**Projeto LPII**

**Alunos: Canoi Gomes e Victor Hugo de Oliveira**

**Introdução:**

A ideia do projeto foi basicamente usar a uma das estruturas vistas em EDB, no caso a árvore binária de busca, para controlar a navegação do jogador pelas fases. Cada nó da árvore tem guardado em si uma fase aleatória, e a partir disso, posso fazer o jogador passar de fase simplesmente dizendo para qual nó da árvore ele deve ir agora, e como no caso essa árvore tem um nó que aponta para o pai, posso fazer a volta para a fase anterior possível. O objetivo final do jogo é navegar pelas fases, procurando por um baú que se encontra em alguma das folhas da árvore.

**Tecnologias e Estruturas Utilizadas:**

Para esse projeto foi usado a linguagem de programação Java, e a biblioteca de interface gráfica Swing.

Só conseguimos usar uma única estrutura, que foi uma árvore binária de busca simples, foi usada para gerenciar as fases.

**Organização:**

O projeto está dividido em Classes de GameObject, que são os objetos presentes no jogo (Inimigos, Player, Portas), GameState, que são as fases do jogo, ou seus gerenciadores(MenuState, LevelState, EditorState), Map, que tem duas classes, uma que serve para o Background, e outra que serve para carregar os dados referentes de cada fase de um TXT. Existe também uma classe Camera, que como o nome já diz, é a classe que representa nossa câmera do jogo.

**Orientação a Objetos:**

A Orientação a Objetos está presente em boa parte do projeto. Todo objeto dentro jogo é uma GameObject, então é lógico que todos eles herdam da classe GameObject. Além disso, tem a classe Enemy, que herda de GameObject, sendo que tem as classes Slime e Zombie, que também são inimigos, logo, ambas herdam de Enemy. E graças ao polimorfismo, pude criar uma lista de Enemy, que representa todos os inimigos de uma determinada fase, porém dentro da lista, tenho objetos do tipo Slime e Zombie.

A Orientação a Objetos também está presente nas classes de States. GameState é uma classe abstrata, e todas as classes de State, com exceção das de gerenciamento(AllStates e GameStateManager), herdam dela. O polimorfismo também foi essencial aqui, pois para fazer a árvore binária, precisei guardar um GameState dentro de cada nó.

**Design Pattern:**

No projeto foi usado somente o padrão Singleton em algumas classes, 3 classes, para ser mais exato, são elas Camera, GameStateManager e AllStates. O uso desse pattern nessa 3 classes tem a mesma explicação, só era necessária uma única instância de cada um deles em todo o programa. A Camera é usada em todas as fases, e ela sempre segue o jogador, então não era necessário uma para cada, basta uma sempre seguindo o jogador. O GameStateManager seria o nosso gerenciador de fases, ele que diz qual é a fase atual, também faz a troca de fases, e é ela que guarda os algoritmos necessários para fazer alterações na árvore, novamente, só posso ter essa classe instanciada uma única vez em todo o programa, se não teríamos que ficar passando ela como parâmetro para cada classe que fosse necessária o uso.

Já a classe AllStates é usada somente uma vez, no início do programa, nela eu tenho uma lista de todas as fases que eu quero que sejam adicionadas na árvore posteriormente.

**Tratamento de Erros:**

Foi criado somente uma classe que trata valores nulos, elas é usada somente em classes que valores assim não podem ser passados como parâmetro em algum método.