FUNDAÇÃO UNIVERSIDADE FEDERAL DO ABC

KAREN OLIVEIRA DOS SANTOS RA: 11110915
NATHAN CHARLES SANTOS RA: 11073415

PROJETO DA DISCIPLINA PROGRAMAÇÃO ORIENTADA A OBJETO: CODE'N'DRAGONS

SANTO ANDRÉ - SP

2018

Introdução

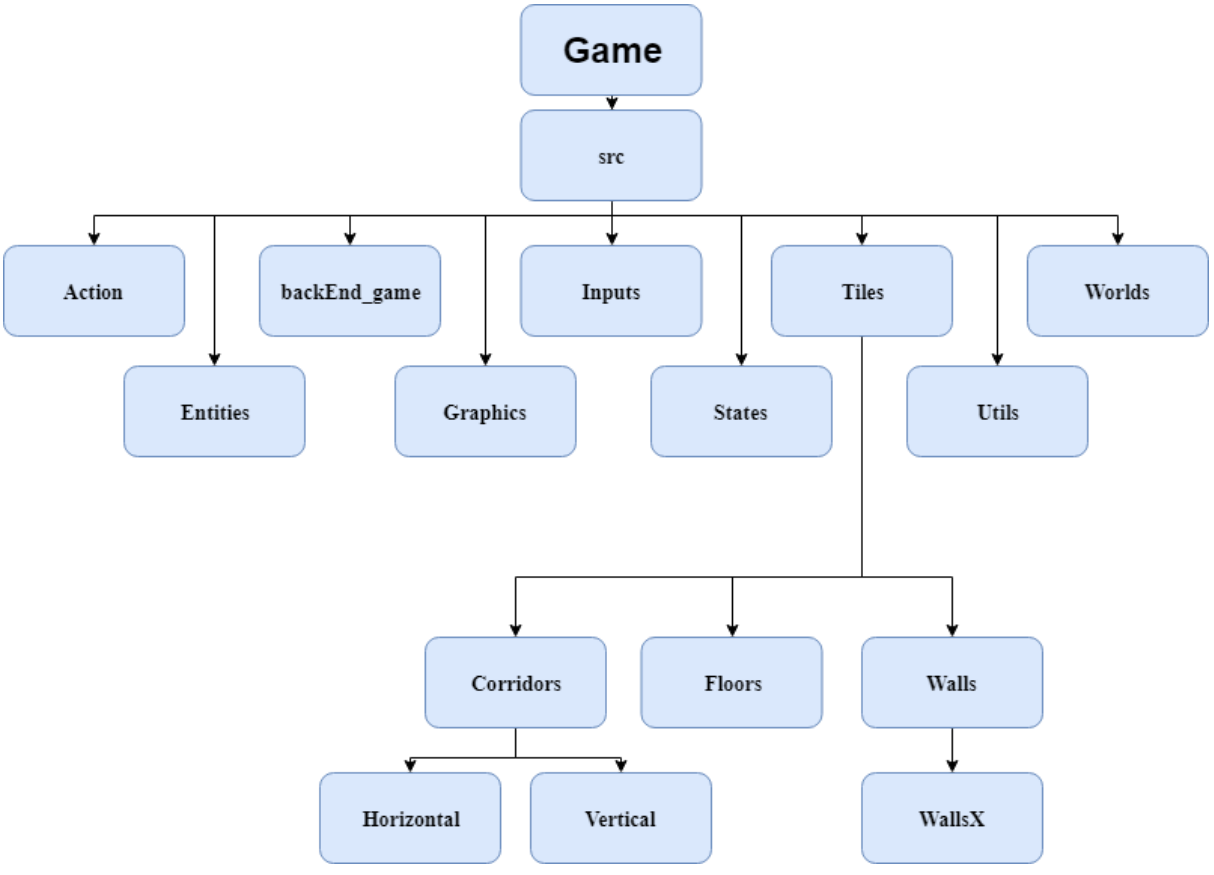
Tendo inspirações em séries como: Final Fantasy, World of Warcraft e The Legend of Zelda, realizamos a criação do jogo tendo em foco a estrutura de programação do mesmo, apesar da concepção mais próximo de uma demonstração, o jogo foi programado tendo a idéia que poderia ser realizado expansões.

Descrição do Programa

Criação da estrutura de um jogo do gênero RPG, utilizando conceitos da aula como encapsulamento de funções, herança e threads.

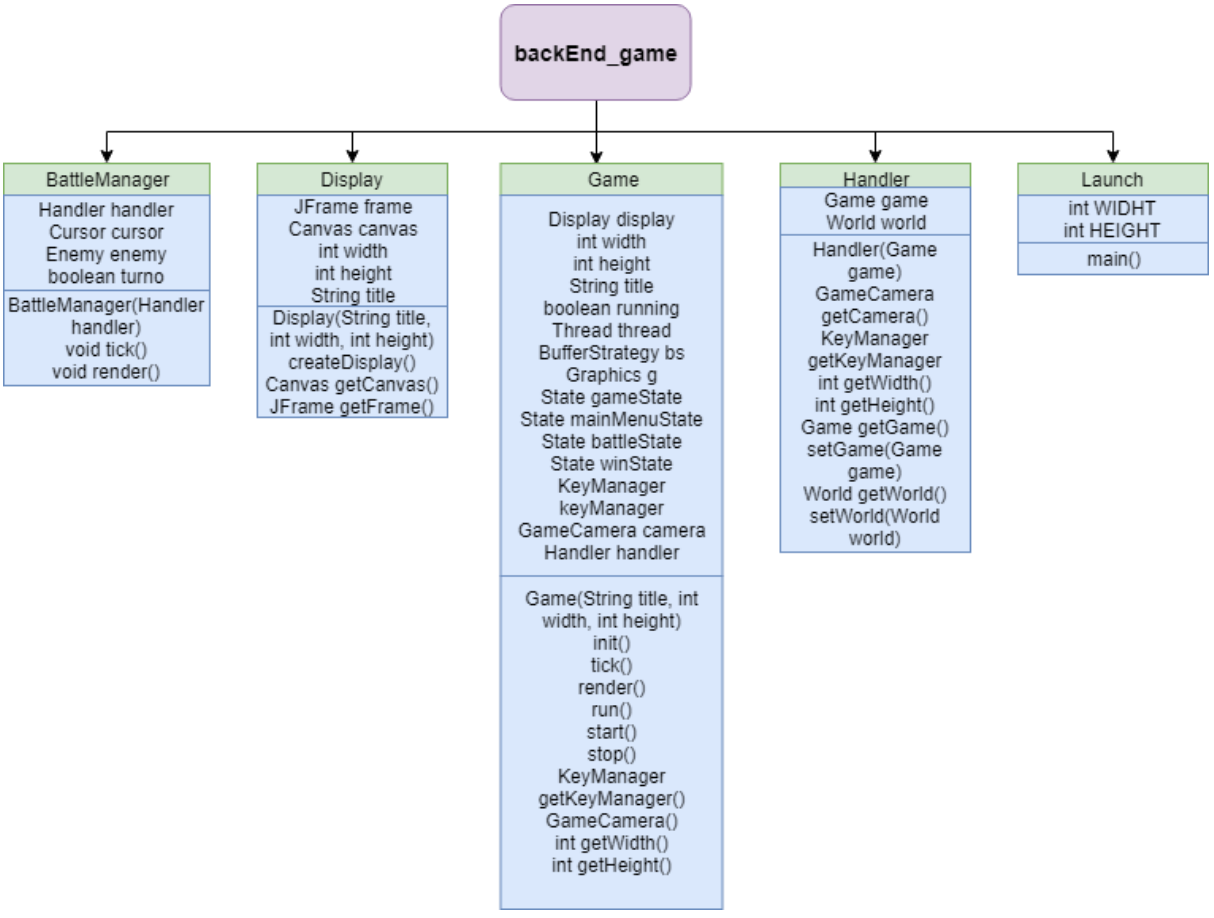
Diagrama de Pacotes

Devido a quantidade de Classes envolvidas no projeto, a documentação será feita tendo em base o diagrama de Pacotes para organização e facilitação de entendimento de classes, métodos e afins.



Package backEnd_game

Esse é um package que tem informações vitais para o funcionamento de suma importância para entendimento da funcionalidade do código devido a conter classes que gerenciam o funcionamento, logo seria lógico começar a explicação por este package.



Classe Display

Essa classe instancia e configura o JFrame e a Canvas, onde será desenhado os gráficos do jogo.

Classe Game

Essa classe instancia os objetos principais de controle do jogo como KeyManager, GameCamera e Thread reponsável pelo loop central do jogo, que consiste de dois métodos principais tick() que é responsável pela atualização das variáveis e gráficos do jogo e render() que é o responsável por renderizar os gráficos na Canvas que vai ser exibida ao jogador.

O método tick() tem de ser limitado quanto a sua velocidade de execução, e consequentemente o render() também, para que os computadores rodem a uma taxa de frames per second única.

Instancia também as States do jogo que são elementos importantes que serão explicados posteriormente.

Classe Handler

Essa classe gerencia as variáveis instanciadas na classe Game, devido a isso sua composição é de getters e setters, é voltada para a organização do código.

Classe Launch

Instancia um objeto da classe Game fazendo com que o jogo seja rodado.

Classe BattleManager

É o gerenciador voltado à batalha, gerenciando turnos e a vida dos lutadores e instanciando um objeto da Classe cursor que é útil na implementação do BattleState.

Package Entities

Classe Entity

Classe de objetos que interagem com o jogador, inclusive o próprio jogador é de uma classe filha desta.

Tem como atributos x e y que são as posições nas quais os objetos dessa classe serão renderizados na janela, também temos height e width que são os tamanhos nos quais os assets serão renderizados.

Uma outra informação é que ela também é responsável por verificar colisões entre entidades, só lembrando que colisões são quando há a interseção de duas collisionbox que no nosso caso é representado pelo Rectangle bounds.

Classe EntityManager

Gerencia as entidades, possui um ArrayList de objetos da Classe Entity, ela também é responsável por setar a ordem de renderização de objetos para caso o player esteja atrás de uma árvore, este seja renderizado primeiro que ela e consequentemente fique atrás dela aos olhos do jogador.

Classe Creature

Essa Classe herda da Classe Entity, são objetos que podem se mover e realizar ações como por exemplo: O próprio Player e o Enemy.

Classe Player

Classe responsável pelas ações do personagem como mover, interagir, atacar, atualizar (tick) e renderizar as animações do jogador.
Também é quem realiza o processamento das teclas de movimentação.

Classe Cursor

Representa o Player nas batalhas agindo como indicador da ação selecionada, renderiza as possibilidade de ações na tela.

Classe StaticEntity

Classe separada devido a esses serem objetos que não se movem (estáticos), porém tem ações com os objetos do jogo.

Package Graphics

Classe Animations

Realiza a sincronização da execução de animações do player, sendo que as animações são armazenadas em Vetores do tipo BufferedImage.

Classe Assets

Realiza o carregamento e processamento das imagens (assets) que serão utilizadas no jogo

Classe ImageLoader

Responsável por localizar e a leitura da imagem.

Classe spriteSheet

Realiza o corte da Sprite Sheet em Buffered Image. Isso é devido à uma Sprite Sheet ter vários conteúdos para o jogo, como por exemplo, as animações dos personagens mas, precisamos na verdade de uma certa parte da imagem, então entra o método crop() dessa classe que corta (crop) partes específicas.

Classe gameCamera

Essa Classe tem duas váriaveis importantes xOffset e yOffset, ambas indicam as coordenadas nas quais as Camera do jogo será centralizada, o método centerEntity(Entity e) faz com que a Câmera siga uma Entity por meio das coordenadas dessas.

Package Inputs

Classe KeyManager

Realiza reconhecimento das entradas da entrada do usuário por meio do KeyListener

Package States

Para um jogo cada tipo de frame diferente é um state.

Temos:

- BattleState: Batalha.
- GameState: State no qual o jogador pode andar pelo mapa.
- DefeatState: Quando o jogador é derrotado.
- MainMenuState: Menu principal.
- WinState: Estado de vitoria.

Package Tiles

Classe Tile

Instancia os blocos, cada um com sua id, que farão parte do mundo.

Classes-filhas de Tile

Essas Classes servem para configurar os blocos e seus respectivos assets.

Package Utils

Classe Util

Processa a informação das coordenadas de cada tipo de bloco no mundo e do spawn point do player que está em um txt.

Package World

Classe World

Organiza as informações do mundo na tela, especificamente os Tiles,tem os métodos necessários para carregar o mundo