Kawan Gomes Moreira
Lucas Santos de Souza
Luís Felipe Custódio Pietschmann
Marcos Felipe Leocadio Rodrigues
Pedro Lucas Brito de Oliveira Vidal
Victor Dias da Silva

DOCUMENTAÇÃO TÉCNICA - PROJETO FINAL EM C

Brasília - DF

INTRODUÇÃO

Este documento tem como objetivo descrever detalhadamente a arquitetura, estruturas de dados, lógica de funcionamento, gerenciamento de arquivos e procedimentos de teste do jogo estilo Flappy Bird implementado em linguagem C.

O projeto é organizado em módulos lógicos e utiliza apenas alocação automática de memória, garantindo portabilidade e simplicidade.

ARQUITETURA DO SISTEMA

A organização do código segue uma divisão modular clara, com arquivos como model.h, game.c, util.c, io.c, logic.c e main.c.

Visão Geral dos Módulos

MODEL: define constantes e estruturas centrais do jogo.
GAME: configuração inicial e reset do jogo.
UTIL: funções de sistema e utilidades.
IO: gerenciamento de arquivos: save, load, recordes e ranking.
GAME: implementa regras de física, colisão, movimentação e pontuação.
MAIN: controla menus, fluxo principal e interação com o usuário.

Fluxo de Execução

1. Inicialização do ambiente (*util_unit* e criação da pasta *data*);
2. Exibição do menu principal;
3. Execução de uma das opções: novo jogo, carregar, salvar, ranking ou sair;
4. Loop principal do jogo:
- desenhar cena
- ler tecla
- aplicar lógica do jogo
- aplicar atraso (*sleep*); e

5. Encerramento, gravação da pontuação e ranking.

ESTRUTURA DE DADOS E MODELO DE JOGO

As estruturas definem o estado completo usado.

Constantes

- *ALTURA_TELA* = 20
- *LARGURA_TELA* = 40
- *MAX_CANO* = 16
- *PASSARO_X* = 5

Essas constantes controlam a exibição e o limite de elementos do jogo.

Diagrama das Structs

Passaro

O personagem central.

- y: posição vertical.
- velocidade: velocidade vertical.

Cano

Cada cano (obstáculos) é definido assim:

- x: posição horizontal.
- gap_y: início da abertura vertical.
- gap_tam: tamanho da abertura.

EstadoJogo

- passaro: struct Passaro.
- canos: vetor fixo de Cano.
- total_canos
- pontuacao
- quadro
- jogo_acabou

Essa struct encapsula todo o jogo, como se fosse um “snapshot” pra salvar e carregar.

ARQUITETURA DE MEMÓRIA E ALOCAÇÃO DINÂMICA

Um ponto relevante é que não existe uso de malloc, pois o jogo é simples e cabe totalmente em memória estática:

- O objeto EstadoJogo jogo é alocado automaticamente na stack.
- O vetor canos[MAX_CANO] é totalmente estático.
- Arrays e strings também são declarados com tamanho fixo.

LÓGICA DO JOGO

Física do Pássaro

A cada quadro:

- velocidade += 1
- y += velocidade

O pulo é implementado assim:
velocidade = -3;

Geração de Canos

A cada 15 quadros, um novo cano é criado na borda direita:
c->x = LARGURA_TELA - 1;
c->gap_tam = 5;
c->gap_y = rand() % (ALTURA_TELA - gap);

Detecção de Colisão

Uma colisão ocorre quando o pássaro bate no chão/teto, ou quando alcança a posição x de um cano e não está dentro do “gap”.

Remoção de Canos

Canos com x < 0 são descartados e o vetor é reorganizado.

EVIDÊNCIAS DE TESTES

Os testes foram executados manualmente para validar o comportamento geral.

Teste 1 (loop básico do jogo)

Objetivo: verificar funcionamento geral do jogo.
Entradas: teclas W, espaço, nenhuma tecla.

Resultados:

- Pássaro sobe e desce corretamente.

- Canos aparecem com espaçamento aleatório.

- Colisão detectada corretamente.

Status: aprovado.

Teste 2 (salvamento)

Depois de iniciar o jogo:

1. entrar no menu, selecionar “Salvar jogo”.

2. verificar arquivo salvo

Resultado esperado: arquivo *data/salvamento.txt* criado.

Conteúdo real observado:

10 1 = pássaro (y, velocidade)

3 = total de canos

38 6 5 = cano 1 (x, gap_y, gap_tam)

27 4 5 = cano 2

16 9 5 = cano 3

12 120 0 = pontuação, quadro, jogo_acabou

Status: aprovado

Teste 3 (Carregamento)

1. Após salvar, fechar e reabrir o jogo;

2. Usar a opção “Continuar”.

O jogo volta exatamente no ponto anterior.

Status: aprovado.

Teste 4 (Melhor Pontuação)

Jogar até morrer, com pontuação maior que o recorde.

Arquivo *melhor_pontuacao.txt* atualizado corretamente.

Status: aprovado.

Teste 5 (Ranking)

Ao finalizar o jogo:

- digitar um nome;

- verificar entrada em *ranking.txt*.

Saída real observada:

Pedro 12

Ana 5

Victor 9
Status: aprovado.

CONCLUSÃO

Todos os testes realizados confirmam o funcionamento correto do sistema.