Hermit Crab Game Test

Autor: Lucas Zaranza

Estilo de jogo: Runner 2D

Versão da Unity: 2020.3.44f1 LTS

Tempo decorrido: 6 dias

Objetivos do game:

Um robô que percorre um cenário e vai acumulando pontos à medida em que sobrevive. Ele deverá evitar os obstáculos e ver o quão longe consegue ir.

Mecânicas:

- Ganha 1 ponto a cada meio segundo;
- Jogador tem 100 pontos de vida;
- Tocar no acido custa 20 pontos de vida;
- Tocar nos espinhos custa 30 pontos de vida;
- Atirar 3 vezes no contêiner vermelho o destrói;
- Ficar preso no cenário e desaparecer da tela causa Game Over;

Input:

- Swipe up: Pulo
- Swipe down: Dash
- Touch: Tiro

Arquitetura utilizada:

- Sintaxe CamelCase;
- Fluxo de Controllers;
- Singletons;
- Orientação a Eventos utilizando Actions do C#;

- Orientação a Objetos;

- SOLID;

- Organização dos scripts:

    1 – Constantes;

    2 – Eventos;

    3 – Campos serializáveis no Editor;

    4 – Membros privados;

    5 – Props;

    6 – Funções padrão (Start, Awake, Enable)

    7 – Triggers de Colisão / Funções de Destroyer

Controllers: Unidades que são as responsáveis pelo controle das mecânicas do jogo. São controladas por uma outra controladora principal chamada *GameController*, que é a que dispara suas ações e seta todos os *event handlers*.

Cada *Controller* gerencia uma mecânica específica do game: desde gerenciamento do Input até o disparo da criação de novos blocos que o jogador poderá pisar. As *Controllers* acessam outros scripts menores que ficam responsáveis por funcionalidades específicas de cada sistema que gerenciam, como por exemplo, a *UIController* gerencia a atualização da barra de vida do jogador acessando os métodos do script *HealthBar*. Outro exemplo é o *FloorController* que muda os blocos de um lado pro outro acessando o script *FloorScrolling*, que por sua vez altera as velocidades de movimentação dos blocos, ou acessando o *TileSpawner* muda seu *sprite*.

Uma *Controller* ou script não se comunica com outro diretamente, mas através do disparo de eventos, a *GameController* realiza a comunicação de um pro outro se necessário. Apenas em alguns casos menores que um script dispara algum evento que será tratado por outro menor, como por exemplo o jogador trata o evento *HandleOnBlinkStopped*, que vai dar ao player o controle do que fazer quando ele parar de piscar. É um comportamento mais isolado e que não afeta diretamente o fluxo do game.

Maiores dificuldades: Implementar a lógica da Dash e organizar o Animator e o disparo correto das animações adequadas. Precisei lidar com a detecção de colisão de alguns tiles de uma maneira diferente, o que adicionou algumas horas a mais de trabalho.

Próximos updates:

- Implementar o tiro e suas animações;

- Criar obstáculo destrutível pelo tiro;

- Item pra recuperar vida que pode ser deixado pelo obstáculo destrutível;

- Item pra aumentar a pontuação que pode ser deixado pelo obstáculo destrutível;

- Polir o algoritmo da escolha de obstáculos e encaixar melhor com o tile spawning;

- Power-ups;

- Inimigos;

- Mais tiles que causem dano além do ácido e dos espinhos;

- Novos cenários / passagem de estágio;

Lucas Zaranza