09/11/2023

Projeto Final

The Village PEDRO CYRNE

LUCAS FERREIRA

GABRIEL FELIPPE

URIEL RELVAS

CELSO VINICIUS

PROFESSOR:

LUIZ FERNANDO

SUMÁRIO

3- Anamnese

4- Caso de uso descritivo 5- Diagrama de caso de uso 6-8 - Telas

9-33 - Códigos

ANAMNESE

Parte gráfica:

O jogo é 2D, ou seja, duas dimensões X e Y (cima e baixo respectivamente).

Devido ao jogo ser 2D o gráfico dele terá base em “sprites” que são, basicamente, imagens em png.

Jogabilidade:

Não terá nenhum tipo de combate direto. O jogo teria como base uma exploração do mundo.

História:

A história é linear, segue uma sequência de acontecimentos e interações simples.

CASO DE USO

Descritivo

O jogador abre o jogo em seu computador, que logo em seguida, entrará no menu principal. Onde será apresentado algumas opções, como Jogar e Sair. O botão sair encerra o processo do jogo, fechando-o

Já o botão jogar o levará para um nível onde será capaz de começar o jogo em si.

A Game Engine, Unity, é responsável não só pela maior parte dos cálculos mas ambém pelo registro de "inputs" do usuário. A movimentação é restrita nos eixos X e Y. A mecânica de "vault", que é o que permite o jogador escalar/se pendurar em certos objetos, interagir com objetos, também irá fazer parte da jogabilidade.

Também é possível salvar o progresso, como a última localização que o jogador estava e o seu progresso nas “quests” (objetivos).

DIAGRAMA DE CASO DE USO:

—————————->

Telas:

Menu Inicial:

Em Jogo:



**Código responsável pela movimentação dos pássaros:**

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**public class Bird : MonoBehaviour**

**{**

**private float velocidade;**

**public float limiteDeDestruicao;**

**public int right;**

**private void Start(){**

**velocidade = Random.Range(0.4f, 2.2f);**

**right = Random.Range(0, 1);**

**}**

**void FixedUpdate()**

**{**

**if(right == 1){**

**transform.Translate(Vector3.right \* velocidade \* Time.deltaTime);**

**if (transform.position.x > limiteDeDestruicao) {**

**Destroy(gameObject);**

**}**

**}else{**

**transform.Translate(Vector3.left \* velocidade \* Time.deltaTime);**

**if (transform.position.x < limiteDeDestruicao) {**

**Destroy(gameObject);**

**}**

**}**

**}**

**}**

Códigos:

**Código responsável pelo surgimento dos pássaros na fase:**

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**public class BirdSpawner : MonoBehaviour**

**{**

**[Header("Valores")]**

**[SerializeField] private float Tempo\_de\_Spawn\_Min; [SerializeField] private float Tempo\_de\_Spawn\_Max;**

**[Header("Componentes")]**

**[SerializeField] private GameObject Passaro;**

**private int Randomizer;**

**void Start()**

**{**

**InvokeRepeating("Spawn\_Passaro",**

**Random.Range(Tempo\_de\_Spawn\_Min, Tempo\_de\_Spawn\_Max), Random.Range(Tempo\_de\_Spawn\_Min, Tempo\_de\_Spawn\_Max)); }**

**private void Spawn\_Passaro(){**

**Vector2 posicao = new Vector2(transform.position.x, Random.Range(1, 4));**

**Instantiate(Passaro, posicao, Quaternion.identity); }**

**}**

**Código responsável pelos botões da tela inicial.**

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**using UnityEngine.UI;**

**using TMPro;**

**using UnityEngine.SceneManagement;**

**public class Buttons : MonoBehaviour {**

**public AudioClip meuSom;**

**public AudioSource meuAudioSource;**

**public void Play(){**

**meuAudioSource.clip = meuSom;**

**meuAudioSource.Play();**

**SceneManager.LoadScene("Loading"); }**

**public void Settings(){**

**meuAudioSource.clip = meuSom;**

**meuAudioSource.Play();**

**SceneManager.LoadScene("Settings"); }**

**public void Quit(){**

**meuAudioSource.clip = meuSom;**

**meuAudioSource.Play();**

**Application.Quit();**

**}**

**public void Next(){**

**SceneManager.LoadScene(1);**

**}**

**}**

Códigos

**Código responsável pela movimentação da câmera:**

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**public class CameraMovement : MonoBehaviour**

**{**

**private Vector3 Camera\_Offset; //Esse é o valor responsavel pelo offset da**

**camera, nesse caso, o quão longe ela vai se manter do player**

**[Header("Values")]**

**[SerializeField] private float Camera\_Smooth\_Multipliyer; //Esse é o valor**

**que sera responsavel pela suavização**

**[Header("Components")]**

**[SerializeField] private Transform Camera\_Target\_Follow; //esse é onde o**

**unity se referira ao transform que a camera ira seguir, ate momento e**

**geralmente o avatar do player**

**//Others**

**private Vector3 Camera\_Velocity = Vector3.zero;**

**private void Awake() => Camera\_Offset = transform.position -**

**Camera\_Target\_Follow.position; //Toda vez que a camera for ativada, isso era**

**ativado junto. Calculo do offset basicamente.**

**private void LateUpdate() //a ultima coisa calculada depois dos updates**

**{**

**Callculus();**

**}**

**private void Callculus(){**

**Vector3 Camera\_Targeted\_Position = Camera\_Target\_Follow.position +**

**Camera\_Offset; //faz os calculos de qual posição a camera deverá ou deveria**

**estar.**

**transform.position = Vector3.SmoothDamp(transform.position,**

**Camera\_Targeted\_Position, ref Camera\_Velocity, Camera\_Smooth\_Multipliyer);**

**// Basicamente faz a movimentação da camera pelo mapa depois de todas as**

**contas**

**}**

**}**

**Código responsável pela primeira “quest” do level:**

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**using UnityEngine.UI;**

**using TMPro;**

**public class firstquest : MonoBehaviour**

**{**

**[Header("Valores")]**

**[SerializeField] private int QuestID;**

**[SerializeField] private int QuestPosition\_X;**

**[SerializeField] private int QuestPosition\_Y;**

**[Header("Componentes")]**

**[SerializeField] private Animator QuestCache;**

**[SerializeField] private TextMeshProUGUI Name;**

**[SerializeField] private GameObject DialogBox;**

**[SerializeField] private TextMeshProUGUI dialogtext;**

**[SerializeField] private Transform QuestTracker;**

**[Header("Dialogo")]**

**[SerializeField] private string[] lines;**

**[SerializeField] private float textSpeed;**

**public int index;**

**private string clear;**

**private void Start() {**

**dialogtext.text = string.Empty;**

**}**

**private void OnTriggerEnter2D(Collider2D other){ if(QuestCache.GetInteger("QuestStatus") == QuestID){ DialogBox.SetActive(true);**

**startdialoge();**

**QuestTracker.position = new Vector2 (QuestPosition\_X, QuestPosition\_Y); QuestCache.SetInteger("QuestStatus", QuestID + 1); }**

**if(QuestCache.GetInteger("QuestStatus") == 2){ QuestCache.SetInteger("QuestStatus", 3);**

**}**

**}**

**private void OnTriggerStay2D(Collider2D other){ if(Input.GetKeyDown(KeyCode.F)){**

**DialogSkipper();**

**}**

**}**

**private void OnTriggerExit2D(Collider2D other){ DialogBox.SetActive(false);**

**}**

**private void startdialoge(){**

**dialogtext.text = clear;**

**index = 0;**

**StartCoroutine(TypeLine());**

**}**

**private void DialogSkipper(){**

**index += 1;**

**dialogtext.text = clear;**

**StartCoroutine(TypeLine());**

**}**

**IEnumerator TypeLine(){**

**foreach (char c in lines[index].ToCharArray()){**

**dialogtext.text += c;**

**yield return new WaitForSeconds(textSpeed);**

**}**

Obrigado!