
**UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO**
Computação Gráfica - Turma 1
Prof.ª: Lis Custódio
2025.1

# Trabalho 2: Implementação de um Jogo 3D

**Nome:** Arthur Henrique de Souza Fernandes | **Matrícula:** 202110035911

**Nome:** Maria Eduarda Alves da Silva | **Matrícula:** 202210033211

**Nome:** Samira dos Santos Magalhães Ferreira | **Matrícula:** 202210035511

## 1. Visão Geral

Este trabalho consiste na evolução do jogo 2D do pinguim, desenvolvido no primeiro trabalho, para três dimensões. Foi utilizado C++ e a biblioteca OpenGL/GLUT.

O objetivo principal permanece: o jogador controla uma "mãe pinguim" que deve coletar peixes e levá-los para alimentar seu filhote. Agora, o desafio se desenrola em um cenário 3D, onde o jogador deve navegar e evitar cair em buracos no gelo. O jogo incorpora modelagem 3D, texturização, iluminação e um sistema de múltiplas câmeras para enriquecer a jogabilidade.

## 2. Principais Mecânicas

### 2.1. Controles e Movimentação

A movimentação foi adaptada para o cenário 3D, adotando um esquema de controle de "tanque":

- **Setas cima/baixo:** Movem a pinguim para frente e para trás, respectivamente, na direção para a qual ela está virada.

- **Setas esquerda/direita:** Rotacionam a pinguim sobre seu próprio eixo (eixo Y), alterando a direção do movimento.

O cenário é delimitado por um grande plano de gelo, e o jogador deve se manter dentro de seus limites.

### 2.2. Sistema de Energia

O sistema de objetivos e condições de vitória/derrota foi mantido e adaptado do trabalho anterior:

- **Energia do Filhote:** O pinguim filhote possui uma barra de energia que se esgota com o tempo (começa em 60 segundos). Cada vez que é alimentado, sua energia é restaurada. Se a energia chegar a zero, o jogo termina.

- **Tempo de Jogo:** A partida tem uma duração total de 5 minutos (300 segundos). Se o jogador sobreviver até o final do tempo, ele vence.

- **Obstáculos:** Diferente da versão 2D, não há um pássaro predador. O principal obstáculo são os buracos no gelo. Cair em um deles reposiciona a mamãe pinguim na sua posição inicial, custando tempo.

### 2.3. Elementos do Jogo

- **Mamãe Pinguim:** Personagem controlável pelo jogador. Responsável por explorar o cenário, coletar os peixes e alimentar o filhote.

- **Pinguim Filhote:** Personagem estático que aguarda em uma posição fixa para ser alimentado.

- **Peixes:** Itens coletáveis que surgem em posições aleatórias no cenário.

- **Buracos:** Obstáculos distribuídos aleatoriamente pelo gelo.

- **Cenário 3D:** Composto por um chão de gelo texturizado e um *Skybox* para criar a ilusão de um ambiente ártico infinito.

## 3. Funcionalidades e Códigos

### 3.1. Variáveis Globais

As variáveis de estado do jogo, como *tempoTotal*, *energiaPinguimzinho*, pontos e *jogoAtivo*, foram mantidas. As principais adições foram:

- **Coordenadas 3D:** Variáveis como *pinguimX*, *pinguimY* e *pinguimZ* para controlar a posição dos objetos no espaço.

- **Rotação:** A variável *pinguimRotacao* armazena o ângulo de orientação da mamãe pinguim.

- **Estruturas de Dados:** structs para Peixe e Buraco foram criadas para organizar suas propriedades (posição, raio, estado de captura). *std::vector* é utilizado para gerenciar múltiplos peixes e buracos de forma eficiente.

- **Recursos Gráficos:** Variáveis do tipo *GLuint* para armazenar os identificadores das texturas e um ponteiro *GLUquadric* para desenhar formas curvas como esferas e cilindros.

### 3.2. Função *init()*

A função *init()* foi expandida para configurar um ambiente de renderização 3D. Suas principais responsabilidades são:

- **Configurações 3D:** Habilita o teste de profundidade (*GL_DEPTH_TEST*) para que os objetos sejam desenhados na ordem correta, e a iluminação (*GL_LIGHTING*, *GL_LIGHT0*) para dar volume e realismo à cena.

- **Carregamento de Texturas:** Utiliza a função auxiliar *carregarTextura* (que emprega a biblioteca *stb_image.h*) para carregar arquivos de imagem (*.jpg*) para a neve, pinguins, peixes e o céu.

- **Inicialização de Objetos:** Cria um objeto *GLUquadric* com a função *gluNewQuadric()*, que é essencial para desenhar as formas primitivas usadas nos modelos.

### 3.3. Modelagem 3D e Texturização

Os personagens e objetos são construídos a partir de primitivas 3D do GLUT/GLU:

- ***desenhaPinguim* e *desenhaPeixe*:** Utilizam *gluSphere* e *gluCylinder* para compor seus corpos, cabeças e caudas.

- **Texturização:** A função *gluQuadricTexture(quadric, GL_TRUE)* habilita a aplicação de coordenadas de textura sobre as primitivas GLU. Com isso, as imagens carregadas na função *init()* são mapeadas nos modelos, conferindo-lhes aparência de pelos, escamas, etc.

### 3.4. Cenário 3D

O cenário é construído com dois elementos principais:

- **Plano do Gelo:** Um grande polígono (*GL_QUADS*) é desenhado no plano XZ com coordenadas de textura que se repetem, dando a aparência de um vasto chão de gelo.

- **Skybox:** A função *desenhaSkybox* desenha um grande cubo centralizado na câmera. Cada uma das seis faces internas do cubo recebe uma textura do céu, criando um fundo panorâmico e imersivo que envolve toda a cena.

### 3.5. Função *verificaColisao()*

A detecção de colisão foi adaptada para o ambiente 3D, calculando a distância no plano XZ ( *sqrt(dx*dx + dz*dz)* ):

- **Colisão com Peixe:** Se a mamãe pinguim se aproxima o suficiente de um peixe, ela o coleta (*carregandoPeixe* = *true*).

- **Colisão com Filhote:** Se a mamãe estiver carregando um peixe e se aproximar do filhote, ela o alimenta. Isso zera o status *carregandoPeixe*, restaura a energia do filhote, incrementa a pontuação e reposiciona todos os peixes e buracos no cenário.

- **Colisão com Buraco:** Se a pinguim se aproxima do centro de um buraco (dentro de seu raio), sua posição é reiniciada para seu local inicial, e qualquer peixe que ela estivesse carregando é perdido.

### 3.6. Função *update()*

A função *update()*, chamada a cada segundo por *glutTimerFunc*, orquestra a lógica do jogo. Ela decrementa a energia do filhote, atualiza o tempo total de jogo, chama *verificaColisao()* para processar interações e verifica as condições de fim de jogo (energia zerada ou tempo esgotado).

### 3.7. Câmera e Viewports

Nesse segundo trabalho, a tela dividida em quatro *viewports*:

A. **Visão Superior (Topo-Esquerda):** Câmera posicionada diretamente acima da pinguim (*gluLookAt(pinguimX, 20.0, pinguimZ, ...)*).

B. **Visão em Perspectiva Livre (Topo-Direita):** Câmera em uma posição diagonal, oferecendo uma visão geral da ação.

C. **Visão Frontal (Baixo-Esquerda):** Câmera posicionada atrás e um pouco acima, similar a uma câmera de perseguição.

D. **Visão Lateral (Baixo-Direita):** Câmera posicionada ao lado da pinguim.

Todas as quatro câmeras são dinâmicas e seguem a posição da mamãe pinguim, garantindo que ela seja sempre o foco do jogo.

### 3.8. Função *display()*

A função *display()* é responsável por renderizar a cena. Ela itera quatro vezes, uma para cada *viewport*. Em cada iteração, ela:

A. Define a área da janela para o *viewport* atual com *glViewport()*.

B. Configura a projeção em perspectiva (*gluPerspective()*) e a câmera (*gluLookAt()*) específica daquela visão.

C. Desenha todos os elementos da cena: o *skybox*, o chão de gelo, os buracos, os pinguins e os peixes.

D. Após desenhar os *viewports*, chama *exibeTempo()* para renderizar as informações de status (tempo, energia, pontos) em 2D sobre a tela.

### 3.9. Função *specialKeys()*

A função *specialKeys()* lida com a entrada do teclado (setas). A movimentação para frente/trás é calculada usando seno e cosseno do ângulo de rotação (*pinguimRotacao*), o que move a pinguim na direção em que ela está apontando. As teclas de esquerda/direita simplesmente ajustam esse ângulo.

### 3.10. Função *main()*

A função *main()* inicializa o GLUT e cria a janela do jogo. A principal mudança em relação ao trabalho anterior é a definição do modo de exibição para *GLUT_DOUBLE* | *GLUT_RGB* | *GLUT_DEPTH*, onde *GLUT_DEPTH* é crucial para habilitar o buffer de profundidade necessário para a renderização 3D. Ela registra as funções de callback (*display*, *specialKeys*, *update*) e inicia o loop principal do jogo.

## 4. Considerações Finais

Este projeto representou um passo significativo em relação ao trabalho anterior, permitindo a aplicação prática de conceitos fundamentais da computação gráfica 3D. A transição de um ambiente 2D para 3D nos trouxe desafios interessantes, como a implementação de uma câmera dinâmica, controles de navegação em três dimensões, modelagem 3D e o uso de texturas e iluminação para criar uma cena.

Consideramos que os objetivos propostos foram alcançados de forma satisfatória. O resultado é um jogo funcional e completo que demonstra a evolução do nosso entendimento sobre os tópicos estudados, consolidando o aprendizado de forma prática.