1. Endereço do Repositório no GitHub

O repositório para o jogo Pong está disponível no GitHub: GitHub - josue-oliveiraa/ping_Pong_game: O Pong é uma recriação de um dos primeiros jogos eletrônicos já desenvolvidos, originalmente lançado pela Atari em 1972. Este projeto tem como objetivo não apenas reproduzir o jogo em sua essência, mas também explorar conceitos de computação, como algoritmos eficientes, interatividade e jogabilidade fluida.

Certifique-se de acessar o link para visualizar o código-fonte, documentação e instruções detalhadas.

2. Checklist do Desenvolvimento do Jogo

Definição do Tema

O tema do jogo é inspirado no clássico Pong, um simulador de tênis de mesa eletrônico. Dois jogadores controlam raquetes para rebater uma bola, competindo para marcar pontos.

Planejamento

• Etapas do Projeto:

1. Pesquisa: Revisão sobre o clássico Pong e tecnologias necessárias.
(2 dias): Matheus Henrique de Sousa 30228981

2. Design: Criação da interface simples com raquetes, bola e placar. (1 dia): Danilo Silva 28641205

3. Programação Base: Implementação das movimentações das raquetes e da bola. (3 dias): Josué Oliveira 29786932

4. Regras do Jogo: Adição de limites, pontuação e reinício de rodadas.
(2 dias): Matheus Henrique de Sousa 30228981

5. Testes e Ajustes: Testes para corrigir bugs e balancear jogabilidade.
(2 dias): Natan Martins 29677491

6. Documentação: Organização e descrição do código e processo. (1 dia): Victor Kardec de Mello 30223971

Desenvolvimento

O jogo foi desenvolvido seguindo o plano, com melhorias incrementais em cada

etapa.

Testes

Testes foram realizados para validar:

• Movimento das raquetes.

• Física da bola (rebotes e ângulos).

• Detecção de pontuação.

• Experiência fluida em diferentes velocidades.

Documentação

A documentação detalha:

• Estrutura do código e funções principais.

• Processo de desenvolvimento e decisões tomadas.

• Instruções para execução do jogo.

3. Tecnologias Utilizadas

• Linguagem de Programação: Python

• Biblioteca Gráfica: Pygame

• Ferramenta de Controle de Versão: Git

• Plataforma de Repositório: GitHub

4. Identificação da Complexidade do Jogo

Análise da Complexidade

• Movimento da Bola: Algoritmo para calcular ângulos de rebote baseado no ponto de colisão (complexidade constante, O(1)).

• Controle de Raquetes: Algoritmo simples de detecção de tecla pressionada para ajustar a posição das raquetes (complexidade constante, O(1)).

• Pontuação: Sistema que incrementa a pontuação quando a bola ultrapassa o limite de um dos lados (complexidade constante, O(1)).

Gerenciamento da Complexidade

A simplicidade do Pong foi mantida com algoritmos leves e eficientes. A lógica de

física foi otimizada para evitar cálculos excessivos, garantindo desempenho estável.

5. Regras (Jogabilidade)

Regras do Jogo

1. O jogo é para dois jogadores.

2. Cada jogador controla uma raquete com as seguintes teclas:

o Jogador 1: W (subir), S (descer).

o Jogador 2: ↑ (subir), ↓ (descer).

3. O objetivo é rebater a bola para o lado do adversário.

4. Um ponto é marcado quando a bola ultrapassa a borda oposta.

5. O jogo reinicia automaticamente após um ponto ser marcado.

6. O primeiro jogador a alcançar 10 pontos vence.

Instruções para Jogadores

• Execute o arquivo pong.py em Python.

• Use as teclas apropriadas para mover as raquetes.

• Divirta-se competindo contra seu amigo!

Checklist para o Projeto Pong em Computabilidade e Complexidade de Algoritmo

Fase 1: Análise

• Problema selecionado e definido claramente: Criar o clássico jogo Pong, simulando a interação de duas raquetes com uma bola em um plano bidimensional.

•Compreensão aprofundada da natureza e desafios do problema: O jogo requer movimentação fluida de objetos, detecção de colisões e gerenciamento de pontuações.

• Modelo matemático ou teórico desenvolvido para representar o problema:

o Representação das raquetes e bola como retângulos.

o Movimento da bola baseado em vetores (x, y).

o Detecção de colisão por interseção de áreas retangulares.

Fase 2: Planejamento

• Objetivos do algoritmo definidos com clareza:

o Implementar as regras básicas do Pong.

o Garantir fluidez e responsividade durante o jogo.

• Métricas para avaliação de eficiência do algoritmo estabelecidas:

o Tempo de resposta em movimentação e colisão.

o Uso eficiente de recursos computacionais.

• Estratégia geral de resolução do problema proposta:

o Divisão em subproblemas: movimentação da bola, detecção de colisão, movimentação das raquetes e gerenciamento de pontuação.

• Subproblemas identificados e divididos, se aplicável:

o Lógica de movimento das raquetes.

o Física da bola (rebotes e colisões).

o Sistema de pontuação.

• Estrutura geral do algoritmo esboçada:

o Loop principal para renderização e lógica de jogo.

o Funções específicas para cada subproblema.

• Casos limite ou situações especiais identificados:

o Quando a bola atinge as bordas do campo.

o Quando a bola passa pelas raquetes.

• Análise teórica realizada para verificar a correção do algoritmo:

o Colisões corretamente detectadas usando lógica baseada em coordenadas.

o Movimentação fluida gerenciada pelo loop principal com taxa de

quadros controlada.

Fase 3: Desenho

• Análise de complexidade realizada para avaliar a eficiência teórica do algoritmo:

o Detecção de colisões: O(1)O(1)O(1), pois verifica interseções apenas nos limites da bola e das raquetes.

o Movimentação de objetos: O(n)O(n)O(n), onde nnn é o número de objetos (neste caso, fixo em 3: bola e 2 raquetes).

• Pontos críticos do algoritmo identificados para otimização, se necessário:

o Lógica de colisão para evitar cálculos redundantes.

o Limitar o uso de recursos gráficos para garantir desempenho estável.

Fase 4: Programação e Teste

• Algoritmo traduzido com precisão em código de programação:

o Implementação feita em Python com Pygame.

• Código de programação escrito de forma clara e organizada:

o Uso de funções para modularização e clareza.

• Testes rigorosos realizados em uma variedade de casos de teste:

o Testes realizados para movimentação das raquetes e colisão da bola.

• Casos limite e situações especiais testados:

o Bola atingindo as bordas superiores e inferiores.

o Bola ultrapassando as raquetes para pontuação.

• Erros e problemas durante o teste de programa identificados e corrigidos:

o Ajustes no cálculo de colisão e velocidade da bola.

Documentação e Avaliação do Projeto

• Documentação completa, incluindo especificação do algoritmo e análise de complexidade:

o Arquivo README com explicação detalhada do projeto.

• Documentação revisada para clareza e rigor técnico:

o Código comentado e descrições detalhadas.

• Avaliação da eficácia do algoritmo em termos de tempo de execução, uso de recursos e precisão na resolução do problema:

o O algoritmo mantém taxa de quadros estável (60 FPS) em sistemas convencionais.

Apresentação e Conclusão do Projeto

• Apresentação do projeto preparada com informações claras e objetivas:

.O jogo e sua lógica foram apresentados no check-list e no vídeo explicativo.

• Conclusões do projeto destacando os resultados e aprendizados:

.O projeto demonstrou a aplicação prática de algoritmos de movimentação e colisão em jogos.