CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Trabalho Prático:

DOTS AND BOXES

André Victor Gonçalves Nascimento

Pietro Campos de Sousa

Tiago Helvécio de Mello Rosa

Contagem

2025

Sumário

1. Introdução

Este relatório apresenta o desenvolvimento de uma versão do jogo "Pontos e Caixas" utilizando a biblioteca LibGDX em Java. O objetivo foi aplicar conceitos de orientação a objetos e recursos multimídia (imagens e sons) para criar uma interface interativa com detecção de cliques e regras de jogo adequadas.

O jogo foi desenvolvido por meio de um programa em classes, no qual cada classe representa um objeto da tela ou atuante no jogo e segue as regras básicas do jogo “Pontos e Caixas”. Dentre as regras estão um tabuleiro 6x6, opções de selecionar linhas até formarem quadrados, o jogador e o bot devem jogar em turnos – sendo que se um completar um quadrado, pode jogar novamente – o jogo termina assim que todos os quadrados são selecionados e o jogador tem a opção de reiniciar a partida sem precisar fechar o programa. Por fim, o programa não pôde conter variáveis globais, não devia conter funções com mais de 30 linhas e deve apresentar elementos de imagem e som.

2. Metodologia e Estratégias de Desenvolvimento

Foram adotadas as seguintes estratégias:

- **Divisão em Módulos:** O programa foi dividido em classes (Dots, Lines, Square, Text) que implementam a interface Drawables para padronizar métodos de desenho e descarte de recursos, demais classes para o funcionamento do jogo (Bot, Main, SquareEnemy, SquarePlayer) e um enum de telas para indicar se está no menu, no jogo ou na tela final (Screen).
- **Estrutura de Dados:** Linhas, pontos e quadrados são armazenados em ArrayList, com vetores boolean[] respectivos para cada linha e quadrado, para controle de estado das mesmas, se estão clicadas, sobrepostas pelo mouse e se quadrados estão concluídos .
- **Detecção de Colisões:** Uso de Polygon para verificar se o cursor está sobre uma linha, analisando as coordenadas do cursor em relação à tela do jogo.
- **Regras de Jogo:** Implementação das regras de turno, pontuação e detecção de quadrados completos.
- **Iteração Contínua:** Testes incrementais a cada funcionalidade implementada, seguindo boas práticas de programação.

As principais ferramentas utilizadas foram:

- **Visual Studio Code:** Para codificar o jogo;

- **GitHub:** Para documentar as alterações feitas no projeto por cada participante;
- **Hiero:** Para converter fontes do formato .ttf para .fnt.

3. Arquitetura do Sistema

3.1 Classe Dots

- **Responsabilidade:** Carregar e desenhar 36 pontos de conexão.
- **Atributos:** ArrayList<Texture> dots.
- **Métodos:** Construtor (carrega texturas), draw(SpriteBatch), dispose().

3.2 Classe Lines

- **Responsabilidade:** Gerenciar sprites de linhas, detectar hover/click, verificar formação de quadrados e gerenciar turnos.
- **Atributos Principais:** Texture lineTexture, ArrayList<Sprite> sprites, ArrayList<Polygon> polygons, boolean[] isHovering, boolean[] isClicked, Map<Integer, int[]> squareIndexMapColumn, Map<Integer, int[]> squareIndexMapRow, int playerPoints e int enemyPoints.
- **Módulos Internos:** Métodos para posicionamento (checkIfIsUp), rotação de sprites verticais, detecção de colisão (generatePolygon, checkIfMouseIsHovering e changeColor) e verificação de quadrados (checkSquare, checkSquaresWithRowLines, checkSquaresWithColumnLines).

3.3 Classe Square

- **Responsabilidade:** Representar graficamente um quadrado completado.
- **Atributos:** Texture square, Sprite sprite.
- **Métodos:** Construtor (String texturePath), setPosition(float x, float y), draw(SpriteBatch), dispose().

3.4 Classe Text

- **Responsabilidade:** Exibir textos na tela (título e instruções) com animação de fade.
- **Atributos:** BitmapFont font, CharSequence text, float x, float y, boolean canFade.
- **Métodos:** Construtores sobrecarregados, draw(SpriteBatch), dispose(), animateText().

3.5 Classe bot

- **Responsabilidade:** Executar jogadas no turno contrário ao do jogador.
- **Atributos:** lines, squaresEnemy, squaresPlayer, random, turn
- **Métodos:** Construtor de Bot, playTurn(), checkCompletedSquares(int lineIndex)

4. Relações entre as Classes

- A classe principal de jogo instancia Dots, Lines, Text, SquareEnemy, SquarePlayer e Bot para compor a cena.
- Lines checa os quadrados corretos para serem ativados.
- Bot decide qual linha será selecionada pela máquina.
- Dots, Lines, Text e Square implementam a interface Drawables, SquarePlayer e SquareEnemy herdam a classe Square e as classes Main e Text utilizam o enum Screen para verificar qual é a cena atual do jogo.

5. Conclusão

Ao final deste projeto, foi possível desenvolver um jogo onde o jogador pode interagir com o tabuleiro e com o *bot* de forma responsiva, seguindo as regras definidas conforme as orientações e dando ao jogador a possibilidade de jogar novamente. O código possui elementos de herança de classes, como as classes SquarePlayer e SquareEnemy que herdam a classe Square, além do uso da interface Drawables para facilitar o uso dos métodos dispose() e draw(). Não foram utilizadas variáveis globais, o jogo possui elementos de imagem e som, como foi solicitado e foi completamente documentado no GitHub.

Infelizmente, as funções checkSquareWithColumnLines, switchBoxCheckCasesColumn, checkSquaresWithRowLines e switchBoxCheckCasesRow da classe Lines possuem mais de 30 linhas, tornando-as grandes, porém nenhuma chega ao número de 50 linhas. Muitas funções conseguiram ser reduzidas com êxito para atender os requisitos de tamanho, porém não foi possível reduzir mais estas citadas, tornando este o único débito do nosso trabalho.

6. Referências

libGDX. libGDX. Disponível em: https://libgdx.com/. Acesso em: 15 abril 2025

AGAME. Dots and Boxes. Disponível em: https://www.agame.com/game/dots-and-boxes. Acesso em: 15 abril 2025.

libGDX. Source & Documentation. Disponível em: https://libgdx.com/dev/. Acesso em: 15 abril 2025.

libGDX. Wiki. Disponível em: https://libgdx.com/wiki/. Acesso em: 16 abril 2025.

STACK OVERFLOW. How to convert .ttf to .fnt with FontForge? Disponível em: https://stackoverflow.com/questions/24529369/how-to-convert-ttf-to-fnt-with-fontforge. Acesso em: 21 abril 2025.

STACK OVERFLOW. How can I draw text using libGDX Java? Disponível em: https://stackoverflow.com/questions/12466385/how-can-i-draw-text-using-libgdx-java. Acesso em: 21 abril 2025.

GAMEDEV STACK EXCHANGE. libGDX: Get mouse position in entire game. Disponível em: https://gamedev.stackexchange.com/questions/90124/libgdx-get-mouse-position-in-entire-game. Acesso em: 26 abril 2025.

STACK OVERFLOW. libGDX: How would I go about checking if a mouse is hovering over a TextField? Disponível em: https://stackoverflow.com/questions/35471809/libgdx-how-would-i-go-about-checking-if-a-mouse-is-hovering-over-a-textfield. Acesso em: 26 abril 2025.

LIBGDX. Rendering shapes. Disponível em: https://libgdx.com/wiki/graphics/opengl-utils/rendering-shapes. Acesso em: 27 abril 2025.

STACK OVERFLOW. How can I rotate rectangles in libGDX? Disponível em: **https://stackoverflow.com/questions/30554629/how-can-i-rotate-rectangles-in-libgdx.** Acesso em: 27 abril 2025.

JVM GAMING. Changing button color on mouse hover – LibGDX. Disponível em: https://jvm-gaming.org/t/changing-button-color-on-mouse-hover-libgdx/57925. Acesso em: 27 abril 2025.

HAPPY CODING. Multiple Game Screens. Disponível em: https://happycoding.io/tutorials/libgdx/game-screens. Acesso em: 03 maio 2025.

STACK OVERFLOW. Returning the index number of an ArrayList in Java. Disponível em: https://stackoverflow.com/questions/4605388/returning-the-index-number-of-an-arraylist-in-java. Acesso em: 14 maio 2025.

STACK OVERFLOW. What is the use of Map.ofEntries instead of Map.of? Disponível em: https://stackoverflow.com/questions/46601959/what-is-the-use-of-map-ofentries-instead-of-map-of. Acesso em: 27 maio 2025.