# OVNI vs. Vaca

## Computação Gráfica

## Integralização GB

Carolina Paz Prates
Kevin Mulinari Kuhn
Vítor Mello

# DECISÕES DE PROJETO

**Estrutura de Classes**
*Câmera:* encapsula a movimentação e orientação da câmera.
*Modelo:* carrega e armazena dados de objetos .obj, suas texturas e buffers.
*Shaders:* dois sistemas de shader, um para a cena principal, outro para o céu (skybox).
*Funções utilitárias:* carregamento de texturas, compilação de shaders, carregamento de modelos com .obj e .mtl.

**Iluminação** Phong com spotlight, ajustando:
*Ka, Kd, Ks:* componentes ambiental, difusa e especular.
*Shininess:* controle da reflexão especular.
*Intensidade e cutoff:* simula uma lanterna (feixe cônico do OVNI ou luz da casa).
*Atenuação pela distância:* realismo na dissipação da luz. Quando a luz da casa está acesa, ela é a fonte principal (cor branca, posição fixa).
Quando apagada, o foco do OVNI se torna a fonte de luz, com coloração esverdeada.

**Câmera**
Livre (freecam) controlada via teclado e mouse
O teclado move a posição da câmera no espaço.
O mouse altera o ângulo de visão (yaw e pitch).
A movimentação é suavizada com deltaTime, garantindo fluidez independente do FPS.

**Animação**
Ocorre com base em estados lógicos (luz acesa ou apagada). Se a luz da casa está:
*Acesa:* o OVNI sobe e a vaca desce.
*Apagada:* o OVNI desce e a vaca é abduzida.
A posição vertical do OVNI e da vaca é interpolada de acordo com o tempo (deltaTime) para fluidez.
O OVNI também gira continuamente ao redor do eixo Y para dar dinamismo visual.

**Interação**
O usuário interage via teclado (`W`, `A`, `S`, `D` para mover a câmera) e mouse (rotação da câmera).
A tecla `H` liga ou desliga a luz da casa, disparando a sequência de abdução ou fuga.

config.ini

```ini
[window]
width=800
height=600
title=OVNI vs Vaca!

[modelo_paths]
ovni=../assets/Modelos3D/final/Nave.obj
vaca=../assets/Modelos3D/final/vaca.obj
casa=../assets/Modelos3D/final/casa.obj

[alturas]
abducao=5.0
fuga=15.0

[curvas]
amplitude=1.0
```

# Informações técnicas sobre a cena

Utilizamos materiais carregados automaticamente e um skybox fixo para compor visualmente a cena, com controle individual de iluminação e textura por objeto.

## Materiais e Texturas

Cada .obj carrega materiais do .mtl, incluindo textura e propriedades.

Ka, Kd, Ks e shininess são aplicados via struct Material.

O chão é um plano manual com textura grama.png.

Coordenadas de textura Y são invertidas para corrigir a imagem.

## Skybox e Iluminação

O fundo da cena usa um skybox 2D com a imagem ceu.png.

A skybox é renderizada antes dos objetos, sem profundidade.

O shader do skybox é separado do shader principal da cena.

Cada objeto tem controle individual de iluminação (Ka, Kd, Ks).

# Animação e Execução da Cena

A lógica da cena depende do estado da luz da casa. A movimentação e os efeitos visuais são atualizados a cada frame, com fluidez garantida por `deltaTime`.

```
if (!casaLuz && vacaY >= alturaAbducao) {
    float curvaT = t * 2.0f; // velocidade da curva
    float vacaX = curvaAmplitude * sin(curvaT);
    float vacaZ = curvaAmplitude * sin(curvaT) * cos(curvaT);
    posVaca = vec3(vacaX, vacaY, vacaZ);
```

```
// Aplica rotação na vaca apenas quando estiver caindo (casaLuz == true)
if (casaLuz && vacaY < alturaAbducao && vacaY > 0.0f)
    modelVaca = modelVaca * rotate(mat4(1.0f), vacaRot, vec3(1, 0, 0));
```

## Pipeline de Renderização

Entrada do usuário (teclado e mouse) é processada a cada frame.

Skybox é desenhada antes da cena, com profundidade desativada.

Variáveis de iluminação, câmera e materiais são atualizadas.

Objetos são renderizados e buffers trocados com glfwSwapBuffers().

## Animação Avançada

A vaca sobe ou desce conforme a luz da casa (tecla H).

Quando descendo, ela gira até 720° no eixo X.

Quando abduzida, sobe com movimento curvo (seno/cosseno).
O OVNI gira continuamente no eixo Y para dar dinamismo.

## Controle de Tempo

deltaTime é usado para suavizar movimentos em qualquer FPS.

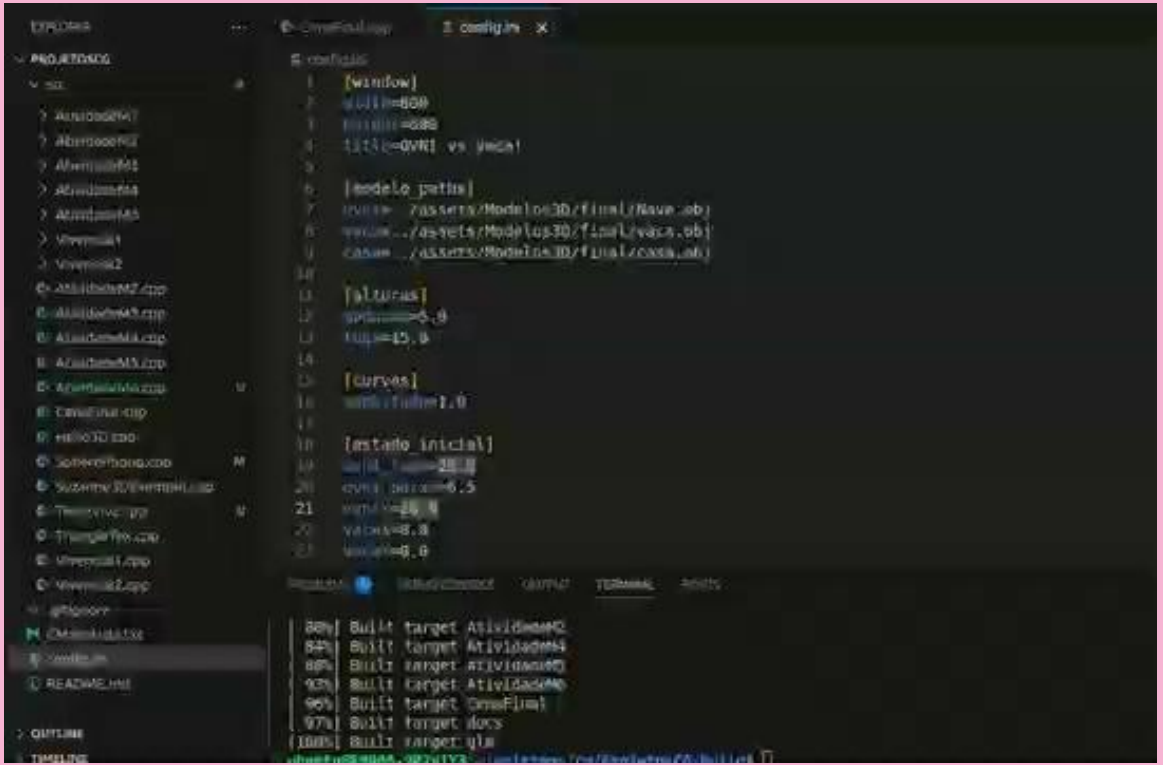
O tempo é calculado com base nos frames (glfwGetTime()).

A interpolação garante consistência visual.

Toda animação responde à lógica e ao estado da luz em tempo real.

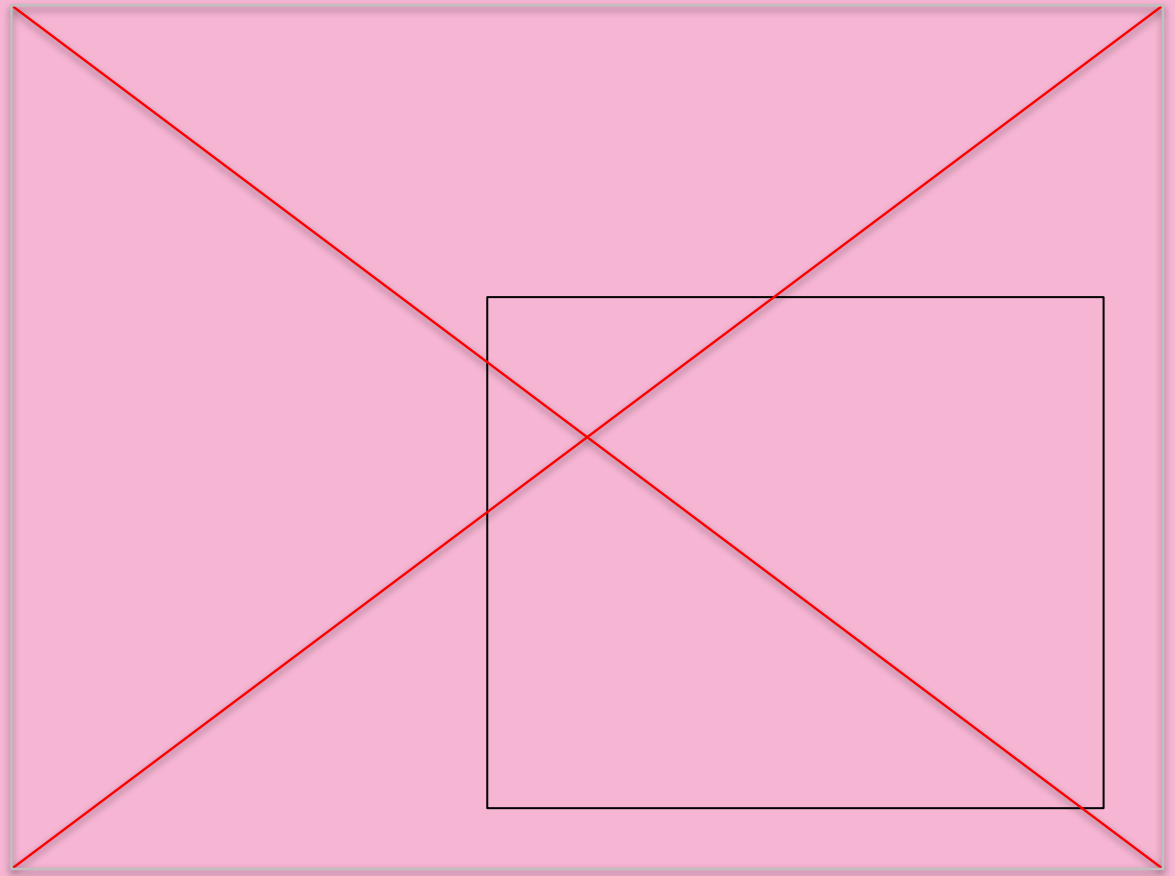
# Demonstração do arquivo de configuração

A importância do arquivo de configuração para não precisar recompilar o código a cada mudança.



# Demonstração da Cena Final

Abaixo, apresentamos a execução da cena em tempo real, com foco na iluminação dinâmica, movimentação da câmera e lógica de abdução.



## DESAFIOS

Encontrar obj triangularizados

Desenvolver o load obj com submashes

Lidar com diversos objetos na cena (céu, grama, casa, etc)

## MELHORIAS

Adicionar mais animais com diferentes reações (correndo, fugindo, sendo abduzidos).

Controlar o OVNI com o jogador (teclado/mouse) e permitir voar pela cena.

Sistema de partículas para feixe de abdução.

# Referências


OpenGL objetos e texturas
https://github.com/srcres258/learnopengl-rust/tree/master

Learn OpenGL Tutoriais
https://learnopengl.com/Code-repository

Learn OpenGL Tutoriais
https://learnopengl.com/Code-repository

INI file
https://en.wikipedia.org/wiki/INI_file

Código Prof. Rossana
https://github.com/fellowsheep/CG2024-2

Código-Fonte  da Cena
https://github.com/cpprates/ProjetosCG/blob/main/src/AtividadeGB/README.md