Checkpoint 3 - Eliton Sena de Souza

- Diagrama de Classes

Diagrama de classes em UML aqui.

- Justificativas de Design

Pacote dronefront e dronefront.game

- **Game:** ela gerencia o loop do jogo, alternando entre a fase de construção (BUILD_PHASE) e a fase de ataque (WAVE_IN_PROGRESS), funcionando como uma maquina de estados. além disso, também cuida dos recursos do jogador, como a vida da base e as moedas, e processa os comandos de entrada, como mover o cursor e construir torres.
- **WaveManager:** Responsável por controlar as ondas de inimigos. Ele lê as configurações de cada onda (quantidade de inimigo, probabilidade de surgir x inimigo e o intervalo) e os instancia no mapa no momento certo.
- **Wave:** Uma classe que define a estrutura de uma única onda, contendo uma lista de eventos de spawn de inimigos.
- **ConsoleUI:** gerencia toda a parte visual do jogo no console. desenha o mapa, os inimigos, as torres e exibe informações essenciais como vida da base, moedas e a onda atual.

Pacote dronefront.map

- **GridMap:** cria o mapa do jogo como uma grade de tiles. define o caminho que os inimigos seguirão e quais locais estão disponíveis para construção.
- **Caminho:** Representa a rota que os inimigos devem seguir, armazenando uma lista ordenada de pontos (Ponto) no mapa.
- **TileMap:** É um único "azulejo" do mapa. Ele sabe sua posição e se é permitido ou não construir uma torre nele.
- **Ponto & Position:** duas classes para localização. Position usa coordenadas inteiras (int) para se referir a um tile específico na grade. Ponto usa coordenadas de ponto flutuante (double) para a localização exata de inimigos e projéteis, permitindo um movimento mais suave, além do cálculo de distâncias entre elementos do jogo (deltatime).

Pacote dronefront.enemy

- **Enemy**: é a superclasse abstrata para todos os inimigos. Ela define os atributos e comportamentos básicos, como pontos de vida (hp), velocidade, dano, e a moeda que o jogador ganha. Gerencia efeitos de status, como lentidão, queimadura e um fator de vulnerabilidade que aumenta o dano recebido.
- **ScoutDrone**: Um inimigo rápido, mas com pouca vida e que causa pouco dano. Ele possui uma implementação customizada de slow que o torna lento, mas não aplica o bônus de vulnerabilidade a dano que outros drones recebem.

- **BomberDrone**: Um inimigo mais lento e resistente que o Scout. Ele é vulnerável ao fogo: sua implementação de burn faz com que ele receba **1.5x** mais dano de queimadura e por **1.5x** mais tempo.
- **TankDrone**: Um novo tipo de inimigo, extremamente lento e com uma quantidade massiva de pontos de vida (500 HP). Ele é resistente a controle: sua implementação de slow **reduz pela metade** a eficácia da lentidão (tanto a redução de velocidade quanto a duração do efeito).

Pacote dronefront.tower

- **Tower:** uma superclasse abstrata para todas as torres. Ela estabelece os parâmetros fundamentais como posição, alcance (range), tempo de recarga (cooldown) e custo (cost). A lógica de encontrar um alvo (findTarget) e o gerenciamento do tempo de recarga são definidos aqui. Agora, ela também gerencia o nível (level) da torre, com um nível máximo (padrão de 3), e define métodos abstratos para upgrade permitindo aprimoramentos.
- **GunTower**: Uma torre de dano direto. Ela procura o inimigo que está mais avançado no caminho (mais próximo da base). Ao disparar, cria um projétil (Bullet). Seu upgrade aumenta o dano, o alcance e diminui o tempo de recarga.
- **PEMTower**: Uma torre de suporte que aplica lentidão. Ela mira no inimigo com mais HP dentro do seu alcance. Seu upgrade aumenta o alcance, a duração da lentidão e diminui o tempo de recarga.
- **FireTower**: Uma nova torre de dano ao longo do tempo. Ela aplica um efeito de queimadura no alvo. Ela mira no inimigo mais próximo da própria torre. Seu upgrade aumenta o dano por segundo da queimadura, a duração do efeito e o alcance da torre.

Pacote dronefront.projectile

- **Projectile:** A classe base para todos os projéteis. Define o comportamento de seguir um alvo (target), a velocidade e o dano que causa. A lógica de movimento em direção ao inimigo e de detecção de colisão é implementada aqui.
- **Bullet:** Um tipo específico de projétil, disparado pela GunTower. Ele simplesmente se move em direção ao alvo para causar dano ao atingi-lo.

to-do:

- penso em apagar a classe ponto e colocar apenas a classe position e um método para transformar double para integer p ser usado quando necessário.
- resolver bug do projétil “pulando” as tiles no console. (deve ser resolvido com interface)
- ter um banco de dados de mapas diferentes. a cada início de jogo ele vai selecionar um aleatoriamente.