

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «Прикладное программирование»**

**Тема: «разработка механик для игры “2D runner”»**

Выполнил студент

Завадский А.М

Группа П2-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

**Королёв 2021 г.**

**Оглавление**

[Введение 3](#_Toc69815292)

[1 Теоретическая часть 4](#_Toc69815293)

[1.1 Описание предметной области 4](#_Toc69815294)

[1.2 Описание существующих разработок 6](#_Toc69815295)

[1.3 Описание существующих разработок 7](#_Toc69815295)

[1.4 Среда разработки 9](#_Toc69815294)

[2. ПРОЕКТНАЯ ЧАСТЬ  
2.1. Диаграмма прецедентов 10  
2.2. Диаграмма классов 11](#_Toc43894630)

[2.3. Выбор языка программирования 12](#_Toc43894631)

[3. ЭКСПЛУТАЦИОННАЯ ЧАСТЬ 13](#_Toc43894636)

[3.1. Условия выполнения программы 13](#_Toc43894637)

[3.2. Выполнение программы 14](#_Toc43894638)

[3.3. Возможные улучшения 16](#_Toc43894639)

[ЗАКЛЮЧЕНИЕ 17](#_Toc43894643)

[Список использованных источников. 18](#_Toc43894644)  
Приложение 19

# 

# Введение

Данный курсовой проект заключается в разработке 2d игры, включающую в себя управление персонажем, случайное появление объектов, и взаимодействие между ними.

# Теоретическая часть

## Описание предметной области

**Что такое компьютерная игра**

**Компьютерная игра** (или видеоигра) — компьютерная программа, служащая для организации игрового процесса, связи с партнёрами по игре, или сама выступающая в качестве партнёра.

В настоящее время в некоторых случаях вместо компьютерной игры может использоваться термин видеоигра, то есть данные термины могут использоваться как синонимы и быть взаимозаменяемыми. В компьютерных играх, как правило, игровая ситуация воспроизводится на экране дисплея или обычного телевизора.

**Когда и кто изобрел компьютерные игры.**

Изобретение компьютерных игр обычно приписывают одному из трех человек: Ральф Баер, инженер, который выдвинул идею интерактивного телевидения в 1951 году, А.С. Дуглас, который написал «OXO» в 1952 году, компьютерную реализацию Tic-Tac - Тое, или Уильям Хигинботам, создавший в 1958 году игру «Теннис для двоих».

Данный курсовой проект посвящен созданию 2D игры жанра “Раннер”. Разработка программы будет происходить на языке программирования Python. Выбором данной темы послужили особенности данного жанра:

* Игра должна быть простой, но усложняться по мере зарабатывания игроком очков.
* Игровые объекты должны генерироваться случайным образом;
* Игра должна содержать систему жизней но при их потери игрок должен начать сначало

Runner и Endless Runner — жанры игр, в центре внимания которых герой-бегун. Практически всегда персонаж бежит автоматически и от игрока требуется лишь корректировать его траекторию движения, вовремя совершать прыжки и уклоняться от препятствий. Это требует в особенности концентрации внимания и хорошей реакции.  
  
Endless Runner — этот обычный раннер с одной особенностью — персонажу не предстоит достичь финиша просто потому, что его нет. Бег, сбор бонусов, очков, преодоление препятствий — вот из чего состоят игры такого жанра.

## Классификация игр по целевой платформе

Основным видом разделения видеоигр на категории является разделение по платформам. Оно указывает, на каком конкретном устройстве можно запустить игру. Если у игрока нет соответствующей платформы, то он не сможет запустить в игру, рассчитанную именно для этой платформы. Такая классификация применима ко всему программному обеспечению.

* **Одноплатформенная игра**

Игра, которая разработана для одной конкретной платформы или устройства. Большинство из них это игры, которые разработаны для игровых приставок (консоли).

* **Мультиплатформенная игра**

Игра, которая разработана сразу для нескольких платформ. Игры данного типа являются наиболее распространёнными.

## Изучение существующих программ

На данный момент в интернете находится куча подобных приложений. Игры жанра «раннер» являются весьма популярными в наши дни и имеют ряд плюсов. Из них можно выделить такие как:

* **Сложность игры.** раннеры бывают очень разными от совсем «детских» сложностей до безумно сложных. Это может привлечь самую разную аудиторию.
* **Доступность.** Множество игр можно как найти в открытом доступе в интернете без скачивания.
* **Простота использования.** Тут мало что можно добавить – простое и понятное всем управление не могло не подкупить пользователей.
* **Заработок.** Если ориентироваться на создание мобильных приложений, то они легко могут принести доход на рекламе или на покупке пользователем внутри игровой валюты.
* **Создание.** Игры этого жанра очевидно более простые в создании, чем других жанров. Поддерживать игру обновлениями не всегда будет нужно.

Вот некоторые подобные игры, которые можно предоставить в качестве примера:

**1.Subway Surfers** — мобильная игра в жанре Раннер для [iOS](https://ru.wikipedia.org/wiki/IOS" \o "Английский язык) и [Android](https://ru.wikipedia.org/wiki/Android" \o ").

По сюжету персонаж игрока изрисовал [граффити](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%84%D0%B8%D1%82%D0%B8) один из вагонов метро. Бомбера замечает инспектор с собакой, после чего начинается погоня по железнодорожным путям. Основная цель игры состоит в том, чтобы игрок, ускоряясь, пробежал как можно дальше, уворачиваясь от встречных поездов и преодолевая препятствия. Если игрок с чем-то сталкивается, то инспектор его догоняет (причём вне зависимости от того, насколько далеко он успел убежать) и игра заканчивается.

рис.1

**2.Temple Run2** — платформенная игра на платформы [iOS](https://ru.wikipedia.org/wiki/IOS" \o "IOS), [Android](https://ru.wikipedia.org/wiki/Android" \o "Android), а также [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone), симулятор бесконечного бега. Игра начинается каждый раз в новой самогенерирующейся локации, с архитектурой, внешне напоминающей храмы южноамериканских культур. Постепенно по мере прохождения какого-нибудь уровня растёт скорость бега.

рис.2

## Среда разработки

Для разработки игры мы будем использовать unity2D с языком программирования c#

**Unity** — [межплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), разработанная американской компанией [Unity Technologies](https://ru.wikipedia.org/wiki/Unity_Technologies" \o "Unity Technologies). Unity позволяет создавать приложения, работающие на более чем 25 различных [платформах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), включающих [персональные компьютеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), [игровые консоли](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C), [мобильные устройства](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B0), [интернет-приложения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и другие.

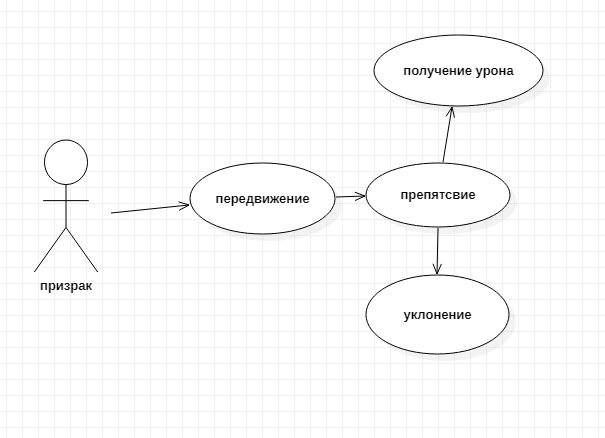
**C#** (произносится си шарп) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft

**2. ПРОЕКТНАЯ ЧАСТЬ**

**2.1. Диаграмма прецедентов**

В игре происходит взаимодействие пользователя с управляемым призраком. Тот в свою очередь может передвигаться.

1. Совершается передвижение   
 2. Если игрок врезается в препятствие он теряет 1-ну жизнь.  
 3. Игроку начисляется очки за каждое препятствие от которого он уклоняется .



**2.2. Диаграмма классов**

В диаграмме представлены восемь классов player (игрок), и Obstacle (ловушка).

У класса target структура полостью совпадает со структурой класса player, за исключением условий движения объекта.

|  |  |
| --- | --- |
| Класс player | Класс Obstacle |
| Включает в себя:   * Координаты переменной player * Изображение для переменной * Движение объекта | Включает в себя:   * Координаты переменной target * Изображение для переменной * Условия движения объекта |
| Класс Score | Класс ObstacleSpawn |
| Включает в себя:   * Получение очков за каждое пройдённое препятствия | Включает в себя:   * Варианты расположения ловушек |
| Класс Destroyer | Класс Spawner |
| Включает в себя:   * Уничтожает ловушки от которых игрок уже уклонился(вне видимости игрока) | Включает в себя:   * Установка ловушек(вне видимости игрока) |
| Класс Restart | Класс Background |
| Включает в себя:   * Отвечает за рестарт игры | Включает в себя:   * Замена заднего фона со временем |

**2.3. Выбор языка программирования**

**C#** объединяет лучшие идеи современных **языков программирования** Java, C++, Visual Basic и т. д. Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой . Net, **C#** позволяет быстрее, чем любой другой язык, разрабатывать программные решения.

**3. ЭКСПЛУАТАЦИОННАЯ ЧАСТЬ.**

**3.1. Условия выполнения программы**

Для запуска проекта потребуется Unity 2020.2.3f1

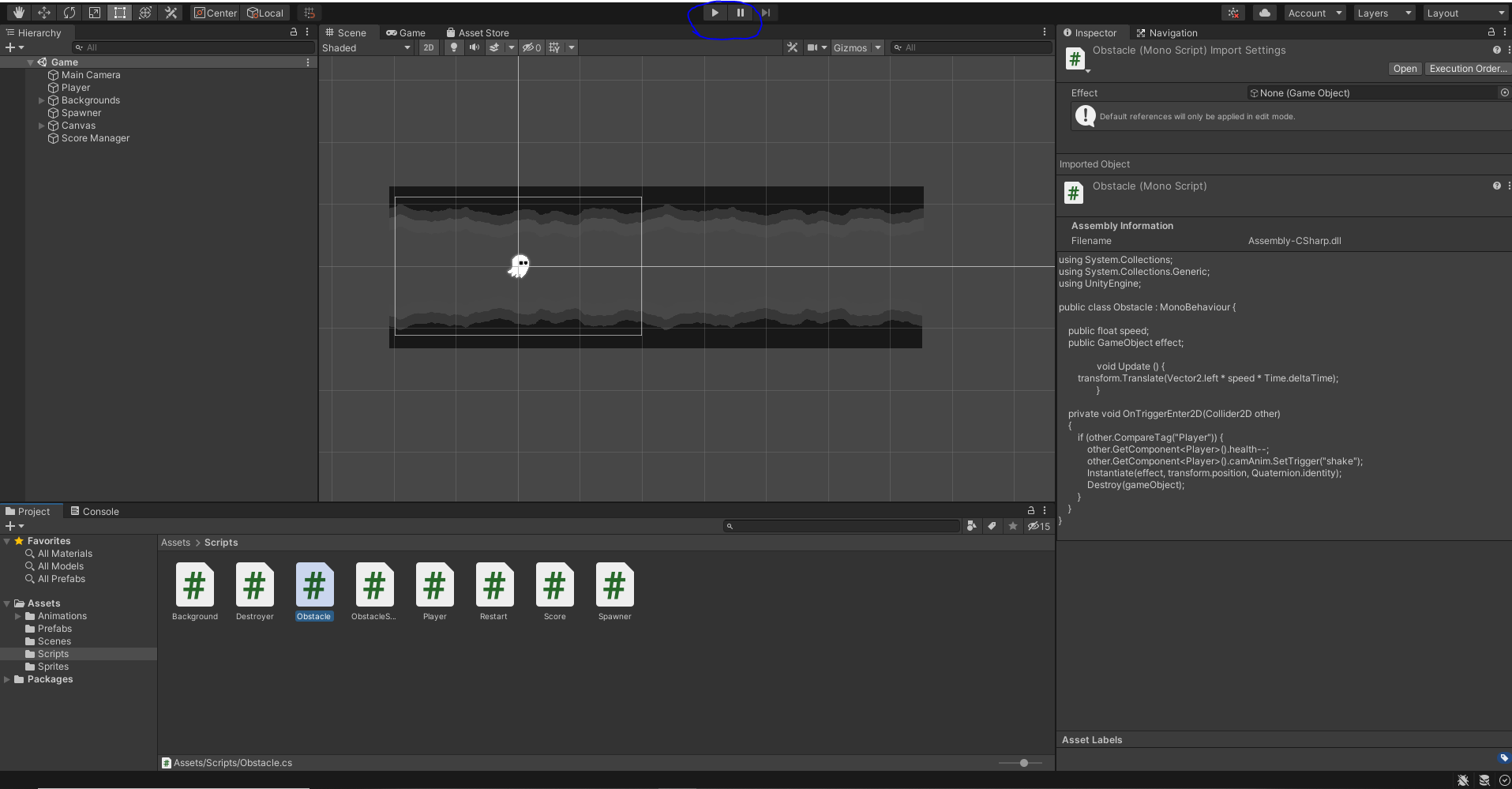
Минимальные системные требования Unity для windows

|  |  |
| --- | --- |
| **Operating system version** | Windows 7 (SP1+) and Windows 10 |
| **CPU** | x86, x64 architecture with SSE2 instruction set support. |
| **Graphics API** | DX10, DX11, DX12 capable. |
| **Additional requirements** | Hardware vendor officially supported drivers. For development: IL2CPP scripting backend requires Visual Studio 2015 with C++ Tools component or later and Windows 10 SDK. |

**3.2. Выполнение программы**

Для запуска нам нужно выбрать наш проект в Unity

После запуска проекта для начала игры нужно запустить игру для этого нажмите кнопку play:рис.3



После чего запустится игра:



Рис.4

Для выхода из игры нужно нажать на play.

**3.3. Возможные улучшения**

1. Добавление разных бонусов.

2. Добавление новых ловушек.

3. Создание системы уровней.

4. Создание разных режимов например: режим с 1-ой жизнью.

**Список использованных источников**

1. https://docs.microsoft.com/ru-ru/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp

2. https://metanit.com/sharp/tutorial/

**ЗАКЛЮЧЕНИЕ**

По окончанию написания курсового проекта было создана компьютерная игра жанра раннер на языке c#. Для разработки я ознакомился с движком unity.

**ЛИСТИНГИ**

Spawner

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Spawner : MonoBehaviour {

private float timeBtwSpawns;

public float startTimeBtwSpawns;

public float timeDecrease;

public float minTime;

public GameObject[] obstacleTemplate;

private void Start()

{

timeBtwSpawns = startTimeBtwSpawns;

}

private void Update()

{

if (timeBtwSpawns <= 0)

{

int rand = Random.Range(0, obstacleTemplate.Length);

Instantiate(obstacleTemplate[rand], transform.position, Quaternion.identity);

timeBtwSpawns = startTimeBtwSpawns;

if (startTimeBtwSpawns > minTime) {

startTimeBtwSpawns -= timeDecrease;

}

}

else {

timeBtwSpawns -= Time.deltaTime;

}

}

}

Score

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Score : MonoBehaviour {

public int score;

public Text scoreDisplay;

private void Update()

{

scoreDisplay.text = score.ToString();

}

private void OnTriggerEnter2D(Collider2D other)

{

score++;

Destroy(other.gameObject);

}

}

Restart

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Restart : MonoBehaviour {

void Update () {

if (Input.GetKeyDown(KeyCode.R)){

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

}

}

}

ObstacleSpawn

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ObstacleSpawn : MonoBehaviour {

public GameObject obstacle;

private void Start()

{

Instantiate(obstacle, transform.position, Quaternion.identity);

}

}

Destroyer

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Destroyer : MonoBehaviour {

public float lifetime;

private void Start()

{

Destroy(gameObject, lifetime);

}

}

Background

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Background : MonoBehaviour {

public float speed;

public float Xend;

public float Xstart;

private void Update()

{

transform.Translate(Vector2.left \* speed \* Time.deltaTime);

if (transform.position.x < Xend) {

Vector2 pos = new Vector2(Xstart, transform.position.y);

transform.position = pos;

}

}

}

Player

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Player : MonoBehaviour {

public float speed;

public float increment;

public float maxY;

public float minY;

private Vector2 targetPos;

public int health;

public GameObject moveEffect;

public Animator camAnim;

public Text healthDisplay;

public GameObject spawner;

public GameObject restartDisplay;

private void Update()

{

if (health <= 0) {

spawner.SetActive(false);

restartDisplay.SetActive(true);

Destroy(gameObject);

}

healthDisplay.text = health.ToString();

transform.position = Vector2.MoveTowards(transform.position, targetPos, speed \* Time.deltaTime);

if (Input.GetKeyDown(KeyCode.UpArrow) && transform.position.y < maxY) {

camAnim.SetTrigger("shake");

Instantiate(moveEffect, transform.position, Quaternion.identity);

targetPos = new Vector2(transform.position.x, transform.position.y + increment);

} else if (Input.GetKeyDown(KeyCode.DownArrow) && transform.position.y > minY) {

camAnim.SetTrigger("shake");

Instantiate(moveEffect, transform.position, Quaternion.identity);

targetPos = new Vector2(transform.position.x, transform.position.y - increment);

}

}

}

Obstacle

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Obstacle : MonoBehaviour {

public float speed;

public GameObject effect;

void Update () {

transform.Translate(Vector2.left \* speed \* Time.deltaTime);

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player")) {

other.GetComponent<Player>().health--;

other.GetComponent<Player>().camAnim.SetTrigger("shake");

Instantiate(effect, transform.position, Quaternion.identity);

Destroy(gameObject);

}

}

}