Relatório do Projeto de Computação Gráfica

Cenário Virtual

1. Identificação

- Integrantes do Grupo:

• Rodrigo Yamaya Gonçalves

• Lucas dos Santos Ottvagen

• Luiz Felippe Almeida Veloso

2. Instruções de Instalação e Execução

Bibliotecas Necessárias

Para executar a aplicação, é necessário ter o Python instalado (versão 3.10) e instalar as seguintes dependências:

• **PyGame (pygame)**: Responsável pela criação da janela, gerenciamento de contexto OpenGL e captura de entrada (teclado/mouse).

• **PyOpenGL (PyOpenGL)**: Bindings do OpenGL para Python.

• **PyGLM (PyGLM)**: Biblioteca matemática para cálculos vetoriais e matriciais (essencial para a câmera e transformações).

• **NumPy (numpy)**: Manipulação eficiente de arrays de dados.

• **Pillow (Pillow)**: Utilizada para carregar texturas de imagem.

Comando para instalação rápida:

pip install pygame PyOpenGL PyGLM numpy Pillow

Instancing e Modelos 3D:

Para ler os personagens implementados no projeto é necessário habilitar o uso do formato .fbx

Para isso deve-se baixar a Autodesk FBX Python SDK:

(FUNCIONA APENAS NA VERSÃO 3.10 DO PYTHON)

Como baixar:

-Acesse o site de SDKs da Autodesk: https://aps.autodesk.com/developer/overview/fbx-sdk

-Procure a sessão “FBX Python SKD”; Baixe o SDK conforme o seu sistema operacional e instale;

-Vá até a pasta de instalação e leia o arquivo “README.md” para habilitar a biblioteca no python;

-No caso do Windows: a pasta de instalação deve ser algo como: “C:\Program Files\Autodesk\FBX\FBX Python SDK\2020.3.7”;

-E para habilitar, deve se abrir o prompt nessa pasta e executar “python -m pip install %name_of_the_wheel_file%.whl”.

-Após isso o fbx estará baixado na máquina

Por fim realizamos:

• Carregamento de múltiplos personagens convertidos para FBX.
• Distribuição aleatória de 100 instâncias pelo terreno, com escala e rotação variadas.

• Ajuste automático de altura para garantir que os pés dos personagens toquem o solo corretamente.

Script Inicial

O ponto de entrada da aplicação é o arquivo **main.py**. Para iniciar, abra a pasta do projeto e execute no terminal:

“cd projeto”

“py -3.10 main.py”

Após acionar o programa **main.py**, aparecerá 3 mensagens no terminal:

1 = Gerar cena (spawn_personagens.py)

2 = Visualizar cena (OpenGL)

3 = Sair

Ao digitar a opção 1, o usuário irá gerar a cena.

Ao digitar a opção 2, o usuário será redirecionado para visualizar a cena.

Ao digitar a opção 3, o usuário irá fechar o programa, encerrando a aplicação.

3. Descrição dos Controles (Interação do Usuário)

-A aplicação simula uma câmera em primeira pessoa (FPS), onde o usuário controla a visão e o movimento de um observador no cenário.

Movimentação:

• **Teclas W / S**: Movem a câmera para frente e para trás no plano horizontal (eixo XZ). A movimentação é travada no chão para evitar que o observador "voe" ou entre na terra.

• **Teclas A / D**: Movem a câmera lateralmente para a esquerda e direita.

• **Tecla SHIFT (Esquerdo)**: Ativa a corrida, dobrando a velocidade de movimento enquanto pressionada.

• **Tecla ESPAÇO**: Realiza um pulo. A física inclui gravidade, trazendo o observador de volta ao chão (altura dos olhos em 1.8 unidades).

Orientação (Visão):

Mouse: Controla a direção do olhar.

• Movimento horizontal: Gira a câmera para os lados.

• Movimento vertical: Olha para cima ou para baixo, com limite de 89 graus para evitar inversão da câmera.

• **Cursor**: O cursor do mouse é ocultado e travado na janela para permitir rotação infinita (estilo FPS).

Outros Comandos:

• **Tecla ESC**: Encerra a aplicação.

• **Setas Direita / Esquerda**: Aceleram ou desaceleram a passagem do tempo (ciclo dia/noite) para fins de demonstração.

4. **Técnicas de Computação Gráfica Utilizadas**

O projeto foi desenvolvido utilizando OpenGL Avançado com shaders programáveis (GLSL).

As principais técnicas implementadas foram:

1- **Geração de Terreno**:

• Carregamento de malha irregular a partir de arquivo .OBJ (FBX models/terreno.obj).

• Aplicação de textura difusa (grass.png).

2- **Iluminação Dinâmica (Ciclo Dia/Noite)**:

• Uma fonte de luz direcional simula o Sol, orbitando a cena de Leste para Oeste.

• A cor do céu (glClearColor) e da luz ambiente é interpolada dinamicamente baseada na altura do sol, criando transições suaves entre amanhecer, dia, entardecer e noite.

• Visualização da posição do sol através de uma esfera renderizada no céu.

3- **Sombras em Tempo Real (Shadow Mapping)**:

• Implementação da técnica de Shadow Mapping em dois passos.

• **Passo 1**: Renderização da cena do ponto de vista da luz (Sol) para um Framebuffer de profundidade (Depth Map).

• **Passo 2**: Renderização da cena normal, comparando a profundidade do fragmento com o mapa de sombra para determinar oclusão.

• Uso de PCF (Percentage-Closer Filtering) nos shaders para suavizar as bordas das sombras.

• Correção de artefatos visuais (Shadow Acne) utilizando glCullFace(GL_FRONT) durante o passo da sombra.

4- **Neblina Volumétrica (Fog)**:

• Cálculo de neblina exponencial quadrática no Fragment Shader.

• A cor da neblina adapta-se automaticamente à cor do céu (ciclo dia/noite), garantindo coesão visual.