Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1 - 2

за 5 семестр

По дисциплине: «ОСиСП»

Выполнил:

Студент 2 курса

Группы ПО-4(1)

Иваненко И. Л.

Проверил:

Дряпко А. В.

2021

Цель работы: приобрести практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом.

Общее задание

1) Выбрать тему из перечисленных ниже или предложить свою (тематика – игры, системные программы и утилиты для ОС Windows);

2) Вписать свою фамилию напротив выбранной темы в файле;

3) Разработать программу с графическим пользовательским интерфейсом, реализующую указанный функционал.

Вариант 10:

Игра «Арканоид». Реализовать игру с одним уровнем. Возможность 2 раза пропустить мяч, после 3-го игра заканчивается. Очки начисляются за разбитые блоки.

Код программы: https://github.com/vangaru/labworks-5-semester/tree/master/OS%26SP/Lab1

Ball.cs:

using System**;**  
using System**.**Collections**;**  
using UnityEngine**;**  
  
public class **Ball** : MonoBehaviour  
**{**  
private SpriteRenderer sr**;**  
  
public bool **isLightningBall;**  
  
public ParticleSystem **lightningBallEffect;**  
  
public float **lightningBallDuration** = 10**;**  
  
public static event Action<Ball> OnBallDeath**;**  
public static event Action<Ball> OnLightningBallEnable**;**  
public static event Action<Ball> OnLightningBallDisable**;**  
  
private void **Awake()**  
 **{**  
this**.**sr = GetComponentInChildren<SpriteRenderer>**();**  
 **}**  
  
public void Die**()**  
 **{**  
OnBallDeath?**.**Invoke**(**this**);**  
Destroy**(**gameObject**,** 1**);**  
 **}**  
  
public void StartLightningBall**()**  
 **{**  
if **(**!this**.**isLightningBall**)**  
 **{**  
this**.**isLightningBall = true**;**  
this**.**sr**.**enabled = false**;**  
lightningBallEffect**.**gameObject**.**SetActive**(**true**);**  
StartCoroutine**(**StopLightningBallAfterTime**(**this**.**lightningBallDuration**));**  
  
OnLightningBallEnable?**.**Invoke**(**this**);**  
 **}**  
 **}**  
  
private IEnumerator StopLightningBallAfterTime**(**float seconds**)**  
 **{**  
yield return new WaitForSeconds**(**seconds**);**  
  
StopLightningBall**();**  
 **}**  
  
private void StopLightningBall**()**  
 **{**  
if **(**this**.**isLightningBall**)**  
 **{**  
this**.**isLightningBall = false**;**  
this**.**sr**.**enabled = true**;**  
lightningBallEffect**.**gameObject**.**SetActive**(**false**);**  
  
OnLightningBallDisable?**.**Invoke**(**this**);**  
 **}**  
 **}**  
**}**

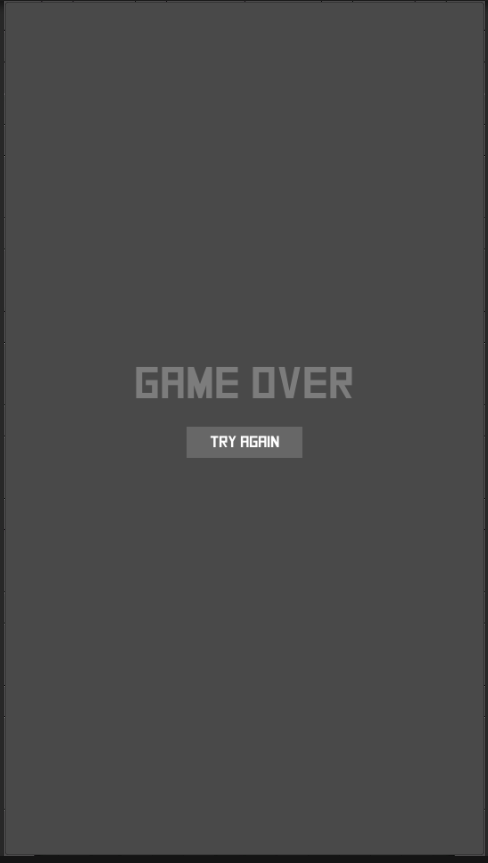
Brick.cs:

using System**;**  
using System**.**Collections**.**Generic**;**  
using AboutWindowApi**;**  
using UnityEngine**;**  
using static UnityEngine**.**ParticleSystem**;**  
  
public class **Brick** : MonoBehaviour  
**{**  
private SpriteRenderer sr**;**  
private BoxCollider2D boxCollider**;**  
  
public int **Hitpoints** = 1**;**  
public ParticleSystem **DestroyEffect;**  
  
public static event Action<Brick> OnBrickDestruction**;**  
  
private void **Awake()**  
 **{**  
this**.**sr = this**.**GetComponent<SpriteRenderer>**();**  
this**.**boxCollider = this**.**GetComponent<BoxCollider2D>**();**  
Ball**.**OnLightningBallEnable += OnLightningBallEnable**;**  
Ball**.**OnLightningBallDisable += OnLightningBallDisable**;**  
 **}**  
  
private void OnLightningBallDisable**(**Ball obj**)**  
 **{**  
if **(**this != null**)**  
 **{**  
this**.**boxCollider**.**isTrigger = false**;**  
 **}**  
 **}**  
  
private void OnLightningBallEnable**(**Ball obj**)**  
 **{**  
if **(**this != null**)**  
 **{**  
this**.**boxCollider**.**isTrigger = true**;**  
 **}**  
 **}**  
  
private void **OnCollisionEnter2D(**Collision2D collision**)**  
 **{**  
bool instantKill = false**;**  
  
if **(**collision**.**collider**.**tag == "Ball"**)**  
 **{**  
Ball ball = collision**.**gameObject**.**GetComponent<Ball>**();**  
instantKill = ball**.**isLightningBall**;**  
 **}**  
  
if **(**collision**.**collider**.**tag == "Ball" || collision**.**collider**.**tag == "Projectile"**)**  
 **{**  
this**.**TakeDamage**(**instantKill**);**  
 **}**  
 **}**  
  
private void **OnTriggerEnter2D(**Collider2D collision**)**  
 **{**  
bool instantKill = false**;**  
  
if **(**collision**.**tag == "Ball"**)**  
 **{**  
Ball ball = collision**.**gameObject**.**GetComponent<Ball>**();**  
instantKill = ball**.**isLightningBall**;**  
 **}**  
  
if **(**collision**.**tag == "Ball" || collision**.**tag == "Projectile"**)**  
 **{**  
this**.**TakeDamage**(**instantKill**);**  
 **}**  
 **}**  
  
private void TakeDamage**(**bool instantKill**)**  
 **{**  
this**.**Hitpoints--**;**  
  
if **(**Helper**.**ReadyToKill**(**this**.**Hitpoints**,** instantKill**))**  
 **{**  
BricksManager**.**Instance**.**RemainingBricks**.**Remove**(**this**);**  
OnBrickDestruction?**.**Invoke**(**this**);**  
OnBrickDestroy**();**  
SpawnDestroyEffect**();**  
Destroy**(**this**.**gameObject**);**  
 **}**  
else  
 **{**  
this**.**sr**.**sprite = BricksManager**.**Instance**.**Sprites**[**this**.**Hitpoints - 1**];**  
 **}**  
 **}**  
  
private void OnBrickDestroy**()**  
 **{**  
float buffSpawnChance = UnityEngine**.**Random**.**Range**(**0**,** 100f**);**  
float deBuffSpawnChance = UnityEngine**.**Random**.**Range**(**0**,** 100f**);**  
bool alreadySpawned = false**;**  
  
if **(**buffSpawnChance <= CollectablesManager**.**Instance**.**BuffChance**)**  
 **{**  
alreadySpawned = true**;**  
Collectable newBuff = this**.**SpawnCollectable**(**true**);**  
 **}**  
  
if **(**deBuffSpawnChance <= CollectablesManager**.**Instance**.**DebuffChance && !alreadySpawned**)**  
 **{**  
Collectable newDebuff = this**.**SpawnCollectable**(**false**);**  
 **}**  
 **}**  
  
private Collectable SpawnCollectable**(**bool isBuff**)**  
 **{**  
List<Collectable> collection**;**  
  
if **(**isBuff**)**  
 **{**  
collection = CollectablesManager**.**Instance**.**AvailableBuffs**;**  
 **}**  
else  
 **{**  
collection = CollectablesManager**.**Instance**.**AvailableDebuffs**;**  
 **}**  
  
int buffIndex = UnityEngine**.**Random**.**Range**(**0**,** collection**.**Count**);**  
Collectable prefab = collection**[**buffIndex**];**  
Collectable newCollectable = Instantiate**(**prefab**,** this**.**transform**.**position**,** Quaternion**.**identity**)** as Collectable**;**  
  
return newCollectable**;**  
 **}**  
  
private void SpawnDestroyEffect**()**  
 **{**  
Vector3 brickPos = gameObject**.**transform**.**position**;**  
Vector3 spawnPosition = new Vector3**(**brickPos**.**x**,** brickPos**.**y**,** brickPos**.**z - 0.2f**);**  
GameObject effect = Instantiate**(**DestroyEffect**.**gameObject**,** spawnPosition**,** Quaternion**.**identity**);**  
  
MainModule mm = effect**.**GetComponent<ParticleSystem>**().**main**;**  
mm**.**startColor = this**.**sr**.**color**;**  
Destroy**(**effect**,** DestroyEffect**.**main**.**startLifetime**.**constant**);**  
 **}**  
  
public void Init**(**Transform containerTransform**,** Sprite sprite**,** Color color**,** int hitpoints**)**  
 **{**  
this**.**transform**.**SetParent**(**containerTransform**);**  
this**.**sr**.**sprite = sprite**;**  
this**.**sr**.**color = color**;**  
this**.**Hitpoints = hitpoints**;**  
 **}**  
  
private void **OnDisable()**  
 **{**  
Ball**.**OnLightningBallEnable -= OnLightningBallEnable**;**  
Ball**.**OnLightningBallDisable -= OnLightningBallDisable**;**  
 **}**  
**}**

Paddle.cs:

using System**;**  
using System**.**Collections**;**  
using UnityEngine**;**  
  
public class **Paddle** : MonoBehaviour  
**{**  
#region Singleton  
  
 private static Paddle \_instance**;**  
  
public static Paddle Instance => \_instance**;**  
  
public bool PaddleIsTransforming **{** get**;** set**; }**  
  
private void **Awake()**  
 **{**  
if **(**\_instance != null**)**  
 **{**  
Destroy**(**gameObject**);**  
 **}**  
else  
 **{**  
\_instance = this**;**  
 **}**  
 **}**  
  
#endregion  
  
 private Camera mainCamera**;**  
private float paddleInitialY**;**  
private float defaultPaddleWidthInPixels = 200**;**  
private float defaultLeftClamp = 135**;**  
private float defaultRightClamp = 410**;**  
private SpriteRenderer sr**;**  
private BoxCollider2D boxCol**;**  
  
*// Shooting*  
public bool PaddleIsShooting **{** get**;** set**; }**  
public GameObject **leftMuzzle;**  
public GameObject **rightMuzzle;**  
public Projectile **bulletPrefab;**  
  
public float **extendShrinkDuration** = 10**;**  
public float **paddleWidth** = 2**;**  
public float **paddleHeight** = 0.28f**;**  
  
private void **Start()**  
 **{**  
mainCamera = FindObjectOfType<Camera>**();**  
paddleInitialY = this**.**transform**.**position**.**y**;**  
sr = GetComponent<SpriteRenderer>**();**  
boxCol = GetComponent<BoxCollider2D>**();**  
  
 **}**  
  
private void **Update()**  
 **{**  
PaddleMovement**();**  
UpdateMuzzlePosition**();**  
 **}**  
  
private void UpdateMuzzlePosition**()**  
 **{**  
leftMuzzle**.**transform**.**position = new Vector3**(**this**.**transform**.**position**.**x - **(**this**.**sr**.**size**.**x / 2**)** + 0.1f**,** this**.**transform**.**position**.**y + 0.2f**,** this**.**transform**.**position**.**z**);**  
rightMuzzle**.**transform**.**position = new Vector3**(**this**.**transform**.**position**.**x + **(**this**.**sr**.**size**.**x / 2**)** - 0.153f**,** this**.**transform**.**position**.**y + 0.2f**,** this**.**transform**.**position**.**z**);**  
 **}**  
  
public void StartWidthAnimation**(**float newWidth**)**  
 **{**  
StartCoroutine**(**AnimatePaddleWidth**(**newWidth**));**  
 **}**  
  
public IEnumerator AnimatePaddleWidth**(**float width**)**  
 **{**  
this**.**PaddleIsTransforming = true**;**  
this**.**StartCoroutine**(**ResetPaddleWidthAfterTime**(**this**.**extendShrinkDuration**));**  
  
if **(**width > this**.**sr**.**size**.**x**)**  
 **{**  
float currentWidth = this**.**sr**.**size**.**x**;**  
while **(**currentWidth < width**)**  
 **{**  
currentWidth += Time**.**deltaTime \* 2**;**  
this**.**sr**.**size = new Vector2**(**currentWidth**,** paddleHeight**);**  
boxCol**.**size = new Vector2**(**currentWidth**,** paddleHeight**);**  
yield return null**;**  
 **}**  
 **}**  
else  
 **{**  
float currentWidth = this**.**sr**.**size**.**x**;**  
while **(**currentWidth > width**)**  
 **{**  
currentWidth -= Time**.**deltaTime \* 2**;**  
this**.**sr**.**size = new Vector2**(**currentWidth**,** paddleHeight**);**  
boxCol**.**size = new Vector2**(**currentWidth**,** paddleHeight**);**  
yield return null**;**  
 **}**  
 **}**  
  
this**.**PaddleIsTransforming = false**;**  
 **}**  
  
private IEnumerator ResetPaddleWidthAfterTime**(**float seconds**)**  
 **{**  
yield return new WaitForSeconds**(**seconds**);**  
this**.**StartWidthAnimation**(**this**.**paddleWidth**);**  
 **}**  
  
private void PaddleMovement**()**  
 **{**  
float paddleShift = **(**defaultPaddleWidthInPixels - **((**defaultPaddleWidthInPixels / 2**)** \* this**.**sr**.**size**.**x**))** / 2**;**  
float leftClamp = defaultLeftClamp - paddleShift**;**  
float rightClamp = defaultRightClamp + paddleShift**;**  
float mousePositionPixels = Mathf**.**Clamp**(**Input**.**mousePosition**.**x**,** leftClamp**,** rightClamp**);**  
float mousePositionWorldX = mainCamera**.**ScreenToWorldPoint**(**new Vector3**(**mousePositionPixels**,** 0**,** 0**)).**x**;**  
this**.**transform**.**position = new Vector3**(**mousePositionWorldX**,** paddleInitialY**,** 0**);**  
 **}**  
  
private void **OnCollisionEnter2D(**Collision2D coll**)**  
 **{**  
if **(**coll**.**gameObject**.**tag == "Ball"**)**  
 **{**  
Rigidbody2D ballRb = coll**.**gameObject**.**GetComponent<Rigidbody2D>**();**  
Vector3 hitPoint = coll**.**contacts**[**0**].**point**;**  
Vector3 paddleCenter = new Vector3**(**this**.**gameObject**.**transform**.**position**.**x**,** this**.**gameObject**.**transform**.**position**.**y**);**  
  
ballRb**.**velocity = Vector2**.**zero**;**  
  
float difference = paddleCenter**.**x - hitPoint**.**x**;**  
  
if **(**hitPoint**.**x < paddleCenter**.**x**)**  
 **{**  
ballRb**.**AddForce**(**new Vector2**(**-**(**Mathf**.**Abs**(**difference \* 200**)),** BallsManager**.**Instance**.**initialBallSpeed**));**  
 **}**  
else  
 **{**  
ballRb**.**AddForce**(**new Vector2**((**Mathf**.**Abs**(**difference \* 200**)),** BallsManager**.**Instance**.**initialBallSpeed**));**  
 **}**  
 **}**  
 **}**  
  
public void StartShooting**()**  
 **{**  
if **(**!this**.**PaddleIsShooting**)**  
 **{**  
this**.**PaddleIsShooting = true**;**  
StartCoroutine**(**StartShootingRoutine**());**  
 **}**  
 **}**  
  
public IEnumerator StartShootingRoutine**()**  
 **{**  
float fireCooldown = .5f**;** *// TODO: extract this into unity variable*  
float fireCooldownLeft = 0**;**  
  
float shootingDuration = 10**;** *// TODO: extract this into unity variable*  
float shootingDurationLeft = shootingDuration**;**  
  
*//Debug.Log("START SHOOTING");*  
  
while **(**shootingDurationLeft >= 0**)**  
 **{**  
fireCooldownLeft -= Time**.**deltaTime**;**  
shootingDurationLeft -= Time**.**deltaTime**;**  
  
if **(**fireCooldownLeft <= 0**)**  
 **{**  
this**.**Shoot**();**  
fireCooldownLeft = fireCooldown**;**  
*//Debug.Log($"Shoot at {Time.time}");*  
**}**  
  
yield return null**;**  
 **}**  
  
*//Debug.Log("STOP SHOOTING!");*  
this**.**PaddleIsShooting = false**;**  
leftMuzzle**.**SetActive**(**false**);**  
rightMuzzle**.**SetActive**(**false**);**  
 **}**  
  
private void Shoot**()**  
 **{**  
leftMuzzle**.**SetActive**(**false**);**  
rightMuzzle**.**SetActive**(**false**);**  
  
leftMuzzle**.**SetActive**(**true**);**  
rightMuzzle**.**SetActive**(**true**);**  
  
this**.**SpawnBullet**(**leftMuzzle**);**  
this**.**SpawnBullet**(**rightMuzzle**);**  
 **}**  
  
private void SpawnBullet**(**GameObject muzzle**)**  
 **{**  
Vector3 spawnPosition = new Vector3**(**muzzle**.**transform**.**position**.**x**,** muzzle**.**transform**.**position**.**y + 0.2f**,** muzzle**.**transform**.**position**.**z**);**  
Projectile bullet = Instantiate**(**bulletPrefab**,** spawnPosition**,** Quaternion**.**identity**);**  
Rigidbody2D bulletRb = bullet**.**GetComponent<Rigidbody2D>**();**  
bulletRb**.**AddForce**(**new Vector2**(**0**,** 450f**));**  
 **}**  
**}**

Результат выполнения:



Вывод: познакомился с основами разработки и проектирования игр с графическим пользовательским интерфейсом.