ESCOLA TECNICA ESTADUAL REPUBLICA

TÉCNICO DE REDES

PROJETO FINAL - The Village

PEDRO CYRNE
LUCAS FERREIRA
GABRIEL FELIPPE
URIEL RELVAS
CELSO VINICIUS

PROFESSOR
LUIZ FERNANDO

2023

SUMÁRIO

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.

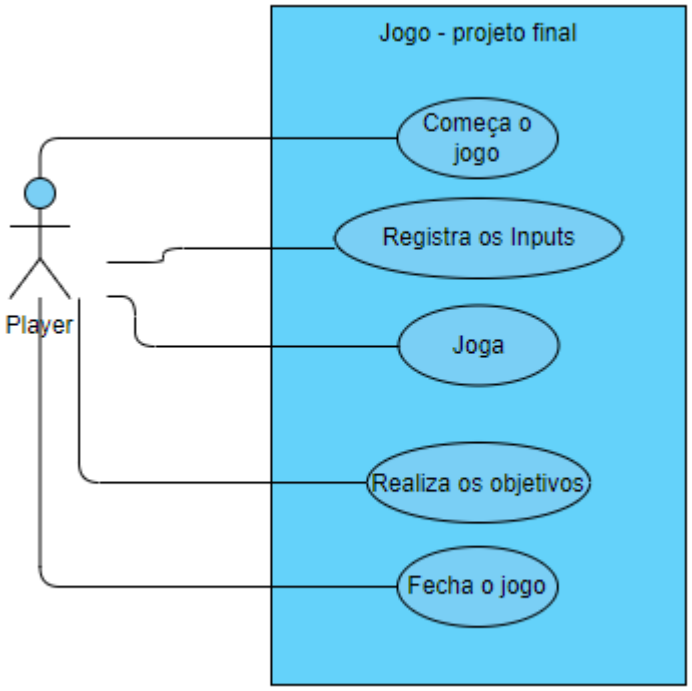
História:
A história é linear, segue uma sequência de acontecimentos e interações simples.

CASO DE USO

Descritivo

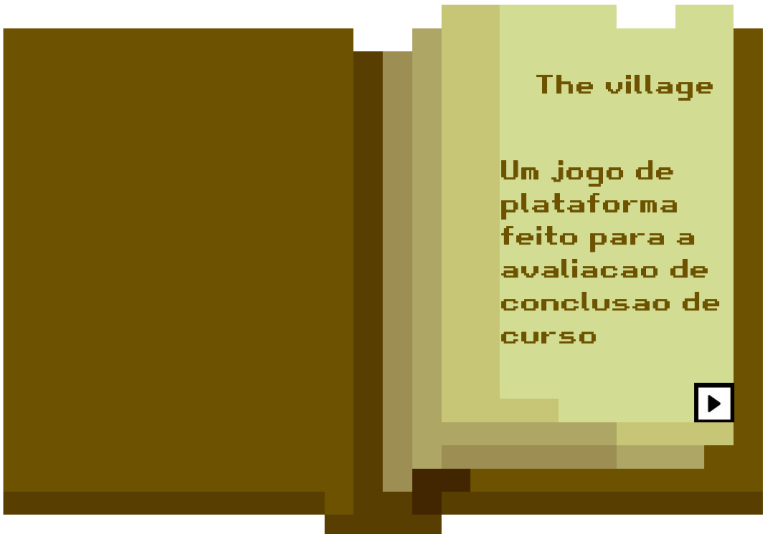
O jogador abre o jogo em seu computador, que logo em seguida, entrará no menu principal. Onde será apresentado algumas opções, como Jogar e Sair. O botão sair encerra o processo do jogo, fechando-o. Já o botão jogar o levará para um nível onde será capaz de começar o jogo em si. A Game Engine, Unity, é responsável não só pela maior parte dos cálculos mas também 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).

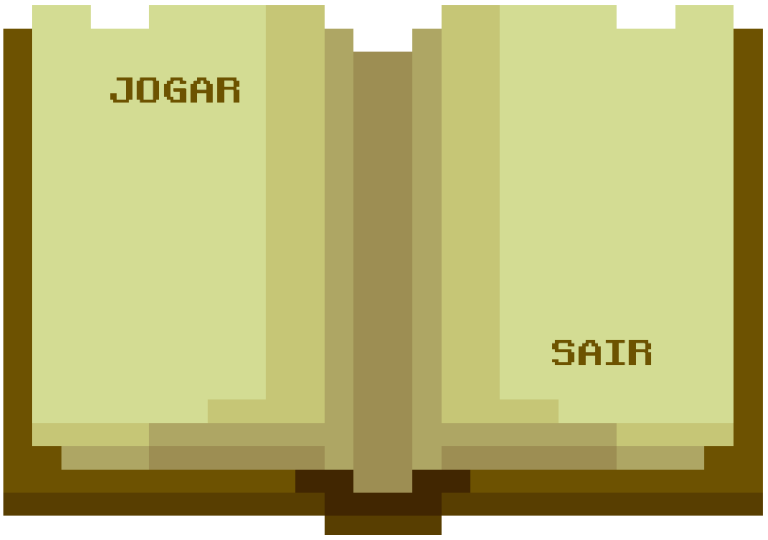
DIAGRAMA DE CASO DE USO



TELAS

(Menu inicial)



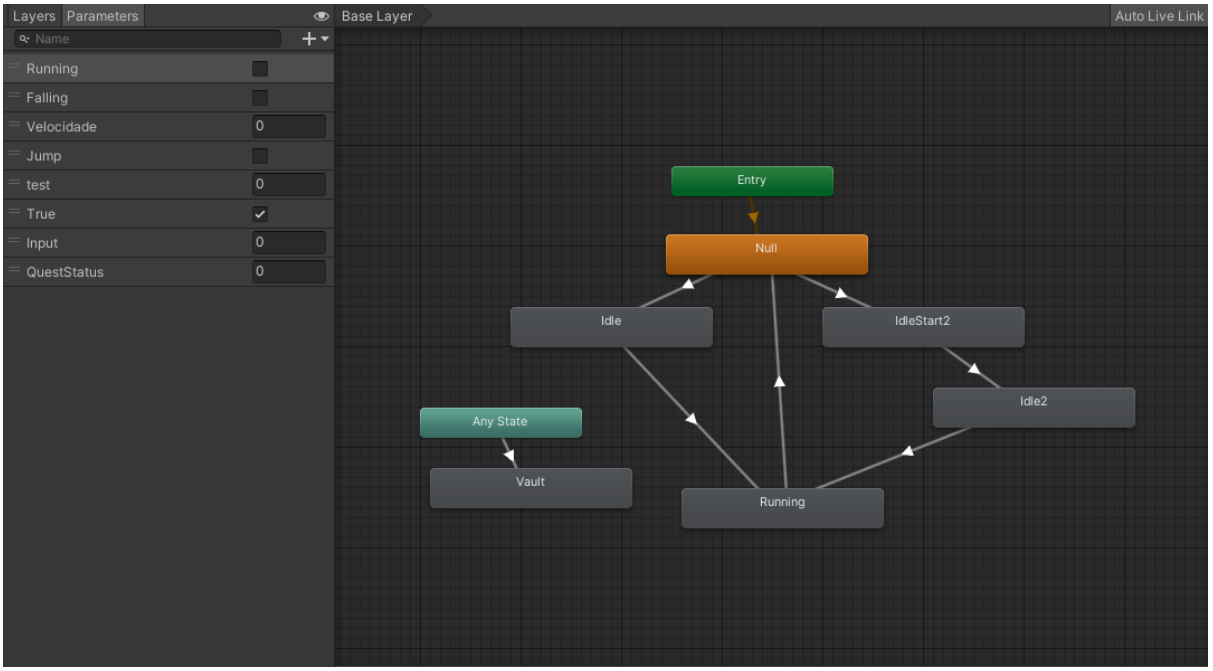


(em jogo)


NAME
Você ainda não tem permissão para entrar.
IGNORAR DIALOGO F
ANDAR A D
INTERAGIR E
Entrar
NAME
Poderia pegar uma planta para meus estudos de alquimia?
IGNORAR DIALOGO F
ANDAR A D
JOGO PAUSADO

Outras áreas relacionadas a Engine:

Controle de Animação:



CÓDIGOS:

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);
            }
        }
    }
}
```

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);
    }
}
```

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);
        }
    }



}
```

Codigo responsavel pela parte2 da primeira “quest do level:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class firstquest2 : MonoBehaviour
{
    [Header("Values")]
    [SerializeField] private int QuestID;
    [SerializeField] private string Namee;
    [SerializeField] private string Interact_Name;
    [SerializeField] private bool quest;
    [SerializeField] private bool teleporte;
    [SerializeField] private int CordX;
    [SerializeField] private int CordY;
    [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 GameObject Interact_Pop_up;
    [SerializeField] private TextMeshProUGUI Interact_Text;
    [SerializeField] private Transform QuestTracker;
    [SerializeField] private Transform Player;
    private int cache;
    private string clear;

    [Header("Dialogo")]
    [SerializeField] private string[] lines;
    [SerializeField] private float textSpeed;
    public int index;

    private void Start() {
        dialogtext.text = string.Empty;
    }

    private void OnTriggerEnter2D(Collider2D other){
        Interact_Text.text = Interact_Name;
        Interact_Pop_up.SetActive(true);
    }
```

```
    private void OnTriggerStay2D(Collider2D other){
        if(QuestCache.GetInteger("QuestStatus") >= QuestID &&
Input.GetKeyDown(KeyCode.E)){
            QuestCache.SetInteger("QuestStatus", QuestID + 1);
            DialogBox.SetActive(true);
            startdialoge();
        }
        if(Input.GetKeyDown(KeyCode.F)){
            DialogSkipper();
        }
    }

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

}
```

Codigo responsável por limitar os frames por segundo do jogo:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FrameLimiter : MonoBehaviour
{
    [SerializeField] private int MaxFps;

    void Start()
    {
        QualitySettings.vSyncCount = 0;
        Application.targetFrameRate = MaxFps;
    }
}
```

Código responsável por quaisquer outros objetos interativos do jogo:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class GenericInteraction : MonoBehaviour
{
    [Header("Values")]
    [SerializeField] private int QuestID;
    [SerializeField] private string Namee;
    [SerializeField] private string Interact_Name;
    [SerializeField] private bool quest;
    [SerializeField] private bool teleporte;
    [SerializeField] private int CordX;
    [SerializeField] private int CordY;
    [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 GameObject Interact_Pop_up;
    [SerializeField] private TextMeshProUGUI Interact_Text;
    [SerializeField] private Transform QuestTracker;
    [SerializeField] private Transform Player;
    private int cache;


    [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){
        Interact_Text.text = Interact_Name;
```

```
        Interact_Pop_up.SetActive(true);
    }

    private void OnTriggerStay2D(Collider2D other){
        if(QuestCache.GetInteger("QuestStatus") >= QuestID &&
Input.GetKeyDown(KeyCode.E)){
            Player.position = new Vector2(CordX, CordY);
        }else{
            if(QuestCache.GetInteger("QuestStatus") < QuestID &&
Input.GetKeyDown(KeyCode.E)){
            DialogBox.SetActive(true);
            startdialoge();
            }
        }
        if(Input.GetKeyDown(KeyCode.F)){
            DialogSkipper();
        }
    }

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

    private void startdialoge(){
        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);
        }
    }

}
```

Código responsável por quaisquer outras quests do jogo:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class GenericQuestLayout : 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);
        }

    }

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

```
            DialogSkipper();
        }
    }

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

    private void startdialoge(){
        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);
        }
    }

}
```

Código responsável pela barra de loading:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;

public class Loading : MonoBehaviour
{
    [Header("Componentes")]
    [SerializeField] Image LoadingBarFill;
    public int sceneID;

    void Start()
    {
        LoadScene(sceneID);
    }

    public void LoadScene(int sceneID){
        StartCoroutine(LoadSceneAsync(sceneID));
    }


    IEnumerator LoadSceneAsync(int sceneID){
        AsyncOperation operation = SceneManager.LoadSceneAsync(sceneID);

        while (!operation.isDone){
            float progressValue=Mathf.Clamp01(operation.progress / 0.9f);

            LoadingBarFill.fillAmount = progressValue;

            yield return null;
        }

    }
}
```

Código responsável pela movimentação das nuvens:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    private float velocidade;
    public float limiteDeDestruicao;
    public int right;

    private void Start(){
        velocidade = Random.Range(0.4f, 2.2f);
    }

    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);
            }
        }
    }
}
```

Código responsável pelo surgimento das nuvens no mapa:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Nuvem : MonoBehaviour
{
    [Header("Valores")]
    [SerializeField] private float Tempo_de_Spawn_Min;
    [SerializeField] private float Tempo_de_Spawn_Max;
    [Header("Componentes")]
    [SerializeField] private GameObject Nuvem1;
    [SerializeField] private GameObject Nuvem2;
    [SerializeField] private GameObject Nuvem3;
    [SerializeField] private GameObject Nuvem4;
    [SerializeField] private GameObject Nuvem5;
    private int Randomizer;
    void Start()
    {
        InvokeRepeating("Spawn_Nuvem", Random.Range(Tempo_de_Spawn_Min,
Tempo_de_Spawn_Max), Random.Range(Tempo_de_Spawn_Min,
Tempo_de_Spawn_Max));
    }
    private void Spawn_Nuvem(){
        Randomizer = Random.Range(0, 4);
        Vector2 posicao = new Vector2(transform.position.x, Random.Range(1, 6));
        if(Randomizer == 0){
            Instantiate(Nuvem1, posicao, Quaternion.identity);
        }else{
            if(Randomizer == 1){
                Instantiate(Nuvem2, posicao, Quaternion.identity);
            }else{
                if(Randomizer == 2){
                    Instantiate(Nuvem3, posicao, Quaternion.identity);
                }else{
                    if(Randomizer == 3){
                        Instantiate(Nuvem4, posicao, Quaternion.identity);
                    }
                    else{
                            Instantiate(Nuvem5, posicao, Quaternion.identity);
                    }
                }
            }
        }

    }

}
```

Código responsável por pausar o jogo:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Pause : MonoBehaviour
{
    private bool jogoPausado = false;
    [SerializeField] private GameObject PauseBox;

    void LateUpdate()
    {
        if (Input.GetKeyDown(KeyCode.P))
        {
            if (jogoPausado)
            {
                ResumeGame();
            }
            else
            {
                PauseGame();
            }
        }
    }

    void PauseGame()
    {
        PauseBox.SetActive(true);
        Time.timeScale = 0;
        jogoPausado = true;
    }

    void ResumeGame()
    {
        PauseBox.SetActive(false);
        Time.timeScale = 1;
        jogoPausado = false;
    }
}
```

Código responsável pela movimentação e animação do Player:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{

    [Header("Camadas")]
    [SerializeField] private LayerMask Chao;

    [Header("Variaveis de deteccao de variaveis")]
    [SerializeField] private float Raycast_Length;
    private bool No_chao;


    [Header("Variaveis")]
    [SerializeField] private float Aceleracao;
    [SerializeField] private float Velocidade_Maxima;
    [SerializeField] private float Resistencia;
    [SerializeField] private float Resistencia_no_Ar = 2f;
    [SerializeField] private float Forca_do_Pulo;
    [SerializeField] private float Multiplicador_de_Queda = 8f;
    [SerializeField] private float Multiplicador_de_Queda_low = 5f;
    private bool Pode_Pular => Input.GetButtonDown("Jump") && No_chao;
    private int RandomN;

    private float Direcao;
    private float Input_X;
    private bool Mudanca_de_direcao => (Corpo.velocity.x > 0f && Input_X < 0f) ||
(Corpo.velocity.x < 0f && Input_X > 0f);

    [Header("Componentes")]
    [SerializeField] private Rigidbody2D Corpo;
    [SerializeField] private Animator Anim;
    //[SerializeField] private AudioSource meuAudioSource;
    [SerializeField] private GameObject AudioSourcee;




    private void Update()
    {
        Entrada();
        Colisao_no_Chao();
```

```
        Animation();
    }

    private void FixedUpdate(){
        if(Anim.GetBool("Ocupied") == false){
            Movimentar();
        }
        if(Pode_Pular) Pulo();
        if(No_chao){
            Desaceleracao();
        }else{
            Desaceleracao_no_Ar();
            Multiplicador_de_Queda_fun();
        }
    }

    private void Entrada(){
        Input_X = Input.GetAxis("Horizontal");
    }

    private void Movimentar(){
        if(Input_X < 0){
            Corpo.AddForce(new Vector2(Input_X, 0f) * Aceleracao * 2);
        }else{
            Corpo.AddForce(new Vector2(Input_X, 0f) * Aceleracao);

        }

        if(Mathf.Abs(Corpo.velocity.x) > Velocidade_Maxima){
            Corpo.velocity = new Vector2(Mathf.Sign(Corpo.velocity.x) * Velocidade_Maxima,
Corpo.velocity.y);
        }
    }

    private void Desaceleracao(){
        if(Mudanca_de_direcao && No_chao){
            Corpo.drag = Resistencia;
        }
        else{
            Corpo.drag = 0f;
        }
    }

    private void Desaceleracao_no_Ar(){
        Corpo.drag = Resistencia_no_Ar;
    }

    private void Pulo(){
```

```
        Corpo.velocity = new Vector2(Corpo.velocity.x, Corpo.velocity.y);
        Corpo.AddForce(Vector2.up * Forca_do_Pulo, ForceMode2D.Impulse);
    }

    private void Multiplicador_de_Queda_fun(){
        if (Corpo.velocity.x < 0){
            Corpo.gravityScale = Multiplicador_de_Queda;
        }
        else if (Corpo.velocity.y > 0 && !Input.GetButton("Jump")){
            Corpo.gravityScale = Multiplicador_de_Queda_low;
        }else{
            Corpo.gravityScale = 1f;
        }
    }

    private void Colisao_no_Chao(){
        No_chao = Physics2D.Raycast(transform.position * Raycast_Length, Vector2.down,
Raycast_Length, Chao);
    }

    private void OnDrawGizmos(){
        Gizmos.color = Color.green;
        Gizmos.DrawLine(transform.position, transform.position + Vector3.down *
Raycast_Length);
    }

    private void Animation(){

        if (Input_X == 0f){
            RandomN = Random.Range(0, 2);
            Anim.SetInteger("test", RandomN);
        }

        if (Input_X != 0f){
            Anim.SetBool("Running", true);
            Anim.SetInteger("Input", 1);
            AudioSourcee.SetActive(true);
        }else{
            Anim.SetBool("Running", false);
            Anim.SetInteger("Input", 0);
            AudioSourcee.SetActive(false);
        }

        if (Corpo.velocity.y > 0.5f){
            Anim.SetBool("Falling", true);
        }
        if (Pode_Pular){
            Anim.SetBool("Jump", true);
```

```
        }else{
            Anim.SetBool("Jump", false);
        }
        if (Corpo.velocity.x > 0.1f) {
            transform.localScale = new Vector3(1.3f, 1.3f, 1.3f);
        } else if (Corpo.velocity.x < -0.1f) {
            transform.localScale = new Vector3(-1.3f, 1.3f, 1.3f);
        }

        Anim.SetFloat("Velocidade", Mathf.Abs(Corpo.velocity.x)/10 + 0.6f);
    }

}
```

Código responsável por mudar a posição do player caso ele atinja uma área não acessível do mapa:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Reset : MonoBehaviour
{


    [SerializeField] private Transform Player;

    private void OnTriggerEnter2D(Collider2D other) {
        Player.position = new Vector2(0f,0f);
    }
}
```

Código responsável por salvar os dados de progresso do Jogador:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Save : MonoBehaviour
{


    [SerializeField] private Transform Playerr;
    [SerializeField] private Animator QuestIDD;
    private Vector2 Playerposs;
    private float PPosx = 0;
    private float PPosy = 0;
    private int QuestIDE = 0;

    void Start()
    {
        if(PlayerPrefs.HasKey("LastQuest")){
            QuestIDE = PlayerPrefs.GetInt("LastQuest");
            PPosx = PlayerPrefs.GetFloat("PlayerPosition_X");
            PPosy = PlayerPrefs.GetFloat("PlayerPosition_Y");
            Playerposs = new Vector2(PPosx, PPosy);
            QuestIDD.SetInteger("QuestStatus", QuestIDE);
            Playerr.position = Playerposs;
        }
    }

    void LateUpdate()
    {
        Playerposs = (Vector2)Playerr.position;
        QuestIDE = QuestIDD.GetInteger("QuestStatus");
        PPosx = Playerposs.x;
        PPosy = Playerposs.y;
        PlayerPrefs.SetInt("LastQuest", QuestIDE);
        PlayerPrefs.SetFloat("PlayerPosition_X", PPosx);
        PlayerPrefs.SetFloat("PlayerPosition_Y", PPosy);
        PlayerPrefs.Save();
    }
}
```

Código responsável pela animação de “digitação” dos textos:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;


public class TextDialogOnly : MonoBehaviour
{
    [Header("Dialogo")]
    [SerializeField] private string[] lines;
    [SerializeField] private float textSpeed;
    private int index;
    [SerializeField] private TextMeshProUGUI dialogtext;


    void Start()
    {
        startdialoge();
    }

    private void startdialoge(){
        index = 0;
        StartCoroutine(TypeLine());
    }

    IEnumerator TypeLine(){
        foreach (char c in lines[index].ToCharArray()){
            dialogtext.text += c;
            yield return new WaitForSeconds(textSpeed);
        }
    }


}
```

Codigo responsavel pela mecanica de “vault”:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Vault : MonoBehaviour
{

    [Header("Valores")]
    [SerializeField] private float Speed = 8;
    [SerializeField] private float Forca_do_Pulo = 12;

    [Header("Componentes")]
    [SerializeField] private Rigidbody2D Player;
    [SerializeField] private Animator Player_Anim;
    private float Player_speed;
    private float cache;

    private void OnTriggerEnter2D(Collider2D other){
        Player_speed = Player.velocity.x;
        Player_Anim.SetBool("Ocupied", true);
        Player_Anim.SetTrigger("Vault");
        if (Player_speed > 0){
            cache = 1;
        }else{
            cache = -1;
        }
        Pulo();
     }

    private void OnTriggerStay2D(Collider2D other){
        Player_Anim.SetBool("Ocupied", true);
        Player.velocity = new Vector2(Speed * cache, Player.velocity.y);
    }

    private void OnTriggerExit2D(Collider2D other){
        Player_Anim.SetBool("Ocupied", false);
    }


    private void Pulo(){
        Player.velocity = new Vector2(Player.velocity.x, Player.velocity.y);
        Player.AddForce(Vector2.up * Forca_do_Pulo, ForceMode2D.Impulse);
    }
}
```
