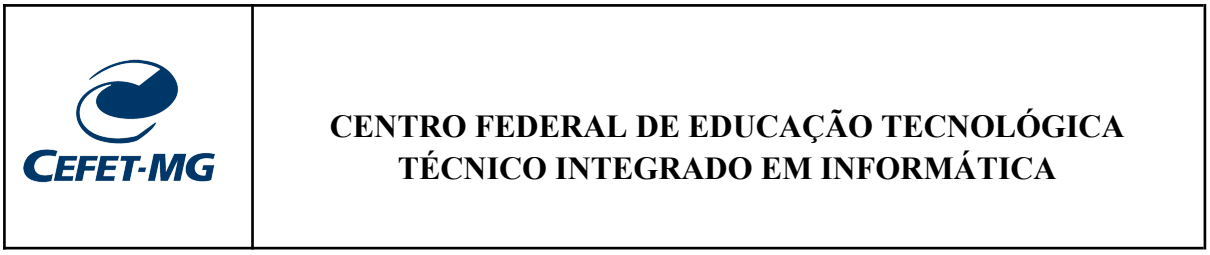

GRUPO 10:

GUILHERME ALCÂNTARA DE SOUZA
HEITOR ARAÚJO SIQUEIRA

RELATÓRIO TÉCNICO REFERENTE AO TRABALHO

Trabalho Prático 2- TP - Scrooling Game - ActRaiser (Modo Cidade)

PROFESSOR: ALISSON RODRIGO DOS SANTOS

CONTAGEM, 2025

SUMÁRIO

1) INTRODUÇÃO

Este trabalho busca colocar em prática os conceitos da programação orientada a objetos juntamente ao manuseio de recursos gráficos através da biblioteca LibGDX, aprendidos em sala de aula. Por meio da linguagem Java, serão implementados o padrão de projeto “Builder” e os pilares da POO, sendo eles a abstração, o encapsulamento, a herança e o poliformismo.

O objetivo central deste projeto é desenvolver uma versão digital do jogo "ActRaiser" modo cidade. Nesta recriação, o jogador, através de seu anjo, orienta os habitantes a construir casas, igrejas e estradas, expandindo a área habitável da cidade. Além disso, deve proteger a cidade de ataques de monstros, selando seus covis e usando milagres para expulsar ameaças.

Podem ser citados como requisitos do projeto:

- Trabalhar com a programação orientada de objetos;
- Implementar o padrão de projeto indicado nas instruções (Builder);
- Explorar a biblioteca LibGDX e suas ferramentas disponíveis;
- Implementar um sistema de jogabilidade interativa com elementos gráficos (cores e formas);
- Utilizar estruturas modulares em classes;
- Colocar em prática os conhecimentos obtidos em sala de aula.

O trabalho seguirá os objetivos estabelecidos na introdução, favorecendo na organização e direção do jogo.

2) IMPLEMENTAÇÃO

O jogo foi programado em Java utilizando a biblioteca gráfica LibGDX e o padrão de projeto “Builder”. Dessa forma, abaixo, as principais classes e interfaces que fazem parte do projeto podem ser observadas.

2.1.1. Classe Director

Oriunda do padrão de projeto, é a classe responsável por gerenciar a construção dos diferentes objetos complexos do jogo, ou seja, aqueles com muitos argumentos no construtor (Player, Morcego, Diabinho, Dragão). Através de um objeto de outra classe que implemente a

interface ObjetosBuilder, a Director define os valores iniciais (posição, velocidade, textura, hitbox e player) desses objetos de forma padronizada e organizada, facilitando a leitura do código.

2.1.2. Classe GameRunner

É a classe principal do jogo, estendendo Game da LibGDX. Ela configura os recursos iniciais, como SpriteBatch, BitmapFont e FitViewport, e define a tela inicial do jogo como StartMenu. Também gerencia o ciclo de vida do jogo com os métodos create, render e dispose.

2.1.3. Interface ObjetosBuilder

Define um contrato para construir objetos do jogo, seguindo o padrão Builder. Ela declara a assinatura dos métodos para configurar propriedades como posição, textura, hitbox, velocidade, pontos de vida, magia, dano e o jogador (Player). Classes que implementam essa interface devem fornecer a lógica para configurar essas características nos objetos do jogo.

2.1.4. Classe ObjetoDoJogo

A classe ObjetoDoJogo é uma classe abstrata que define a estrutura básica para qualquer objeto do jogo, como personagens, inimigos e itens. Ela armazena a posição (posição X, posição Y), a textura (Texture, que representa a imagem do objeto) e a hitbox (Rectangle, usada para detectar colisões). Além disso, fornece métodos de acesso (getters) para esses atributos. Essa classe serve como superclasse para outras mais específicas, permitindo reutilização e organização do código.

2.1.5. Classe Movel

Classe abstrata, representa um objeto do jogo que pode se mover, como inimigos ou o jogador. Ela estende ObjetoDoJogo e adiciona o atributo velocidade, além de dois métodos abstratos.

2.1.6. Classe Personagem

A classe abstrata Personagem representa qualquer personagem do jogo, como heróis ou inimigos, e estende a classe Movel. Ela acrescenta atributos específicos de personagens: pontos de vida, magia e dano. Além dos métodos de acesso (getters), define métodos abstratos para modificar esses atributos, como adicionar ou remover vida, magia e dano. Assim, ela serve como base para a criação de personagens com comportamentos personalizados, forçando subclasses a implementar a lógica desses métodos.

2.1.7. Classe PlayerBuilder

Exemplo de uma classe que implementa ObjetosBuilder. Ela cumpre o "contrato" da interface e possui o método buildPlayer(), que retorna um objeto Player, sendo parte essencial do padrão de projeto. É através de classes como essa que o processo criacional de objetos se torna mais fácil de ler e organizar.

2.1.8. Classe Player

A classe Player representa o personagem controlado pelo jogador e estende a classe abstrata Personagem, herdando atributos como posição, velocidade, textura, hitbox, vida, magia e dano. Ela implementa o método mover(), que usa as teclas W, A, S e D para movimentar o personagem nas direções: cima, esquerda, baixo e direita, respectivamente, de acordo com a velocidade e o tempo entre os quadros (deltaTime). Essa classe conecta a entrada do usuário com o movimento do personagem no jogo.

2.1.9. Classe Fase

A classe Fase é abstrata e implementa a interface Screen da LibGDX, servindo como base para as fases do jogo. Ela, assim como suas filhas, recebe o objeto da GameRunner e utiliza seus campos e métodos para gerenciar os eventos e exibir elementos gráficos e sonoros.

2.2. Funcionamento das Classes do Jogo

O jogo desenvolvido com libGDX é estruturado de forma organizada e modular, seguindo princípios de orientação a objetos e padrões de projeto como o Builder. A classe principal GameRunner gerencia o ciclo de vida do jogo e inicializa os recursos gráficos

básicos, como SpriteBatch e BitmapFont, além de definir a tela inicial, que é o StartMenu. Esta tela apresenta um botão que, e ao ser clicado, leva o jogador para um breve tutorial e, logo após, para a PrimeiraFase, onde o jogo propriamente dito ocorre.

Na PrimeiraFase e SegundaFase, será possível renderizar os elementos do jogo e controlar o personagem principal, representado pela classe Player. Essa classe estende Personagem, que por sua vez herda de Movel e ObjetoDoJogo. Essa hierarquia define propriedades comuns como posição, textura e hitbox, e depois adiciona comportamentos móveis (com velocidade e método mover) e atributos de personagem (vida, magia e dano). O Player implementa o método mover para responder aos comandos do teclado, permitindo que o jogador controle o personagem com as teclas W, A, S e D.

A criação do jogador e de outros personagens no jogo é feita usando o padrão Builder, por meio da interface ObjetosBuilder e de classes como PlayerBuilder. A classe Director centraliza a lógica de construção dos objetos do jogo, chamando métodos do builder para definir atributos como posição, velocidade, textura e hitbox. Isso permite instanciar personagens complexos de forma padronizada e reutilizável, como Player, Morcego, Diabinho e Dragão.

Assim, o jogo é construído sobre uma base sólida, com separação clara entre lógica de construção, renderização e controle de entidades. Cada classe tem uma responsabilidade bem definida, facilitando a expansão e manutenção do jogo — como adicionar novos inimigos, implementar colisões, ou tratar pontos de vida e magia — aproveitando a estrutura já existente.

3) ESTRATÉGIAS DE DESENVOLVIMENTO

3.1. Criação das Classes: Criou-se classes com nomes intuitivos para facilitar a organização e entendimento do código. Cada classe possui suas funções e atributos, permitindo um fácil monitoramento de erros.

3.2. Padrão de Projeto: Estruturou-se o código utilizando o padrão “builder” desde o início de sua execução. Tal implementação permitiu facilitar a criação de objetos complexos do jogo, como personagens e inimigos, de forma modular e reutilizável.

3.3. Orientação a Objetos: Sendo um dos requisitos do trabalho, buscou-se aplicar de forma eficiente para garantir reutilização e modularidade. Por exemplo, o encapsulamento permite o melhor controle dos atributos de cada elemento e a composição ajudou na estruturação do jogo.

4) RESULTADOS OBTIDOS

Ao final do desenvolvimento do jogo, atendeu-se os requisitos propostos e exigências definidas para o projeto na introdução. A biblioteca LibGDX permitiu a criação de interface gráfica que transformaram a experiência e a jogabilidade, entretanto sem perder a simplicidade e o objetivo estabelecido no jogo base.

Os resultados obtidos após a codificação mostram uma estrutura funcional e bem organizada, com destaque para a jogabilidade, o controle do personagem e a mecânica de batalha contra os inimigos. São exemplos disso a animação dos personagens, a movimentação fluida do jogador com as teclas W, A, S e D e a tela inicial (StartMenu) com o botão “Jogar” também contribui para uma experiência de usuário simples e intuitiva, funcionando como ponto de entrada para o jogo.

Entretanto, apesar de uma jogabilidade semelhante a do jogo original, algumas dificuldades foram encontradas. Não foi possível realizar a construção dos edifícios pelo personagem, métodos utilizando o próprio arquivo “.tmx” do mapa foram testados a fim de interagir com o personagem, entretanto, não obteve-se sucesso. Além disso, a função do atributo “magia” presente na gameplay inicial foi adaptada para a versão do grupo, agora, o jogador troca tais pontos por vidas extras.

Para acessar o projeto da versão realizada, o link a seguir pode ser acessado: https://github.com/Heitorarj/TP2-Grupo10.

Imagem 1: ActRaiser (modo cidade) em funcionamento

CONCLUSÃO

O desenvolvimento do jogo “ActRaiser” utilizando Java e a biblioteca LibGDX representou uma oportunidade prática de aplicar os principais conceitos de programação orientada a objetos e arquitetura de software vistos em aula. Durante o processo, foi necessário muito mais do que apenas codificar: foi fundamental planejar a estrutura do código, separar responsabilidades entre classes, aplicar padrões de projeto como o Builder e pensar na interação do jogador com o sistema. Essa organização contribuiu diretamente para a clareza, manutenção e expansão do código, busca de resoluções úteis para as dificuldades encontradas, além de permitir uma base sólida para futuras funcionalidades, como colisões, inimigos inteligentes e efeitos visuais.

A utilização dos recursos gráficos da LibGDX deu vida ao projeto e permitiu a construção de uma experiência de jogo fluida, com controle direto do personagem e elementos visuais interativos. Embora alguns métodos ainda estejam pendentes de implementação, os objetivos principais foram alcançados: estrutura básica do jogo, movimentação do personagem, tela de menu funcional e adaptações criativas. Assim, este projeto se mostrou essencial para o desenvolvimento de habilidades técnicas e criativas no contexto do desenvolvimento de jogos, oferecendo uma base concreta para enfrentar desafios mais complexos na área da tecnologia.

REFERÊNCIAS

- **THE SPRITES RESOURCE**. *ActRaiser - SNES*. Disponível em: https://www.spriters-resource.com/snes/actraiser. Acesso em: 5 jul. 2025.

- **PIXABAY.** Disponível em: https://pixabay.com/pt/sound-effects/search/game/. Acesso em: 20 jul. 2025.

- **FREESOUND.** *Freesound*. Disponível em: https://freesound.org/. Acesso em: 23 jul. 2025.

- **YOUTUBE.** *Tiled Map Editor Tutorial Part One: The Basics*. Disponível em: https://youtu.be/ZwaomOYGuYo?si=52FmDi3E2LAZD1NV. Acesso em: 15 jul. 2025.

- **YOUTUBE**. *Builder Teoria - Padrões de Projeto - Parte 6/45*. Disponível em: https://youtu.be/2VwLvwsIu-8?si=h-9_5Wd7GHPAK7xT. Acesso em: 5 jul 2025.

- **HOLLOWBIT.** *LibGDX 2D Tutorials*: [playlist de vídeos]. YouTube. Disponível em: https://youtube.com/playlist?list=PLrnO5Pu2zAHKAIjRtTLAXtZKMSA6JWnmf&s. Acesso em: 10 jul. 2025.

- **REFATORING GURU.** *Padrão de projeto Builder*. Disponível em: https://refactoring.guru/pt-br/design-patterns/builder. Acesso em: 23 jul. 2025.