatórios dentro do espaço das respectivas salas foram sorteados, verificando se a posição p(x, . , z) estava no mínimo a uma distância de 1 da posição dos outros objetos. Tal abordagem visa impedir colisão de objetos, no entanto, quanto maior o número de objetos, maior o tempo de processamento. A depender do número de objetos em cada sala, esta solução não é a ideal com valores de ponto flutuante, pois para cada coordenada p1(x1, . , z1) existem infinitos pontos p2(x2, . , z2) onde dist(x1, x2) ou dist(z1, z2) é menor que um. Para minimizar esse valor, as coordenadas são tratadas como valores inteiros.

3.3 Texturas

A aplicação de texturas nas paredes e piso é dada de forma aleatória para cada sala.

Inicialmente, quando está se definindo a quantidade de cada objeto nas salas e a sua posição, é associada à parede e ao piso um valor que será posteriormente associado a uma

textura. Nas imagens a seguir, são mostradas capturas de tela do programa executando com o arquivo de entrada dado como exemplo.

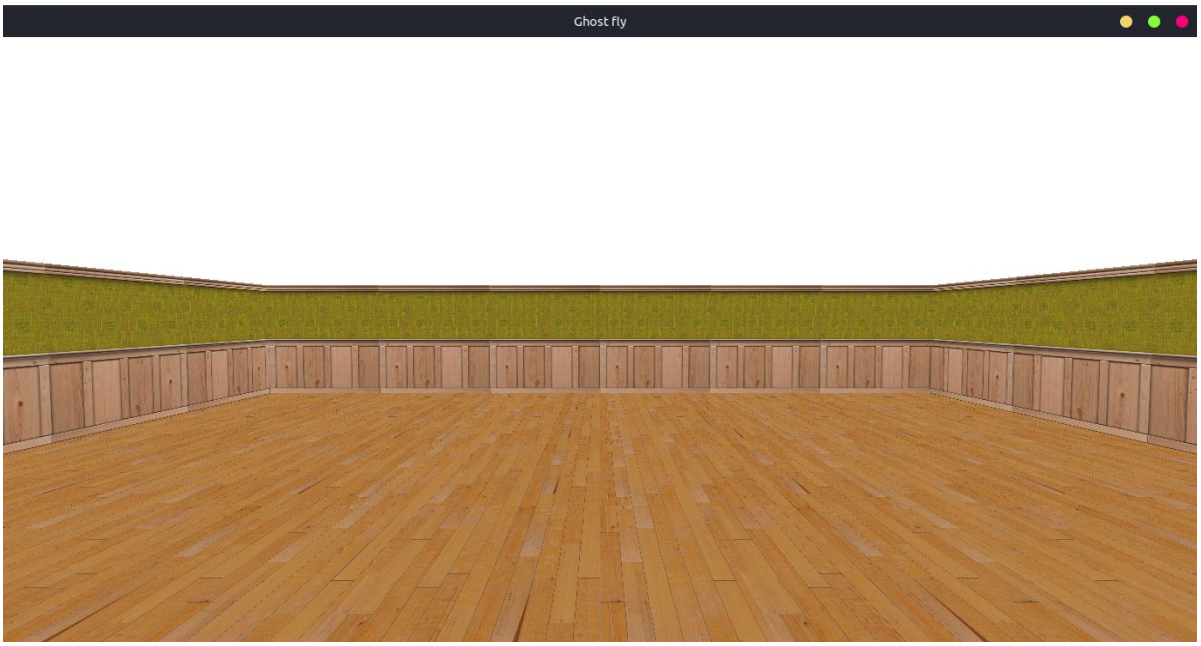

Figura 4 - Sala 1

Figura 5 - Sala 2

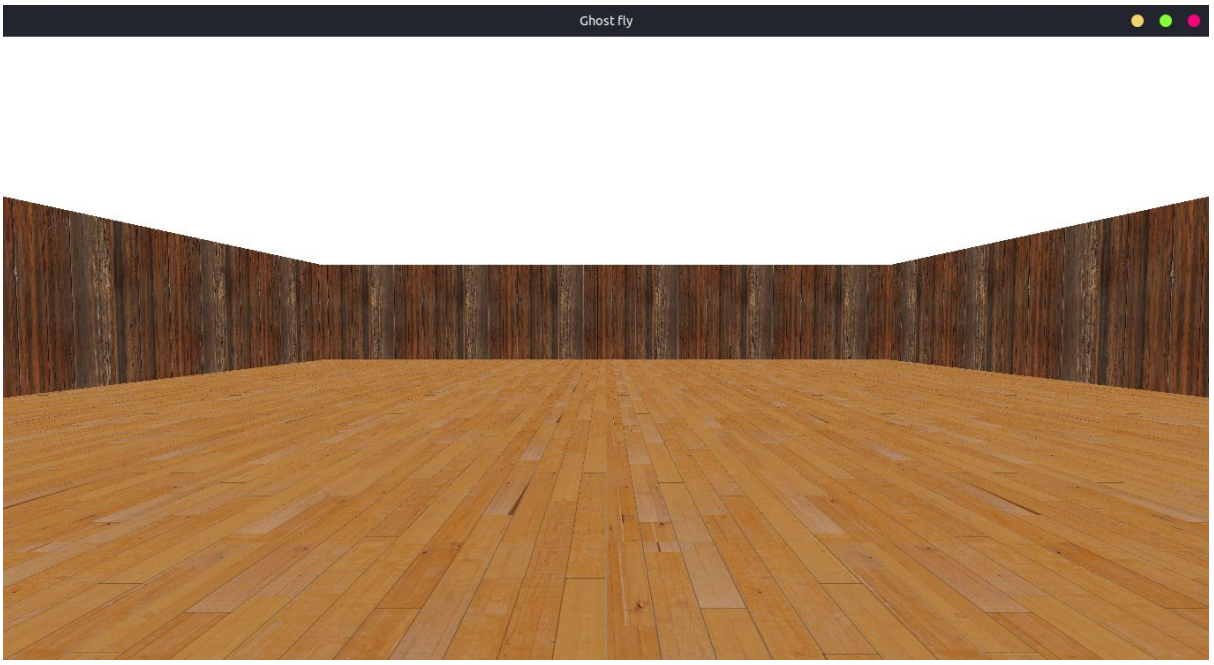

Figura 6 - Sala 3

As texturas utilizadas para esse projeto foram baixadas do site Turbo Squid (https://www.turbosquid.com/).

3.4 Carregamento de objetos

Para o carregamento dos objetos na cena várias maneiras foram buscadas, como por exemplo, as bibliotecas lib3ds e assimp.

Infelizmente essa parte ficou incompleta porque não foi encontrada uma boa documentação da biblioteca lib3ds, tornando não trivial usá-la. A biblioteca assimp chegou a ser utilizada mas os objetos carregados ficaram estranhos e com a textura incorreta.

A versão atual do projeto tentou carregar os objetos com extensão .obj usando array com o '`load_obj`', método utilizado que foi encontrado no seguinte tutorial para carregamento de modelos 3D: <http://www.opengl-tutorial.org/beginners-tutorials/tutorial-7-model-loading/>. Contudo, esse procedimento não funcionou muito bem em outros modelos com a extensão .obj, só funcionava com o exemplo do site e ainda assim a exibição da textura nesse objeto não foi possível reproduzir. Acreditamos que o problema está em `vertexshader.vs` e `fragmentshader.fs`, pois é quando o programa tenta carregá-los que ocorre o problema. Ambos arquivos na encontram-se na pasta `Utils`.

O método `processModels` da classe `SceneManager` seria o responsável por carregar os objetos. A grande parte comentada no método `displayObjects` dessa classe seria responsável por exibir os objetos na cena, no entanto, os objetos não possuem a textura que deveria assim como dado no exemplo do tutorial seguido. A Figura 7 mostra como ficou a cena ao carregar o modelo 3D.

Outros tutoriais e códigos para carregamento de .obj foram utilizados como base para aplicar essa funcionalidade, no entanto, sem sucesso.

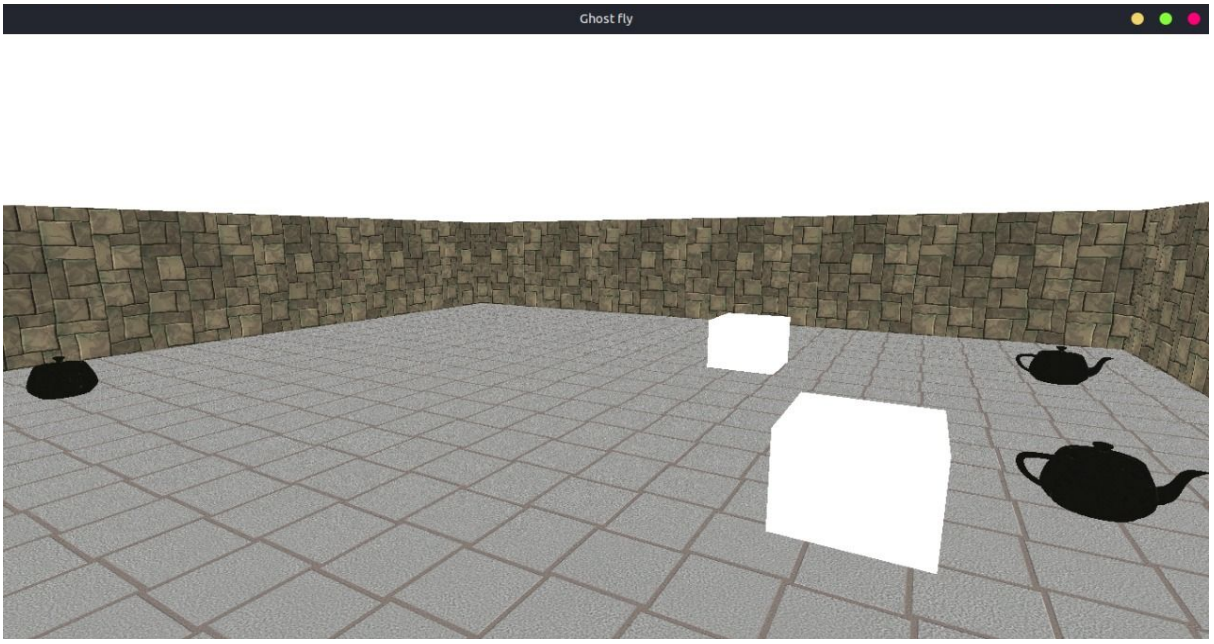

Figura 7 - Versão com carregamento de obj

Textura `Textures/Floor/7.jpg` aplicada ao teapot.

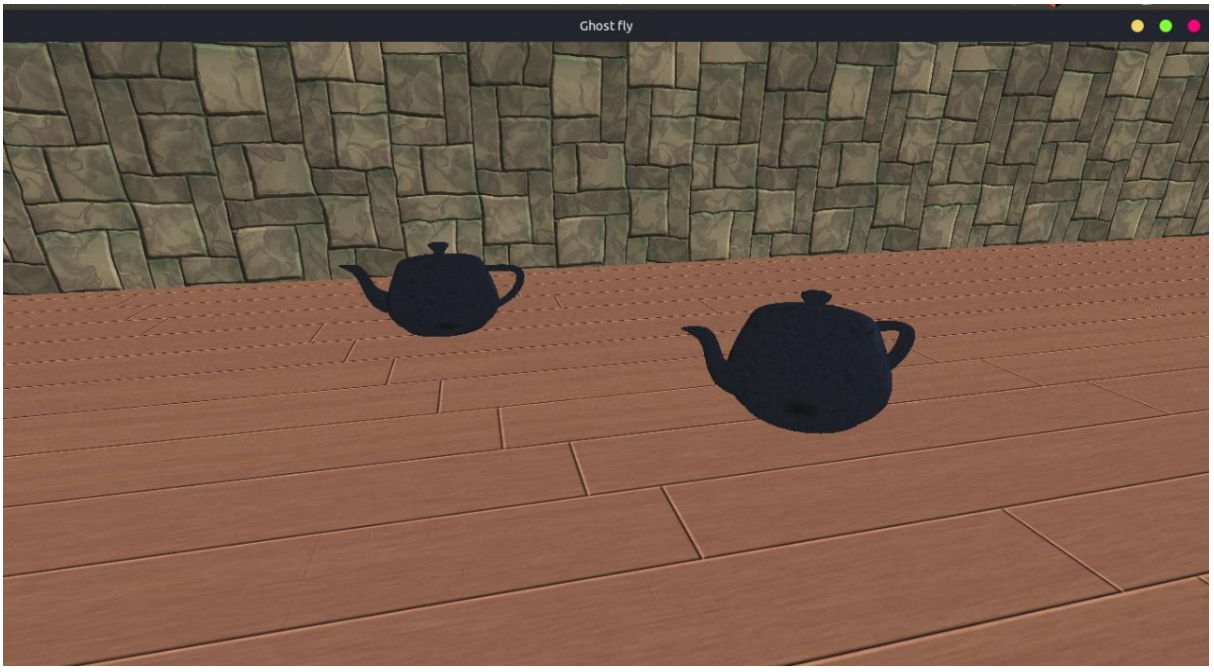

Figura 8 - Teapot com textura

A Figura 9 abaixo mostra um exemplo da execução do projeto exibindo na cena teapots e rosquinhas. Quando a câmera se aproxima do torus, há uma efeito peculiar de sombreamento, certamente causado por alguma falha na configuração das luzes.

Figura 9 - Versão entregue

Alguns dos links utilizados como referência para aplicar o carregamento de .obj

https://github.com/huamulan/OpenGL-tutorial/blob/master/common/objloader.cpp
https://en.wikibooks.org/wiki/OpenGL_Programming/Modern_OpenGL_Tutorial_Load_OBJ
http://syoyo.github.io/tinyobjloader

Nota:
Inicialmente o trabalho havia sido definido como em grupo, no entanto, como alguns participantes da equipe não mostraram interesse em fazer o projeto, o trabalho resultante ficou incompleto.
Diante do prazo e das funcionalidades restantes que deveriam ser adicionadas, eu não consegui fazer tudo. O carregamento de modelos 3D, de forma correta, foi a última funcionalidade que não foi implementada de forma adequada, assim como as devidas configurações de iluminação, as quais deveriam ser tratadas após o carregamento dos objetos para considerar as normais no modelo 3D.