Game Design Document (GDD) - Plataforma Dino

1. **Visão Geral**

- **Título**: Dino Adventure
- **Plataformas**: PC, Mobile
- **Gênero**: Plataforma 2D
- **Público-Alvo**: Jogadores casuais
- **Descrição**: O jogador assume o controle de um dinossauro que deve atravessar cenários repletos de obstáculos e itens colecionáveis, como caixas e vegetação, enquanto enfrenta desafios para progredir nas fases.

2. **Principais Classes**

2.1. **Character Base**

- **Player Movement Script - Pontos Principais**:
  - **Personagem e Câmera**:
    - character: Transform do personagem principal.
    - targetCam: Câmera que acompanha o personagem.
  - **Configurações de Movimento**:
    - CanMove: Booleano que determina se o personagem pode se mover.
    - moveSpeed: Velocidade de movimento do personagem.
    - jumpForce: Força do pulo do personagem.
  - **Entrada de Controle**:
    - moveInput: Captura o input de movimento (direção).
    - jumpInput: Captura o input de pulo.
    - Sistema de input mapeado através do PlayerInputActions.
  - **Checagem de Solo**:
    - groundCheck: Ponto de verificação para determinar se o personagem está no chão.
    - checkRadius: Raio da área de checagem.
    - layerIsGround: Layer utilizada para definir o que é considerado "chão".
  - **Componentes Requeridos**:
    - rb: Rigidbody2D para manipulação física.
    - animator: Controlador das animações do personagem.
  - **Lógica de Movimento**:
    - Se CanMove for verdadeiro, o personagem se move na direção do moveInput.
    - A sprite é invertida de acordo com a direção do movimento.
  - **Lógica de Pulo**:
    - O personagem pode pular apenas se estiver no chão e o input de pulo estiver ativo.
  - **Animações**:
    - As animações de velocidade e pulo são ajustadas com base nos inputs e na checagem de solo.
  - **Eventos de Estágio**:
    - Configuração inicial do estágio e ativação da câmera conforme o estágio atual.
  - **Sistema de Dano**:
    - TakeDamage: Função responsável por receber dano, reduzindo a vida do personagem através do statusCharacter.

2.2. **Status Character Base**

- isAlive: Booleano que indica se o personagem está vivo.
- healthCurrent: Vida atual do personagem.
- healthMax: Vida máxima do personagem.

- **Métodos de Controle de Vida**:
  - SetHealthMax(int value): Define o valor máximo de vida e ajusta a vida atual para esse valor.
  - AddHealth(int value): Aumenta a vida do personagem, respeitando o valor máximo.
  - RemoveHealth(int value): Reduz a vida do personagem, assegurando que não fique abaixo de zero.
- **Morte do Personagem**:
  - Se healthCurrent chegar a zero, a função Did() é chamada, marcando o personagem como morto (isAlive = false).
  - O evento de morte ativa a ação St_StageAction.DidAction.
- **Integração com o Estágio**:
  - Sempre que a vida do personagem for alterada, a função St_StageAction.LifeCharacteUpdate é chamada para atualizar a interface ou outros elementos do jogo.

2.3. **InteractiveObjectBase**

- objectDestroyedPrefab: Prefab a ser instanciado quando o objeto for destruído.
- canOnCollisionEnter2D: Controla se a função OnCollisionEnter2D pode ser executada.
- canOnTriggerEnter2D: Controla se a função OnTriggerEnter2D pode ser executada.
- **Detecção de Colisão com o Player**:
  - **Colisão 2D**:
    - OnCollisionEnter2D: Verifica colisões com objetos que possuem a tag "Player" e chama a função ActionOnCollisionEnter2D.
    - OnCollisionExit2D: Limpa a referência ao player quando a colisão termina e chama ActionOnCollisionExit2D.
  - **Trigger 2D**:
    - OnTriggerEnter2D: Detecta quando o player entra na área de um Trigger e chama ActionOnTriggerEnter2D.
- **Ações Virtuais**:
  - ActionOnCollisionEnter2D, ActionOnCollisionExit2D, ActionOnTriggerEnter2D: Métodos virtuais que podem ser sobrescritos para definir o comportamento específico de cada colisão ou trigger.
- **Destruição do Objeto**:
  - DestroyWithInstantiate: Instancia o prefab definido em objectDestroyedPrefab na posição atual do objeto antes de destruí-lo.
  - Destroy: Método que remove o objeto atual do jogo.

2.4. **St_StageAction**

- currentStageController: Referência ao controlador de estágio atual.
- TextBoxesCurrentStage, TextTimeCurrentStage: Textos na interface (UI) que exibem o progresso e o tempo do estágio.
- IconeCurrentStage: Ícone relacionado ao personagem utilizado no estágio atual.
- LifeCharacterEggs: Array de objetos que representam a vida do personagem na interface.
- CheckPosition: Posição do ponto de respawn (checkpoint).
- MobileCanvas: Interface de controles móveis.
- NotificationMessager: Sistema de notificações do jogo.
- **Ações Estáticas**:
  - SetTagerCamInStageAction: Ação que define a posição da câmera no estágio.
  - DidAction: Ação chamada quando o personagem morre.
  - FinishAction: Ação chamada ao concluir o estágio.
- **Métodos Estáticos**:
  - SetTagerCamInStage(Transform stageController): Ajusta a câmera para focar no estágio atual usando a ação SetTagerCamInStageAction.

- RespawnCheckPoint(): Teleporta o personagem para o último checkpoint salvo, permitindo que ele se mova novamente.
- LifeCharacteUpdate(int value): Atualiza a interface de vida do personagem, desativando todos os indicadores de vida e reativando a quantidade correspondente ao valor de vida atual.

2.5. **StageController**

- statusStage: Armazena o status atual do estágio, como tentativas, tempo e pontuação.
- stageData: Contém informações do estágio, como ID e nome.
- playerCharacter: Referência ao personagem jogável.
- **Estados de Controle**:
  - isStartingStage: Booleano que indica se o estágio está em andamento.
- **Métodos Principais**:
  - OnEnable(): Inicializa o estágio quando o script é ativado.
  - Update(): Verifica se o estágio está ativo e atualiza o tempo do estágio.
  - OnDisable(): Reseta o controlador de estágio quando o script é desativado.
- **Inicialização do Estágio**:
  - Initialized(): Recupera o status salvo do estágio e incrementa o número de tentativas. Define o controlador atual (St_StageAction.currentStageController) e chama SpawnCharacter() para posicionar ou instanciar o personagem no início do estágio.
- **Spawn do Personagem**:
  - SpawnCharacter(): Encontra a posição inicial (startingPosition) e coloca o personagem ali. Se o personagem não estiver presente, instancia o prefab baseado nos dados do personagem selecionado.
- **Pontuação e Interface**:
  - AddScore(int value): Adiciona valor à pontuação das caixas coletadas e atualiza o texto da interface (TextBoxesCurrentStage).
- **Controle de Estágio**:
  - StartingStage(): Inicia o estágio, permitindo que o tempo e as ações ocorram.
  - StopStage(): Pausa o estágio, interrompendo a contagem de tempo e movimentos.
- **Gerenciamento de Tempo**:
  - TimeStage(): Atualiza o tempo total do estágio e converte o valor para um formato legível.
  - ConvertTime(float seconds): Converte o tempo total (em segundos) para o formato HH:MM ou MM.

2.6. **UI_MainMenuManager**

- canvasGroupController: Controla a exibição de diferentes grupos de elementos na interface.
- ui_DashBoard: Interface do painel de controle (dashboard) do jogo.
- ui_MenuInitial: Interface do menu inicial do jogo.
- **Métodos Principais**:
  - Awake(): Método chamado ao iniciar o script, que faz a configuração inicial dos componentes e chama a função de inicialização.
  - GetComponents(): Verifica e obtém os componentes necessários:
    - canvasGroupController: Obtido do próprio objeto.
    - ui_DashBoard e ui_MenuInitial: Obtidos dos filhos do objeto atual.
  - Initialized(): Inicializa o canvas, ativando o primeiro grupo de UI (índice 0).

2.7. **UI_MainStage**

- btn_Start: Botão que inicia o jogo.
- text_Timer: Texto que exibe o tempo restante.

- text_Score: Texto que apresenta a pontuação atual do jogador.
- **Métodos de Controle de UI**:
  - UpdateTime(float time): Atualiza o texto do tempo na interface.
  - UpdateScore(int score): Atualiza o texto da pontuação na interface.

1. **Cenários**
- **Ambientes**: Florestas, montanhas e cenários aquáticos.
- **Obstáculos**: Caixas, vegetação, pedras e criaturas.
- **Coletáveis**: Caixas.

2. **Interface do Usuário (UI)**
- Menus de navegação.
- Interface de HUD para exibir pontuação e tempo.

3. **Música e Efeitos Sonoros**
- Ainda em desenvolvimento.

4. **Arte**
- **Estilo Visual**: Cores vibrantes.
- **Assets Utilizados**: Árvores, cogumelos, placas de informação, pedras, troncos, chão e água.

5. **Desenvolvimento**
- **Tecnologias Utilizadas**: Unity, C#.

Tempo gasto no projeto: Cerca de 28 horas até o momento.

Maiores dificuldades: Encontrar tempo para se dedicar ao desenvolvimento.