//Скрипт на Анимацию бега

using UnityEngine;

[RequireComponent(typeof(SpriteRenderer))]

public class AnimatedSprite : MonoBehaviour

{

public Sprite[] sprites;

private SpriteRenderer spriteRenderer;

private int frame;

private void Awake()

{

spriteRenderer = GetComponent<SpriteRenderer>();

}

private void OnEnable()

{

Invoke(nameof(Animate), 0f);

}

private void OnDisable()

{

CancelInvoke();

}

private void Animate()

{

frame++;

if (frame >= sprites.Length) { // Условие на переключение 2 картинок, вызывая анимацию.

frame = 0;

}

if (frame >= 0 && frame < sprites.Length) {

spriteRenderer.sprite = sprites[frame];

}

Invoke(nameof(Animate), 1f / GameManager.Instance.gameSpeed);

}

}

using UnityEngine;

using UnityEngine.UI;

public class OpenTextFile : MonoBehaviour

{

public Button yourButton;

void Start()

{

Button btn = yourButton.GetComponent<Button>();

btn.onClick.AddListener(TaskOnClick);

}

void TaskOnClick()

{

Application.OpenURL("\"D:\\SandRacingкопия\\Spravka.chm\""); // Открытие справки

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using TMPro;

public class GameManagers : MonoBehaviour

{

public GameObject[] spawnObjects;

public GameObject spawnObject;

public GameObject[] spawnPoints;

public GameObject spawnPoint;

public float timer;

public float timeBetweenSpawn;

public int yournum;

public float speedMultiplier;

public TMP\_Text scoreUI;

public float score;

public float scoreGoal;

void Start()

{

}

void Update()

{

scoreUI.text = Mathf.FloorToInt(score).ToString("D5"); // Начисление учков с учётом изменения скорости

speedMultiplier += Time.deltaTime \* 0.1f;

score += Time.deltaTime \* speedMultiplier;

timer += Time.deltaTime;

if(timer > timeBetweenSpawn)

{

timer = 0;

int randNum = Random.Range(0, yournum);

Instantiate(spawnObjects[randNum], spawnPoints[randNum].transform.position, Quaternion.identity);

}

Flag();

}

public void Flag() // После набранных очков появляется префаб флага

{

if(score >= scoreGoal)

{

Instantiate(spawnObject, spawnPoint.transform.position, Quaternion.identity);

score = 0;

timeBetweenSpawn = 10;

scoreUI.enabled = false;

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Playables;

using UnityEngine.SceneManagement;

public class CutSceneManager : MonoBehaviour // Запуск катсцены после прохождения левела

{

private PlayableDirector director;

void Awake()

{

director = GetComponent<PlayableDirector>();

}

public void Start()

{

director.Play();

StartCoroutine(LoadMainMenu());

}

IEnumerator LoadMainMenu()

{

yield return new WaitForSeconds(5);

SceneManager.LoadScene(1);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Loading : MonoBehaviour // Возвращает на сцену с его номером

{

private Animator anim;

public int Num;

void Awake()

{

anim = GetComponent<Animator>();

}

public void LoadScene()

{

anim.SetBool("loading", false);

SceneManager.LoadScene(Num);

}

}

using UnityEngine;

using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour // Возвращает на выбранную сцену

{

public int Num;

public void Next()

{

Time.timeScale = 1f;

}

public void Reload()

{

SceneManager.LoadScene(Num);

}

public void ReturntoMainMenu()

{

SceneManager.LoadScene(0);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class PlayerCosmos : MonoBehaviour

{

private Rigidbody2D rb;

private bool isGrounded;

bool facingRight = true;

public GameObject PauseM;

void Awake()

{

rb = GetComponent<Rigidbody2D>();

}

void Update()

{

if (Input.GetButtonDown("Jump") && isGrounded) // Скрипт на прыжок и поворот

{

rb.gravityScale \*= -1;

Flip();

}

}

private void PauseWork() // Пауза при смерти

{

PauseM.SetActive(true);

Time.timeScale = 0;

}

private void OnCollisionEnter2D(Collision2D other) // Метка для включения пола

{

if (other.gameObject.CompareTag("Ground"))

{

isGrounded = true;

}

}

private void OnCollisionExit2D(Collision2D other) // Метка для выключения пола

{

if (other.gameObject.CompareTag("Ground"))

{

isGrounded = false;

}

}

private void OnTriggerEnter2D(Collider2D other) // Пауза при смерти и кнопка для возврата в гл меню

{

if (other.gameObject.CompareTag("Enemy"))

{

PauseWork();

}

if (other.gameObject.CompareTag("Finale"))

{

SceneManager.LoadScene(0);

}

}

public void Flip() // Вращение игрока на 180 градусов при прыжке

{

facingRight = !facingRight;

transform.Rotate(180, 0, 0);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SpawnObjectScript : MonoBehaviour

{

private Rigidbody2D rb;

public float speed;

private GameManagers gms;

private float timer;

void Start()

{

gms = GameObject.FindGameObjectWithTag("GameManager").GetComponent<GameManagers>();

rb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update() // Спавн объект

{

timer += Time.deltaTime;

if (timer > 7)

{

Destroy(gameObject);

}

rb.velocity = Vector2.left \* (speed + gms.speedMultiplier);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class PlayerMovementFly : MonoBehaviour

{

public float jump;

private Rigidbody2D rb;

public GameObject PauseM;

void Awake()

{

rb = GetComponent<Rigidbody2D>();

}

void Update() // Прыжок в воздухе

{

if (Input.GetButtonDown("Jump"))

{

rb.AddForce(Vector2.up \* jump);

}

}

private void PauseWork()

{

PauseM.SetActive(true);

Time.timeScale = 0;

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.gameObject.CompareTag("Enemy"))

{

PauseWork();

}

if (other.gameObject.CompareTag("Finale"))

{

SceneManager.LoadScene(7);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DontDestroyMusic : MonoBehaviour

{

public static DontDestroyMusic instance; // Условие без удаления музыки

void Start()

{

if (instance != null)

{

Destroy(gameObject);

}

else

{

instance = this;

DontDestroyOnLoad(gameObject);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Audio;

public class VolumeInit : MonoBehaviour

{

public string volumeParameter = "MasterVolume"; // Стартовое значение музыки

public AudioMixer mixer;

void Start()

{

var volumeValue = PlayerPrefs.GetFloat(volumeParameter, volumeParameter == "MusicVol" ? 0f : -80f);

mixer.SetFloat(volumeParameter, volumeValue);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine.UI;

using UnityEngine;

using UnityEngine.Audio;

public class VolumeControle : MonoBehaviour

{

public string volumeParameter = "MasterVolume";

public AudioMixer mixer;

public Slider slider;

private const float \_multiplier = 20f;

private float \_volumeValue;

private void Awake()

{

slider.onValueChanged.AddListener(HandleSliderValueChanged);

}

private void HandleSliderValueChanged(float value) // Скрипт для смещения слайдер ( изменение громкости )

{

\_volumeValue = Mathf.Log10(value) \* \_multiplier;

mixer.SetFloat(volumeParameter, \_volumeValue);

}

void Start()

{

\_volumeValue = PlayerPrefs.GetFloat(volumeParameter, Mathf.Log10(slider.value) \* \_multiplier);

slider.value = Mathf.Pow(10f, \_volumeValue / \_multiplier);

}

private void OnDisable()

{

PlayerPrefs.SetFloat(volumeParameter, \_volumeValue);

}

}

using TMPro;

using UnityEngine;

using UnityEngine.SocialPlatforms.Impl;

using UnityEngine.UI;

// Настройка Игрока и других компонентов

[DefaultExecutionOrder(-1)]

public class GameManager : MonoBehaviour

{

public static GameManager Instance { get; private set; }

public float initialGameSpeed = 5f;

public float gameSpeedIncrease = 0.1f;

public float gameSpeed { get; private set; }

[SerializeField] private TextMeshProUGUI scoreText;

[SerializeField] private TextMeshProUGUI hiscoreText;

[SerializeField] private TextMeshProUGUI gameOverText;

[SerializeField] private Button retryButton;

[SerializeField] private Button cancelButton;

private Player player;

private Spawner spawner;

public float score;

public float Score => score;

private void Awake()

{

if (Instance != null)

{

DestroyImmediate(gameObject);

}

else

{

Instance = this;

}

}

private void OnDestroy()

{

if (Instance == this)

{

Instance = null;

}

}

private void Start()

{

player = FindObjectOfType<Player>();

spawner = FindObjectOfType<Spawner>();

NewGame();

}

public void NewGame()

{

Obstacle[] obstacles = FindObjectsOfType<Obstacle>();

foreach (var obstacle in obstacles)

{

Destroy(obstacle.gameObject);

}

score = 0f;

gameSpeed = initialGameSpeed;

enabled = true;

player.gameObject.SetActive(true);

spawner.gameObject.SetActive(true);

gameOverText.gameObject.SetActive(false);

retryButton.gameObject.SetActive(false);

cancelButton.gameObject.SetActive(false);

UpdateHiscore();

}

public void GameOver()

{

gameSpeed = 0f;

enabled = false;

player.gameObject.SetActive(false);

spawner.gameObject.SetActive(false);

gameOverText.gameObject.SetActive(true);

retryButton.gameObject.SetActive(true);

cancelButton.gameObject.SetActive(true);

UpdateHiscore();

}

private void Update()

{

gameSpeed += gameSpeedIncrease \* Time.deltaTime;

score += gameSpeed \* Time.deltaTime;

scoreText.text = Mathf.FloorToInt(score).ToString("D5");

}

private void UpdateHiscore()

{

float hiscore = PlayerPrefs.GetFloat("hiscore", 0);

if (score > hiscore)

{

hiscore = score;

PlayerPrefs.SetFloat("hiscore", hiscore);

}

hiscoreText.text = Mathf.FloorToInt(hiscore).ToString("D5");

}

}

using UnityEngine;

[RequireComponent(typeof(MeshRenderer))]

public class Ground : MonoBehaviour

{

private MeshRenderer meshRenderer;

private void Awake()

{

meshRenderer = GetComponent<MeshRenderer>();

}

private void Update()

{

float speed = GameManager.Instance.gameSpeed / transform.localScale.x; // Анимация на песок с учётом скорости

meshRenderer.material.mainTextureOffset += speed \* Time.deltaTime \* Vector2.right;

}

}

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour

{

public void Runner() // Загрузка сцен левелов

{

SceneManager.LoadScene(2);

}

public void PlayLevel1()

{

SceneManager.LoadScene(3);

}

public void PlayLevel2()

{

SceneManager.LoadScene(4);

}

public void PlayLevel3()

{

SceneManager.LoadScene(5);

}

public void PlayLevel4()

{

SceneManager.LoadScene(6);

}

public void ExitGame()

{

Debug.Log("Игра закрылась");

Application.Quit();

}

}

using UnityEngine;

public class Obstacle : MonoBehaviour

{

private float leftEdge; // Метка на препятствие

private void Start()

{

leftEdge = Camera.main.ScreenToWorldPoint(Vector3.zero).x - 2f;

}

private void Update()

{

transform.position += GameManager.Instance.gameSpeed \* Time.deltaTime \* Vector3.left;

if (transform.position.x < leftEdge) {

Destroy(gameObject);

}

}

}

using UnityEngine;

using TMPro;

using UnityEngine.SceneManagement;

// Настройка игрока в забеге , высота прыжка, гравитация, загрузка главного меню при смерти.

[RequireComponent(typeof(CharacterController))]

public class Player : MonoBehaviour

{

private CharacterController character;

private Vector3 direction;

[SerializeField] private TextMeshProUGUI scoreText;

public float jumpForce = 8f;

public float gravity = 9.81f \* 2f;

private void Awake()

{

character = GetComponent<CharacterController>();

}

private void OnEnable()

{

direction = Vector3.zero;

}

private void Update()

{

direction += gravity \* Time.deltaTime \* Vector3.down;

if (character.isGrounded)

{

direction = Vector3.down;

if (Input.GetButton("Jump"))

{

direction = Vector3.up \* jumpForce;

}

}

character.Move(direction \* Time.deltaTime);

}

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Obstacle"))

{

FindObjectOfType<GameManager>().GameOver();

}

}

private void OnCollisionEnter2D(Collision2D other)

{

if (other.gameObject.CompareTag("Finale"))

{

SceneManager.LoadScene(0);

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

// Скрипт для настроек и сохранение их

public class Settings : MonoBehaviour

{

public Dropdown resolutionDropdown;

public Dropdown qualityDropdown;

Resolution[] resolutions;

void Start()

{

resolutionDropdown.ClearOptions();

List<string> options = new List<string>();

resolutions = Screen.resolutions;

int currentResolutionIndex = 0;

for (int i = 0; i < resolutions.Length; i++)

{

string option = resolutions[i].width + "x" + resolutions[i].height + " " + resolutions[i].refreshRateRatio + "Hz";

options.Add(option);

if (resolutions[i].width == Screen.currentResolution.width && resolutions[i].height == Screen.currentResolution.height)

currentResolutionIndex = i;

}

resolutionDropdown.AddOptions(options);

resolutionDropdown.RefreshShownValue();

LoadSettings(currentResolutionIndex);

}

public void SetFullscreen(bool isFullscreen)

{

Screen.fullScreen = isFullscreen;

}

public void SetResolution (int resolutionIndex)

{

Resolution resolution = resolutions[resolutionIndex];

Screen.SetResolution(resolution.width, resolution.height, Screen.fullScreen);

}

public void SetQuality(int qualityIndex)

{

QualitySettings.SetQualityLevel(qualityIndex);

}

public void SaveSettings()

{

PlayerPrefs.SetInt("QualitySettingPreference", qualityDropdown.value);

PlayerPrefs.SetInt("ResolutionPreference", resolutionDropdown.value);

PlayerPrefs.SetInt("FullscreenPreference", System.Convert.ToInt32(Screen.fullScreen));

}

public void LoadSettings(int currentResolutionIndex)

{

if (PlayerPrefs.HasKey("QualitySettingPreference"))

qualityDropdown.value = PlayerPrefs.GetInt("QualitySettingsPreference");

else

qualityDropdown.value = 3;

if (PlayerPrefs.HasKey("ResolutionPreference"))

resolutionDropdown.value = PlayerPrefs.GetInt("ResolutionPreference");

else

resolutionDropdown.value = currentResolutionIndex;

if (PlayerPrefs.HasKey("FullscrenPreference"))

Screen.fullScreen = System.Convert.ToBoolean(PlayerPrefs.GetInt("FullscreenPreference"));

else

Screen.fullScreen = true;

}

}

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.SocialPlatforms.Impl;

// Спаунер на флаг

public class SpawnFlag : MonoBehaviour

{

public GameObject FlagPref;

public GameObject SpawnFlags;

[SerializeField] private TextMeshProUGUI scoreText;

public float gameSpeedIncrease = 0.1f;

public float gameSpeed { get; private set; }

public float initialGameSpeed = 5f;

public float score;

public float scoreG;

public float Score => score;

private void Start()

{

gameSpeed = initialGameSpeed;

}

void Update()

{

gameSpeed += gameSpeedIncrease \* Time.deltaTime;

score += gameSpeed \* Time.deltaTime;

if (score >= scoreG)

{

Instantiate(FlagPref, SpawnFlags.transform.position, Quaternion.identity);

score = 0;

scoreText.enabled = false;

}

}

}