Equipe: Guaravitas

Nomes: Felipe Batista Dos Santos(ADS - Manhã segundo semestre) ,Gustavo Meneses Ruegenberg Rodrigues(ADS - vespertino terceiro semestre), Ricardo Gabriel Camargo Soares(ADS - Manhã segundo semestre)

Título: WildLife Bits

Ferramenta: Godot.

Descrição do projeto: WildLife Bits, é um jogo no estilo pixel art, com visão top Down do gênero Action RPG.

Abordagem do tema: Os temas abordados foram: educação de qualidade(ODS 4), vida terrestre(ODS 15) e Fatec.
Esses elementos foram abordados das seguintes formas, a ODS 4 foi abordada no glossário contendo as algumas informações dos animais presentes no mapa, a ODS 15 foi representada pelo objetivo do jogo de salvar os animais os levando de volta para a flores e a Fatec foi utilizada como parte do cenário sendo ela o ponto de spawn do personagem que será utilizado durante a gameplay.

Descrição geral do projeto: O WildLife Bits é um jogo com objetivo de salvar os animais presentes no mapa o mais rápido possível, e coletar alguns pontos ao redor dele, destruindo os sacos de lixo que ficam pelo mapa, o jogador ganha se salvar todos os animais ou se marcar 40 pontos, esses pontos são adquiridos, salvando animais e destruindo o lixo, o jogador perde se o tempo acabar.

Arquitetura de solução: Utilizamos a godot para o desenvolvimento das funcionalidades do jogo

Instruções: Instalar o arquivo zip, extrair todos os arquivos, acessar a pasta “Jogo”, e dentro dela vai ter um arquivo chamado “WildLife Bits” ao executá-lo irá rodar o jogo

Tecnologias e padrões de projeto: As principais tecnologias utilizadas foram, godot para desenvolvimento, finalbossblues.itch.io, para design do mapa, lyaseek.itch.io, para design dos personagens, kenney.nl para design dos elementos no mapa(carros, árvores, etc).

Mudanças entre relatórios: A ideia principal se manteve, porém, houve uma redução nas funcionalidades do jogo, onde a ideia de haver múltiplas armas, específica para cada tipo de animal, acabou sendo descartada e substituída por ao invés de haver um leve confronto entre o player e os animais agora os animais já seguem o player sem a necessidade de confronto.

Arquitetura de solução:

1. Visão Geral e Padrões de Arquitetura

O projeto é um jogo 2D top-down desenvolvido na engine Godot 4.5.1 com GDScript. A arquitetura é centrada em uma **máquina de cenas** para o fluxo de jogo e um **Singleton (Autoload)** para o gerenciamento de estado.

- **Padrão Singleton:** O ScoreManager.gd atua como a única fonte da verdade para o estado do jogo (pontuação atual, pontuação de vitória). Ele usa sinais (score_updated, victory_achieved) para comunicar mudanças de estado para o resto da aplicação, desacoplando a lógica de pontuação dos controladores de cena.
- **Fluxo de Cena:** O jogo é segmentado em cenas principais (menu, city, glossario, game_over, victory_screen) que são carregadas e descarregadas pelo SceneTree (get_tree().change_scene_to_file()).
- **Gerenciamento de Pausa:** O estado de pausa (get_tree().paused) é usado para exibir pop-ups e telas de fim de jogo. Cenas de UI interativas (pop-ups de mensagem, menu de pausa) usam modos de processamento (PROCESS_MODE_WHEN_PAUSED ou PROCESS_MODE_ALWAYS) para funcionar enquanto o jogo principal está congelado.

---

2. Fluxo de Cenas e Interação do Usuário

1. **Início:** O jogo começa no menu.tscn (controlado por menu.gd).
    - **Ação:** _on_comecar_button_pressed -> Carrega city.tscn.
    - **Ação:** _on_glossario_button_pressed -> Carrega glossario.tscn.
    - **Ação:** _on_sair_button_pressed -> Fecha o jogo.
2. **Início da Fase:** Ao carregar city.tscn (controlado por city.gd):
    - O jogo é **imediatamente pausado** (get_tree().paused = true).
    - mensagem_objetivo.tscn é instanciada e exibida.
    - **Ação (Pop-up 1):** _on_button_pressed em mensagem_objetivo.gd instancia mensagem_controles.tscn e se autodestrói.
    - **Ação (Pop-up 2):** _on_button_pressed em mensagem_controles.gd **despausa o jogo** (get_tree().paused = false) e se autodestrói.
    - O jogo (e o PhaseTimer) começam.
3. **Gameplay (Fase Principal):**
    - **Pausa:** O jogador pode pressionar "esc" a qualquer momento para acionar pause.gd, que pausa/despausa o jogo e exibe um menu de pause.
    - **Objetivo 1 (Animais):** O jogador guia animais para a zona animal_home.tscn. O script animal_home.gd detecta a entrada, chama ScoreManager.add_score(3) e deleta o nó do animal.
    - **Objetivo 2 (Lixo):** O jogador interage com trash.tscn. O script trash.gd detecta a entrada, chama ScoreManager.add_score(1) e **deleta o próprio nó trash.tscn** (queue_free()).
4. **Condições de Fim de Jogo (em city.gd):**
    - **Vitória:** ScoreManager atinge 40 pontos (victory_score) e emite victory_achieved. O city.gd ouve este sinal, chama on_victory(), pausa o jogo e instancia victory_screen.tscn.
    - **Derrota (Tempo):** O PhaseTimer (em city.tscn) atinge 0. O city.gd ouve o sinal timeout, chama on_time_up(), pausa o jogo e instancia game_over.tscn.

5. **Telas Finais:**
    - Em game_over.tscn (controlado por game_over.gd):
        - **Ação:** _on_restart_btn_pressed despausa o jogo, reseta o score e carrega menu.tscn.
        - **Ação:** _on_quit_btn_pressed fecha o jogo.
    - Em victory_screen.tscn (lógica presumida, similar ao game_over): Permite ao jogador voltar ao menu ou sair.

---

3. Detalhamento dos Sistemas Centrais

A. Sistema de Gerenciamento de Estado (ScoreManager.gd - Autoload)

- Atua como o "cérebro" do estado do jogo.
- **Propriedades:** victory_score = 40, current_score = 0, target_score = 10.
- **Sinais:** victory_achieved (quando current_score >= 40), score_updated (a cada pontuação), target_score_reached (quando current_score >= 10, este sinal não é ouvido por nenhum outro script).
- **Funções:** add_score(points), get_score(), reset_score().

B. Orquestrador da Fase (city.gd - Nó Raiz "Mundo")

- Serve como o controlador principal da cena de jogo.
- **Gerencia o Ciclo de Vida da Fase:** Pausa no início para pop-ups, conecta os sinais de vitória e derrota, e atualiza a UI de tempo.
- **Responsabilidade de UI:** Atualiza o timer_label.text a cada frame lendo o timer.time_left.
- **Responsabilidade de Fim de Jogo:** Instancia as cenas game_over e victory quando seus respectivos sinais (timeout, victory_achieved) são recebidos.

C. Sistema de Interface (HUD, Menus e Pop-ups)

- **HUD (label.gd):** O script do ScoreLabel. Ouve o ScoreManager.score_updated e formata o texto com um valor fixo: text = "Score: [pontos] / 40".
- **Menu Principal (menu.gd):** Controlador de navegação simples para iniciar o jogo, ver o glossário ou sair.
- **Menu de Pause (pause.gd):** Um Control que roda em PROCESS_MODE_ALWAYS. Ele se sobrepõe a tudo (incluindo pop-ups e telas de fim de jogo) e monitora a tecla "esc" para pausar ou resumir o jogo.
- **Pop-ups Iniciais (mensagem_objetivo.gd, mensagem_controles.gd):** Um fluxo de duas partes. A primeira (objetivo) chama a segunda (controles), e a segunda (controles) despausa o jogo, efetivamente iniciando a fase.

D. Sistema de Conteúdo (glossario.gd)

- Controla a tela glossario.tscn.
- Armazena todos os dados textuais e de imagem dos animais em um **Array (animais) embutido no script**.

- Popula dinamicamente os campos de UI (nome, descricao, imgAnimal, etc.) baseado no índice selecionado de um ItemList (_on_item_list_item_selected).
- Permite ao jogador retornar ao menu (_on_sair_button_pressed).

E. Sistemas de Pontuação (Entidades de Jogo)

- **animal_home.gd:** Define a "zona de gol". Ao ser acionado por uma área do grupo "animal", adiciona **3 pontos** e deleta o animal (other_area_2d.get_owner().queue_free()).
- **trash.gd:** Define um item colecionável. Ao ser acionado (presumivelmente pelo jogador), adiciona **1 ponto** e deleta a si mesmo (queue_free()), permitindo que funcione apenas uma vez.