RELATÓRIO DO PROJETO - JOGOS CLÁSSICOS EM PROCESSING

Caio Cézar Salomão Andrade; Erick Sena Godinho; Henrique Sabino Sales

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais (IFMG), São João Evangelista-MG

caioandradesje@gmail.com@gmail.com
ericksena621@gmail.com
henriquesabsales912@gmail.com@gmail.com

Resumo: Este projeto consiste na implementação de três jogos clássicos utilizando a linguagem Processing: Pong, Snake (Cobrinha) e Asteroid. Todos os jogos foram desenvolvidos com recursos de áudio e gráficos simples, oferecendo uma experiência interativa completa. O projeto demonstra conceitos fundamentais de programação de jogos, incluindo detecção de colisão, controle de entrada do usuário, gerenciamento de estado e integração multimídia.

Palavras-chave: Processing, Java, Asteroid, Pong, Snake

INTRODUÇÃO

O objetivo deste projeto foi recriar dois jogos icônicos da história dos videogames utilizando Processing, uma linguagem de programação orientada a objetos baseada em Java, especialmente projetada para artistas e designers criarem aplicações visuais interativas.

Processing oferece uma sintaxe simplificada e um ambiente de desenvolvimento integrado que facilita a criação de projetos gráficos e interativos. A escolha desta plataforma permitiu focar nos aspectos fundamentais da lógica de jogos sem se preocupar com complexidades técnicas de baixo nível.

Os jogos selecionados - Pong, Snake e Asteroid - representam marcos importantes na evolução dos videogames e são ideais para demonstrar conceitos essenciais como:

- Programação orientada a objetos
- Detecção de colisão
- Controle de entrada do usuário
- Gerenciamento de estado do jogo
- Integração de áudio e gráficos

MATERIAL E MÉTODOS

Os materiais e métodos utilizados para o desenvolvimento desses jogos estão listados a seguir:

Tecnologias Utilizadas:

- Processing 3.x (linguagem de programação)
- Biblioteca processing.sound para efeitos sonoros
- Arquivos de áudio (formato MP3 e WAV)
- Imagens PNG para elementos gráficos

Estrutura do Projeto:

O projeto está organizado em três pastas principais:

1. Pong/
   - Pong.pde (arquivo principal)
   - Bola.pde (classe da bola)
   - Barra.pde (classe das barras/raquetes)
   - data/ (recursos multimídia)

2. Cobrinha/
   - cobrinha.pde (arquivo principal)
   - data/ (recursos de áudio)

3. AsteroidGame/
   - AsteroidGame.pde (arquivo principal)
   - Asteroid.pde (classe de asteroid)
   - AsteroidBuffer.pde (classe de que armazena os asteroids para melhor uso de memória)
   - Buffer.pde (classe de buffer base para ser reutilizada)
   - Bullet.pde (classe de balas)
   - BulletBuffer.pde (classe de que armazena as bullets para melhor uso de memória)
   - CollisionUtils.pde (classe que disponibiliza verificação de colisões)
   - Entity.pde (classe base para os objetos de jogo)
   - GameState.pde (enum que lista todos os estados do jogo
   - KeyboardListener.pde (classe que disponibiliza verificação de entrada de teclado)
   - Particle.pde (classe de partículas)
   - ParticleEffect.pde (interface base para efeitos de partícula)
   - ParticleEffects.pde (conjunto de classes de efeitos de partícula)
   - ParticleSystem.pde (classe de que armazena as partículas para melhor uso de memória)
   - Spaceship.pde (classe da nave)
   - sounds/ (recursos de áudio)
   - sprites/ (recursos de imagens)

Metodologia de Desenvolvimento:

O desenvolvimento do jogo **Pong** envolveu a implementação da física básica de movimento da bola, um sistema de colisão entre bola e barras, e um controle adaptativo das barras com base na velocidade da bola. Também foi implementado um sistema de pontuação e efeitos sonoros distintos para cada ação do jogo. Além disso, o carregamento de áudio foi tratado com mecanismos de prevenção de erro, garantindo a estabilidade do jogo mesmo na ausência dos recursos sonoros.

No jogo **Snake (Cobrinha)**, foi desenvolvido um sistema de movimento baseado em grade, permitindo o crescimento dinâmico da cobra a cada alimento consumido. O jogo conta com detecção de colisão tanto com as bordas da tela quanto com o próprio corpo da cobra. O sistema de pontuação é atualizado conforme o jogador progride, e a interface se adapta a diferentes resoluções. Efeitos sonoros foram incorporados para dar feedback às principais ações do jogador.

Já no jogo **Asteroid**, o jogador controla uma nave por meio de teclas direcionais e propulsão, podendo disparar projéteis com a barra de espaço. Os asteroides são gerados de forma aleatória e progressiva, com colisões implementadas entre a nave, os projéteis e os próprios asteroides. Asteroides maiores se dividem em menores após o impacto, e a pontuação é acumulada com base na destruição desses objetos. O jogo também apresenta efeitos visuais com partículas para simular explosões, enriquecendo a experiência gráfica.

Técnicas de Progtamação:

Foram aplicados diversos princípios e técnicas de programação para garantir modularidade, eficiência e escalabilidade nos três jogos desenvolvidos. A programação orientada a objetos foi amplamente utilizada, com a criação de classes separadas para representar diferentes elementos dos jogos, como bolas, barras, segmentos da cobra, naves, asteroides e projéteis. Houve encapsulamento adequado de propriedades e métodos, promovendo uma organização clara do código. Além disso, houve reutilização de código por meio de instâncias de classe, interfaces e o uso de *Generics*, tornando o sistema mais flexível e extensível.

A detecção de colisão foi cuidadosamente implementada, com métodos específicos para cálculo das bordas dos objetos e verificação de sobreposição entre eles. Essa detecção foi usada tanto para interações físicas (como rebotes e destruição de asteroides) quanto para eventos lógicos (como pontuação e fim de jogo), assegurando uma resposta adequada a cada tipo de colisão.

O controle de estado do jogo foi gerenciado com o uso de variáveis globais para acompanhar a pontuação e com a definição clara de diferentes estados, como “jogando” e “fim de jogo”. Mecanismos de reinicialização e reset foram incluídos, permitindo que o jogador recomece uma partida de forma fluida.

No que diz respeito ao tratamento de erros, foram utilizadas estruturas *try-catch* para garantir o carregamento seguro de recursos de áudio, evitando travamentos caso os arquivos estejam ausentes ou corrompidos. A função personalizada playSafely() foi criada para executar sons com segurança, permitindo que o jogo continue normalmente mesmo na ausência dos arquivos sonoros.

Por fim, foi adotada uma estratégia eficiente de reutilização de objetos em memória. Buffers foram utilizados para armazenar e gerenciar entidades como projéteis e partículas, aplicando o padrão conhecido como *object pooling*. Essa abordagem evita a criação e destruição constante de objetos a cada quadro, o que reduz significativamente o uso de memória e melhora o desempenho geral dos jogos.

RESULTADOS

No jogo Pong, foram implementadas funcionalidades como a física realista de movimento da bola, controle responsivo das barras (com as teclas A/Z para o jogador da esquerda e setas direcionais para o da direita), sistema de pontuação funcional e efeitos sonoros diferenciados, incluindo música de fundo contínua, sons específicos para colisões e um som especial para pontuação. A bola acelera progressivamente a cada rebatida, e seu ângulo de rebote varia conforme o ponto de impacto. A interface conta com um placar centralizado, modo tela cheia para melhor imersão, e a partida pode ser reiniciada manualmente com um clique do mouse. Erros relacionados ao áudio são tratados de forma robusta. Do ponto de vista técnico, o jogo possui resolução adaptativa, mantém a taxa de quadros padrão do Processing, adota controles intuitivos e apresenta um balanceamento dinâmico com base na direção da bola.

Já o jogo Snake (Cobrinha) traz um sistema de movimento em grade, com controle direcional pelas setas, crescimento da cobra ao consumir comida, detecção de colisões com bordas e consigo mesma, e geração aleatória de alimentos. O sistema de pontuação é progressivo, com efeitos sonoros associados ao ato de comer e ao fim de jogo. A interface se adapta a diferentes resoluções e apresenta uma tela de fim de jogo com instruções para reinício, que ocorre via tecla ‘R’. A taxa de quadros foi ajustada para proporcionar uma jogabilidade mais fluida. Em termos técnicos, o jogo utiliza um sistema de escala automática com cálculo de offset para centralização, além de um ArrayList para gerenciar os segmentos da cobra. Também há mecanismos de prevenção contra movimentos inválidos, como a reversão imediata de direção.

O jogo Asteroids apresenta um controle completo da nave por meio das teclas direcionais, incluindo propulsão e disparo de projéteis com a barra de espaço. Os asteroides são gerados de forma aleatória e crescente, com colisões implementadas entre nave, projéteis e os próprios asteroides. Asteroides maiores se dividem em menores após o impacto, e efeitos visuais com partículas simulam explosões. O sistema de pontuação baseia-se na destruição dos asteroides, com efeitos sonoros integrados, como o som do disparo e das explosões. A interface é responsiva, com um HUD minimalista, sistema de vidas e reinício automático ao fim do jogo. Tecnicamente, o jogo foi estruturado com múltiplas classes (como Spaceship, Asteroid, Bullet, Particle, Entity, entre outras), fazendo uso extensivo de herança e polimorfismo. Utiliza buffers para gerenciamento eficiente de entidades ativas (como BulletsBuffer e AsteroidsBuffer) e reaproveitamento de objetos através da técnica de *object pooling*, otimizando o desempenho e consumo de memória. As colisões são tratadas por um sistema genérico (CollisionUtils) e o estado do jogo é gerenciado por meio da estrutura GameState. Também foi implementado um sistema modular de efeitos de partículas e utilizados recursos externos, como arquivos .mp3 e sprites .png, com organização em arquivos “.pde” distintos.

DISCUSSÃO

A análise comparativa mostra que cada jogo se destacou em diferentes áreas da programação de jogos. O Pong priorizou a física contínua de movimento, interação precisa entre objetos, controle de timing e balanceamento de dificuldade. O Snake focou em lógica de movimento discreto, uso de estruturas dinâmicas, controle de estado e construção de uma interface informativa. Já o Asteroids foi o mais complexo, com múltiplas entidades interativas, gerenciamento de vários estados simultâneos, aplicação de técnicas avançadas de desempenho e colisões mais sofisticadas.

Em relação à qualidade do código, os três jogos apresentam uma estrutura modular, com separação clara de responsabilidades entre classes, comentários explicativos nos pontos críticos, tratamento adequado de exceções, uso consistente de convenções de nomenclatura e reaproveitamento eficiente de código.

Por fim, a experiência do usuário foi cuidadosamente planejada, com controles intuitivos e responsivos, feedbacks visuais e sonoros bem integrados, instruções claras para reinicialização e performance estável mesmo em diferentes dispositivos.

CONCLUSÃO

O projeto alcançou seus objetivos com êxito ao recriar três jogos clássicos, Pong, Snake e Asteroids, mantendo alta fidelidade às versões originais, ao mesmo tempo em que incorporou melhorias modernas, como suporte a áudio, efeitos visuais e interface responsiva. A implementação evidencia o domínio dos principais conceitos de programação de jogos, como detecção de colisão, controle de estado, reaproveitamento de objetos e estrutura orientada a objetos, além do uso eficaz da plataforma Processing para desenvolvimento interativo.

Entre as principais conquistas do projeto, destacam-se a implementação completa de três experiências de jogo distintas, a organização clara e bem estruturada do código-fonte, o tratamento robusto de possíveis erros de execução (em especial no carregamento de mídias), o cuidado com a interface e a experiência do usuário, bem como a integração de elementos multimídia, como sons e animações, que enriquecem a imersão durante o jogo.

Dessa forma, o projeto não apenas cumpre sua função como um exercício prático de desenvolvimento, mas também demonstra o potencial do Processing como uma ferramenta acessível e poderosa para criação de jogos simples. Além disso, estabelece uma base sólida para a expansão futura do trabalho, servindo como ponto de partida para projetos mais complexos e sofisticados dentro da área de desenvolvimento de jogos digitais.