Projeto Final - Avaliação A3

Jogo Cappy Bird

Relatório técnico apresentado na UC Usabilidade, desenvolvimento web e jogos pelo Prof. MSc Flávio Henrique da Silva.

Curitiba

2025

INTEGRANTES DO GRUPO

Alexandre Viebrantz
RA 172420231

João Paulo Lins
RA

SUMÁRIO

1 INTRODUÇÃO

O tema do trabalho foi a elaboração de um jogo em Python, no qual a referência era o FlappyBird, utilizando a biblioteca Pygame e os conhecimentos repassados em sala de aula.

2 DESENVOLVIMENTO

O projeto foi desenvolvido com a base do jogo do FlappyBird.
Trazendo uma regionalidade ao jogo e ao trabalho, o player foi substituido por uma capivara, bem como a mudança de cenário, para que ela nadasse dentro de um lago, ao invés de voar como no FlappyBird.

2.1 Divisão das Tarefas

O João implementou a estrutura do jogo, seu core, com os arquivos main, game, canos e cappy. O Alexandre refinou detalhes do jogo, trazendo os gráficos, sons e telas quando o usuário perder ou ganhar e o menu.

2.2 Estrutura do Projeto

Este projeto foi desenvolvido utilizando o framework **Pygame** para criação de jogos 2D em Python. Também foram utilizadas as bibliotecas **OpenCV (cv2)** e **NumPy** para o controle e exibição de vídeos nas telas de vitória e derrota.

O projeto está organizado em diversos módulos que são os arquivos .py, cada um responsável por uma parte da lógica do jogo, como o personagem principal, obstáculos, lógica do jogo, telas de menu e instrução, entre outros.

Sobre as bibliotecas e frameworks utilizados:

pygame: Biblioteca utilizada em sala de aula e base para o jogo. Utilizada para renderização gráfica, manipulação de eventos e sons.

cv2 (OpenCV): Usado para leitura e exibição de vídeos nas telas finais.

numpy: Auxilia no tratamento de frames de vídeo.

random: Para gerar posições aleatórias dos obstáculos (troncos).

sys: Para controle de saída do programa.

Arquivos e Objetos desenvolvidos:

DRobj.py

Classe obj: Classe base para elementos visuais do jogo.

__init__: Inicializa o objeto com imagem e posição.

drawing: Desenha o objeto na tela.

updateposs: Atualiza a posição.

givethepoints: Retorna pontos.

colision: Detecta colisões com outros objetos.

cappy.py

Classe Cappy: Representa o personagem principal do jogo.

__init__: Inicializa a posição e imagem do personagem.

drawing: Desenha o personagem.

update: Atualiza a posição vertical (efeito da gravidade e pulo).

GetTheCappyNumber: Retorna dados do personagem.

canos.py

Classe Canos: Gerencia os obstáculos (troncos).

AddCano: Adiciona novos troncos.

get_points: Retorna os pontos obtidos.

removethecanos: Remove os troncos que saíram da tela.

drawtheCanos: Desenha todos os troncos.

updateTheCanos: Atualiza posições.

getCanoNcanos, getRectup, getRectdn: Funções auxiliares para acessar e verificar colisões.

game.py

Classe Game: Controla a lógica principal da fase do jogo.

write, draw, setMouse, setStatus, movebg: Controlam exibição gráfica e estados.

set_start: Inicia o jogo.

Cappypulo: Controla o pulo do personagem.

colision: Verifica as colisões.

restart: Reinicia o jogo.

update, status, events: Loop de jogo e eventos.

menu.py

Classe Menu: Tela inicial com opções de iniciar o jogo e o menu.

__init__, draw, status, events: Controlam o layout e as interações do menu.

instruction.py

Classe InstructionScreen: Tela com instruções do jogo.

__init__, draw, events, toggle: Exibe instruções e permite alternar visibilidade.

winner.py

Classe WinnerScreen: Tela quando o jogador vencer.

__init__: Inicializa o vídeo da vitória.

play_video: Reproduz vídeo usando OpenCV.

lose.py

Classe LoseScreen: Tela para quando o jogador perder.

__init__: Inicializa o vídeo da derrota.

play_video: Reproduz vídeo usando OpenCV.

main.py

Classe Main: Classe de controle principal do jogo.

__init__: Inicializa o jogo e os componentes.

updategamestatus, draw, quit, events, update, stop_game_sound: Controlam o fluxo geral do jogo e a alternância entre telas.

2.3 Explicação da Aplicação/Software

Ao iniciar o jogo, o usuário poderá ter duas ações, pressionar enter para iniciar o jogo e pressionar a tecla ‘m’ para acessar o menu.

Pressionando Enter, o jogo inicia e o usuário precisa apenas apertar a barra de espaço para poder nadar no jogo.

A barra de espaço fará com que a capivara suba desviando dos obstáculos mais baixos e a ausência de ação do jogador fará com que a capivara afunde, também desviando dos obstáculos do alto.

Tendo desviado de 10 objetos, o contador de pontos chegará em 10 e o jogador vencerá o jogo.

Ele poderá teclar enter para Jogar novamente.

Agora, caso na tela inicial ele tenha pressionado a tecla ‘m’ para acessar o menu, uma tela com instruções simples é exibida e o usuário pode clicar no botão com o X para fechar ou pressionar ESC para retornar ao menu do jogo.

2.4 Orientações de execução da Aplicação/Software

Para executar corretamente o jogo, é necessário seguir os passos abaixo:

1. Requisitos do Sistema

Antes de iniciar a execução do jogo, verifique de que o sistema possui os seguintes requisitos:

Python 3.8 ou superior instalado.

Sistema operacional compatível (Windows, Linux ou macOS).

Acesso a terminal ou prompt de comando.

2. Instalação das Dependências

O projeto depende das seguintes bibliotecas externas para poder funcionar:

pygame: biblioteca para desenvolvimento de jogos em Python.

opencv-python (cv2): para exibição de vídeos nas telas de vitória e derrota.

numpy: usada em conjunto com o OpenCV para manipulação de dados de imagem.

Para instalar todas as dependências, execute o comando abaixo no terminal:

pip install [nome-da-dependencia]

Você precisa instalar:
pygame, opencv-python e numpy

Observação: No caso de Macs, pode ser necessário substituir o trecho do comando pip por pip3

3. Execução do Jogo

Para iniciar o jogo, basta executar o arquivo main.py. No terminal, navegue até o diretório onde está localizado o projeto e digite:

python main.py ou pip3 main.py (no mac)

4. Navegação e Uso

Ao iniciar, o jogo exibe um menu principal.

E a partir do menu, é possível iniciar o jogo teclando ‘Enter’, acessar instruções teclando ‘M’ ou sair.

Durante o jogo, o personagem é controlado através do teclado, o jogador deverá pressionar a tecla de barra de espaço para a capivara nadar e desviar dos obstáculos.

Ao final da fase, é exibida uma tela de vitória ou de derrota, onde em ambas ele pode reiniciar para jogar novamente.

2.5 Repositório

https://github.com/JoaoPLins/cappybird

3 CONCLUSÃO

Desenvolver o projeto do CappyBird foi uma experiência muito enriquecedora, tanto do ponto de vista técnico quanto do criativo.

A construção permitiu colocar em prática os conceitos de usabilidade repassados em aula e a programação em python.

A utilização das bibliotecas trouxe mais possibilidades ao projeto, visto que nem todas as necessidades foram atendidas pelo pygame.

A proposta era trazer uma experiência divertida e fácil em jogar o CappyBird, trazendo uma proposta de um jogo genuinamente curitibano e com um personagem que estampa o orgulho da cidade.

Considerando a entrega, os desafios foram o conhecimento da linguagem, pelo menos por parte do Alexandre, a organização do projeto e o trabalho em equipe.

No fim das contas, com o jogo finalizado mostrou como diferentes alunos, com seus conhecimentos e utilização de diversas ferramentas podem ser combinadas para criar um projeto engrandecedor e que agregue no conhecimento dos alunos.

REFERÊNCIAS

Como referências, foram utilizados materiais elaborados em sala de aula, além de consultas em fóruns e Chat GPT.