**Лабораторное занятие № 25.**

**Тема: Разработка игры “ Bounce”**

**Цель: Разработка игры “ Bounce”**

**Ход работы:**

1.Перекидываю ресурсы (Asset)

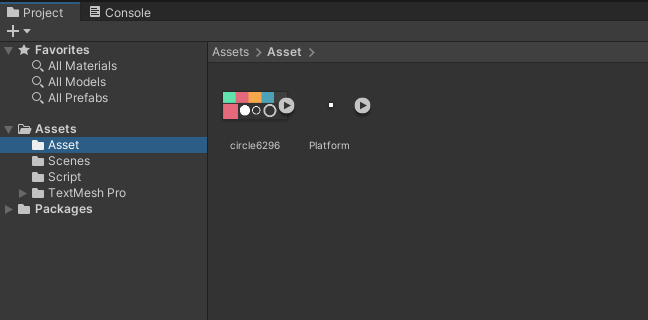


Рис.25.1 – Папка Asset

2.Создание игрока (Ball)

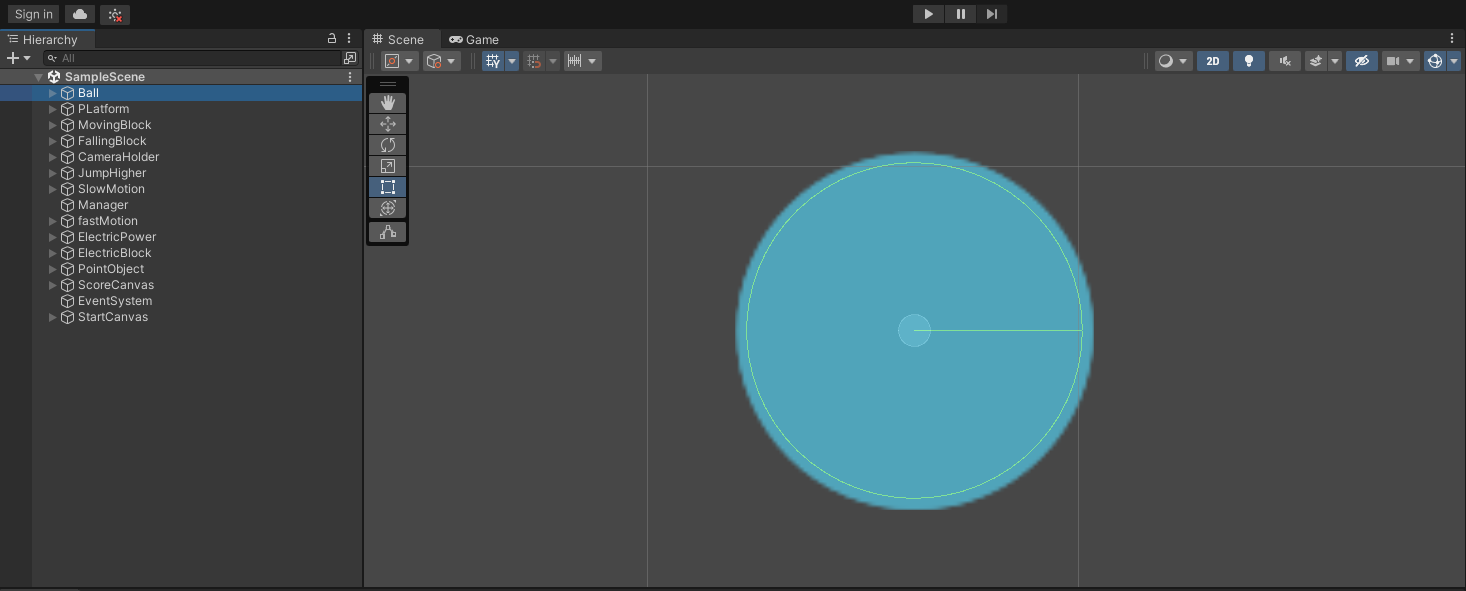


Рис.25.2 – Ball.

3.Инспектор игрока (Ball).

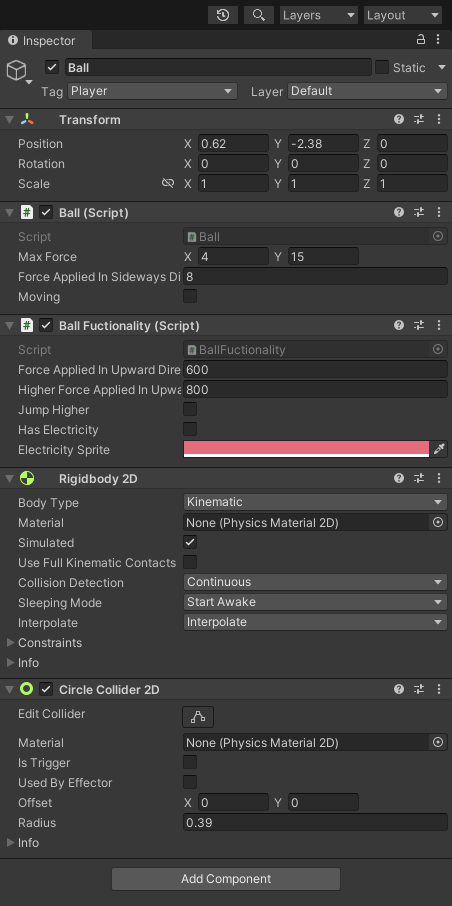


Рис.25.3 – Inspector Ball.

4.Создание платформ.

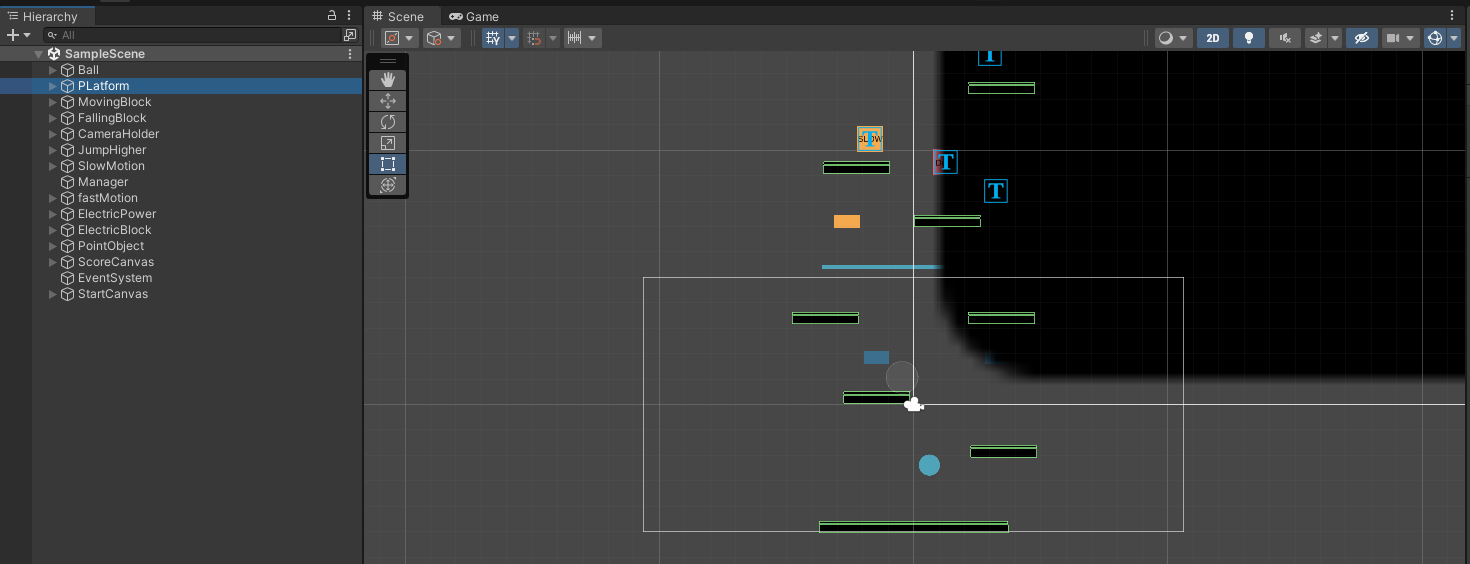


Рис.25.4 – Platform.

5.Создание движущейся платформы.

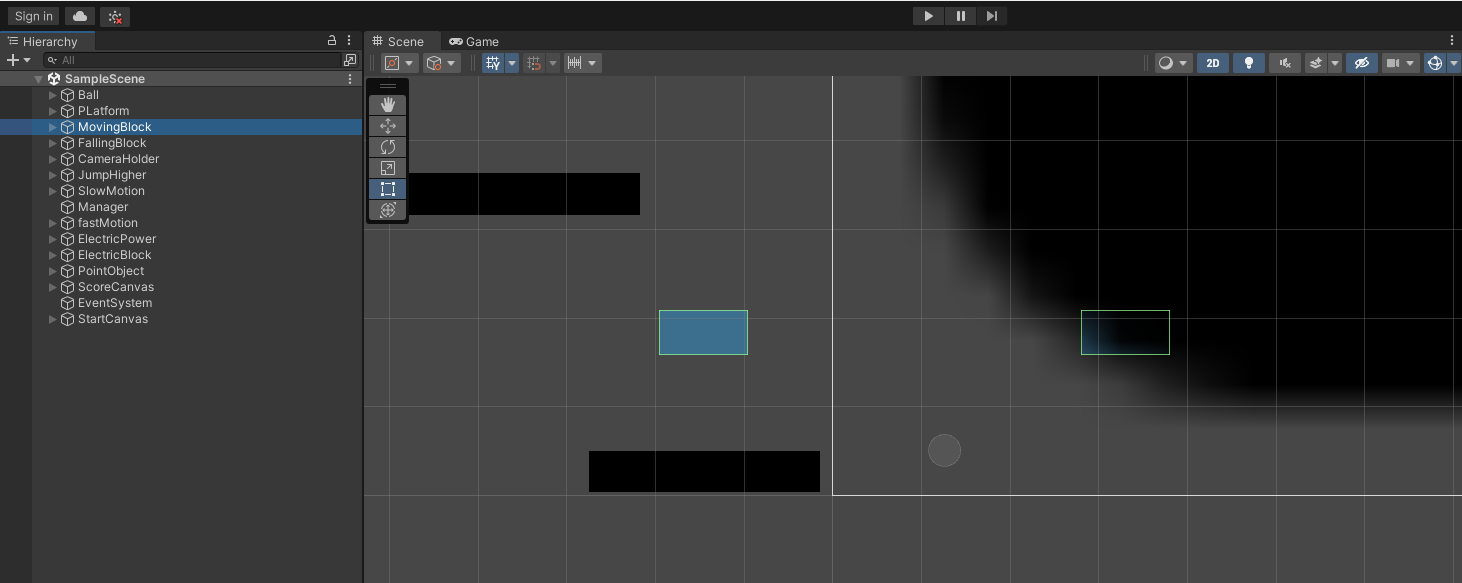


Рис.25.5 – MovingBlock.

6.Создание бустов.

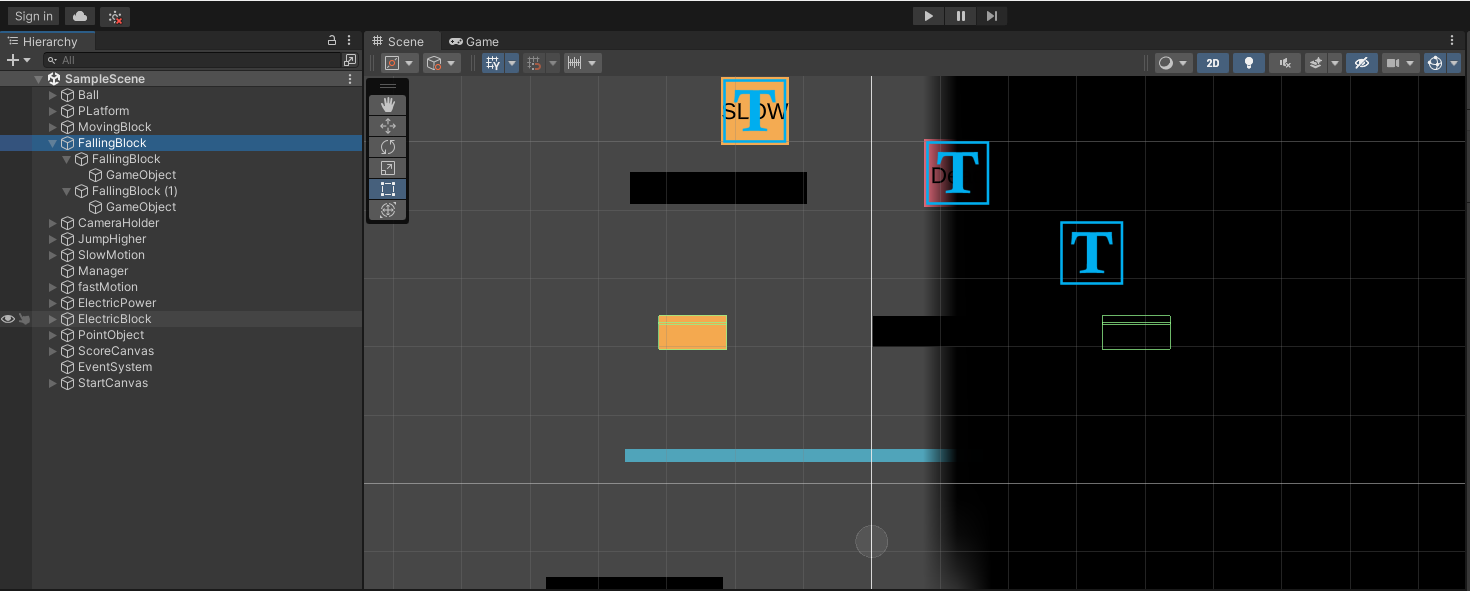


Рис.25.6 –FallingBlock.

7.Привязка камеры к игроку (CameraHolder)

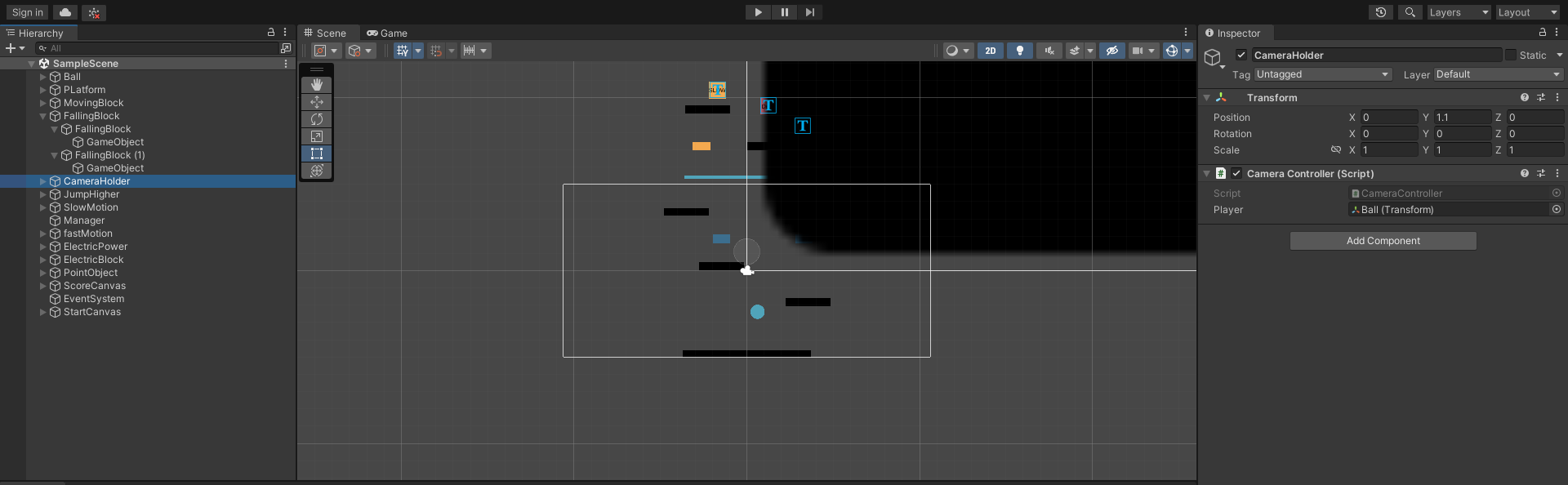


Рис.25.7 –CameraController.

8.Создание меню и очков.

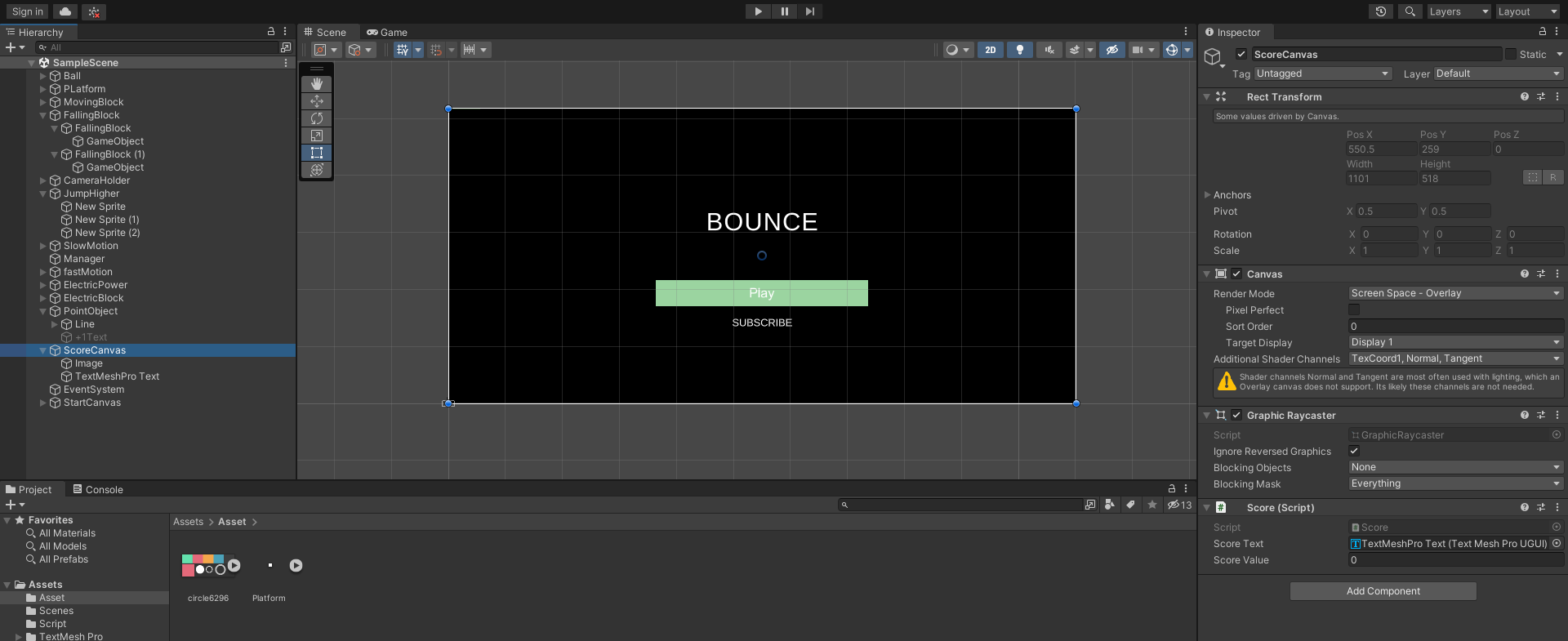


Рис.25.8 –ScoreCanvas.

9.Создание скриптов.

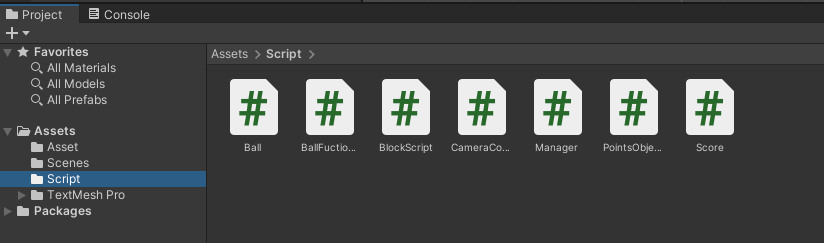


Рис.25.9 –Scripts.

10.Итог работы.

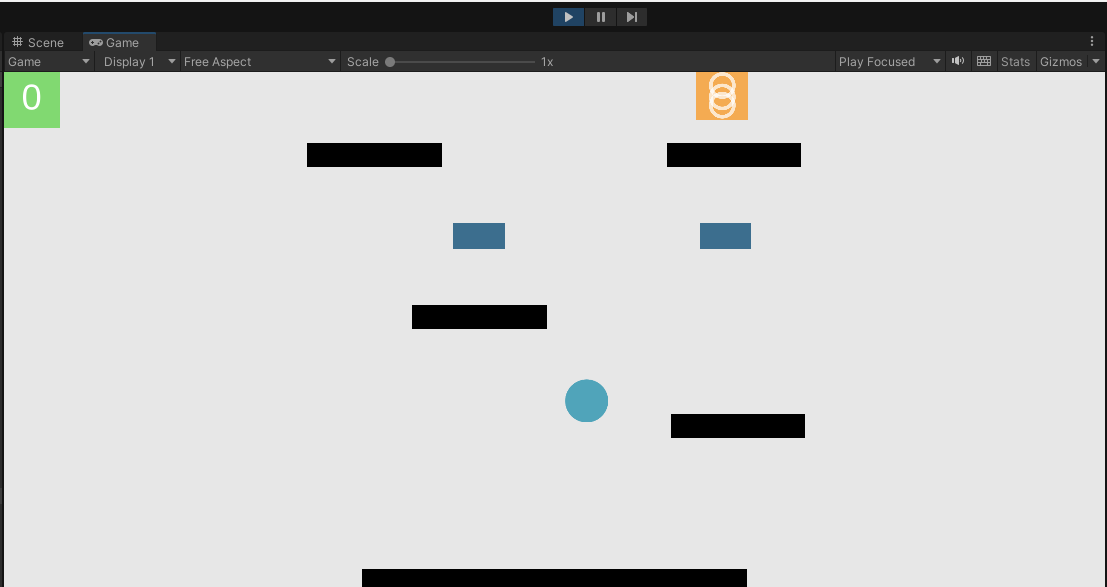


Рис.25.10 –Итог работы.

**Скрипты:**

Ball.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Ball : MonoBehaviour

{ Rigidbody2D rgbd;

public Vector2 maxForce;

public float forceAppliedInSidewaysDirection;

public bool moving;

float sidewaysMovement;

BallFuctionality ballFuctionality;

// Start is called before the first frame update

void Start()

{

rgbd = GetComponent<Rigidbody2D>();

ballFuctionality = GetComponent<BallFuctionality>();

}

// Update is called once per frame

void Update()

{

// clampVeclocity();

// Debug.Log(rgbd.velocity);

movement();

}

public void FixedUpdate()

{

if (!moving)

{

Vector2 vel = rgbd.velocity;

vel.x = Mathf.Lerp(vel.x, 0, 0.2f);

rgbd.velocity = vel;

}

}

void clampVeclocity()

{

Vector2 vel = rgbd.velocity;

if (Mathf.Abs(vel.x) >= maxForce.x)

{

vel.x = maxForce.x \* Mathf.Sign(rgbd.velocity.x);

}

if ((vel.y) >= maxForce.y)

{ if (ballFuctionality.jumpHigher) return;

vel.y = maxForce.y;

}

rgbd.velocity = vel;

}

void movement()

{

if (Input.GetKey(KeyCode.LeftArrow))

{

rgbd.AddForce(new Vector2(-forceAppliedInSidewaysDirection, 0));

}

if (Input.GetKey(KeyCode.RightArrow))

{

rgbd.AddForce(new Vector2(forceAppliedInSidewaysDirection, 0));

}

if (Input.GetKeyDown(KeyCode.RightArrow) || Input.GetKeyDown(KeyCode.LeftArrow))

{

moving = true;

}

if (Input.GetKeyUp(KeyCode.RightArrow) || Input.GetKeyUp(KeyCode.LeftArrow))

{

moving = false;

}

}

}

BallFunctionality.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BallFuctionality : MonoBehaviour

{

Rigidbody2D rgbd;

public float forceAppliedInUpwardDirection, higherForceAppliedInUpwardDirection;

public bool jumpHigher;

Manager manager;

public bool hasElectricity;

SpriteRenderer sr;

public Color electricitySprite;

Color originalColor;

Score score;

// Start is called before the first frame update

void Start()

{

rgbd = GetComponent<Rigidbody2D>();

sr = GetComponentInChildren<SpriteRenderer>();

score = FindObjectOfType<Score>();

originalColor = sr.color;

manager = FindObjectOfType<Manager>();

}

public void OnCollisionEnter2D(Collision2D collision)

{ if (!manager.startGame) return;

if (collision.collider.tag == "JumpingBlock")

{

if (jumpHigher)

{

Debug.Log("Applying higher force");

rgbd.AddForce(new Vector2(0, higherForceAppliedInUpwardDirection));

jumpHigher = false;

}

else

{

rgbd.AddForce(new Vector2(0, forceAppliedInUpwardDirection));

}

}

if (collision.collider.tag == "ElectricityBlock")

{ //gameover

if (hasElectricity) {

rgbd.AddForce(new Vector2(0, forceAppliedInUpwardDirection)); return; }

else

{

Debug.Log("Dead");

manager.RestartTheGame();

}

}

}

public void OnTriggerEnter2D(Collider2D collision)

{

if (!manager.startGame) return;

if (collision.tag == "spring")

{

jumpHigher = true;

Destroy(collision.gameObject);

}

if (collision.tag == "TimeSlower")

{

//call function to slow the game

manager.slowTimerStart();

Destroy(collision.gameObject);

}if (collision.tag == "TimeFaster")

{

//call function to slow the game

manager.FastTimerStart();

Destroy(collision.gameObject);

}if (collision.tag == "ElectricityPower")

{

//change the sprite to charge sprite

//screen flashes

Destroy(collision.gameObject);

StartCoroutine(electricity());

}

if (collision.tag == "PointObject")

{

//add point

if (!collision.GetComponent<PointsObject>().hasBroke)

{

score.AddScore();

}

collision.GetComponent<PointsObject>().Explode();

}

}

IEnumerator electricity()

{

hasElectricity = true;

sr.color = electricitySprite;

yield return new WaitForSeconds(2f);

hasElectricity = false;

sr.color = originalColor;

}

}

BlockScript.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BlockScript : MonoBehaviour

{

public bool moveBlock, fallingBlock;

bool startMovingBlock, startFallingBlock;

Rigidbody2D rgbd;

bool moveLeft;

// Start is called before the first frame update

void Start()

{

rgbd = GetComponent<Rigidbody2D>();

if (Random.value > 0.5)

{

moveLeft = true;

}

}

// Update is called once per frame

void Update()

{

if (startMovingBlock && moveBlock)

{

if (moveLeft)

{

rgbd.velocity = new Vector2(-3, 0);

}

else

{

rgbd.velocity = new Vector2(3, 0);

}

}

}

public void OnCollisionEnter2D(Collision2D collision)

{ if(collision.collider.tag == "Player") {

if (moveBlock)

{

startMovingBlock = true;

}

else if (fallingBlock)

{

rgbd.bodyType = RigidbodyType2D.Dynamic;

}

}

}

}

CameraController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraController : MonoBehaviour

{

public Transform player;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

if (player.position.y > transform.position.y)

{

transform.position = new Vector3(transform.position.x, player.position.y, -10);

}

}

}

Manager.cs

using System.Collections;

using UnityEngine.SceneManagement;

using UnityEngine;

public class Manager : MonoBehaviour

{

public bool slowTime, fastTime;

public float slow;

public bool startGame;

public void slowTimerStart()

{

StartCoroutine(slowTheTime());

}

IEnumerator slowTheTime()

{

slowTime = true;

Time.timeScale = 1 / slow;

Time.fixedDeltaTime = Time.fixedDeltaTime / slow;

FindObjectOfType<BallFuctionality>().forceAppliedInUpwardDirection \*= 2;

yield return new WaitForSeconds(2 \* slow);

Time.timeScale = 1;

Time.fixedDeltaTime = Time.fixedDeltaTime \* slow;

FindObjectOfType<BallFuctionality>().forceAppliedInUpwardDirection /= 2;

slowTime = false;

}

public void FastTimerStart()

{

StartCoroutine(fastTheTime());

}

IEnumerator fastTheTime()

{

fastTime = true;

Time.timeScale = 1 \* slow;

Time.fixedDeltaTime = Time.fixedDeltaTime \* slow;

FindObjectOfType<BallFuctionality>().forceAppliedInUpwardDirection /= 2;

yield return new WaitForSeconds(2 \* slow);

Time.timeScale = 1;

Time.fixedDeltaTime = Time.fixedDeltaTime / slow;

FindObjectOfType<BallFuctionality>().forceAppliedInUpwardDirection \*= 2;

fastTime = false;

}

public void StartTheGame()

{

startGame = true;

FindObjectOfType<BallFuctionality>().GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Dynamic;

}

public void RestartTheGame()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

}

}

PointsObject.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PointsObject : MonoBehaviour

{

Transform[] childObj;

public GameObject Points1;

public int force;

public bool hasBroke;

public void Explode()

{

hasBroke = true;

Points1.SetActive(true);

Points1.GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Dynamic;

Points1.GetComponent<Rigidbody2D>().AddForceAtPosition(Vector2.up \* force \* 3 / 2, Points1.transform.position);

foreach (Transform t in transform)

{

Rigidbody2D rgbd = t.GetComponent<Rigidbody2D>();

if (rgbd != null)

{

//shake camera

rgbd.bodyType = RigidbodyType2D.Dynamic;

rgbd.AddForceAtPosition(Vector2.up \* force,

new Vector2(Random.Range(t.position.x - 2, t.position.x + 2), t.position.y));

//play sound

}

}

}

}

Score.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Score : MonoBehaviour

{

public TMPro.TextMeshProUGUI scoreText;

public int ScoreValue;

public void AddScore()

{

ScoreValue++;

scoreText.text = ScoreValue.ToString();

}

}