Муниципальное бюджетное общеобразовательное учреждение
«Средняя общеобразовательная школа № 9 им. В. И. Некрасова»
г. Сосновый Бор Ленинградской области

ТЕМА

Разработка Android приложения

Индивидуальный проект
обучающегося 10а класса
Зайцева Кирилла

Руководитель проекта:
Нефедьев Вадим Викторович,
учитель информатики

2020 – 2021 учебный год

Тема проекта: разработка Android приложения.

Актуальность: в наше время рынок Android приложений безостановочно растёт. Вместе с количеством приложений растёт и количество компаний и разработчиков-одиночек, а соответственно увеличивается конкуренция. Очень сложно пробиться в эту сферу, не имея опыта или большого количества ресурсов. По этой причине в магазинах приложений появляется огромное количество ширпотреба - сделанные за пару дней продукты, с целью собрать как можно больше денег на рекламе.

Проблема: очень мало по-настоящему качественных приложений на мобильном рынке.

Цель: разработать приложение для OS Android с нуля и показать, как это делается.

Задачи:

1. Выбрать среду разработки.

2. Рассказать про игровую индустрию

3. Рассказать про операционную систему Android

4. Создать игру

Объект: рынок Android приложений.

Предмет: процесс разработки Andorid приложения.

Продукт проекта: готовая игра/приложение для OS Android.

1.Выбор среды разработки

1.1 Unity 3D

Unity — межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями.

Плюсы и минусы: Как правило, игровой движок предоставляет множество функциональных возможностей, позволяющих их задействовать в различных играх, в которые входят моделирование физических сред, карты нормалей, динамические тени и многое другое. В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие визуальной среды разработки и межплатформенная поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и интегрированную среду, цепочку сборки, что направлено на повышение производительности разработчиков, в частности, этапов создания

прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места, но и наличие инструментария разработки.

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов наследования, объекты в Unity создаются посредством объединения функциональных блоков, а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр. В качестве недостатков приводятся ограничение визуального редактора при работе с многокомпонентными схемами, когда в сложных сценах визуальная работа затрудняется. Вторым недостатком называется отсутствие поддержки Unity ссылок на внешние библиотеки, работу с которыми программистам приходится настраивать самостоятельно, и это также затрудняет командную работу.

рис.1

1.2 Unreal Engine

Unreal Engine — игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх. В прошлом движок распространялся на условиях оплаты ежемесячной подписки; с 2015 года Unreal Engine бесплатен, но разработчики использующих его приложений обязаны перечислять 5% роялти от общемирового дохода с некоторыми

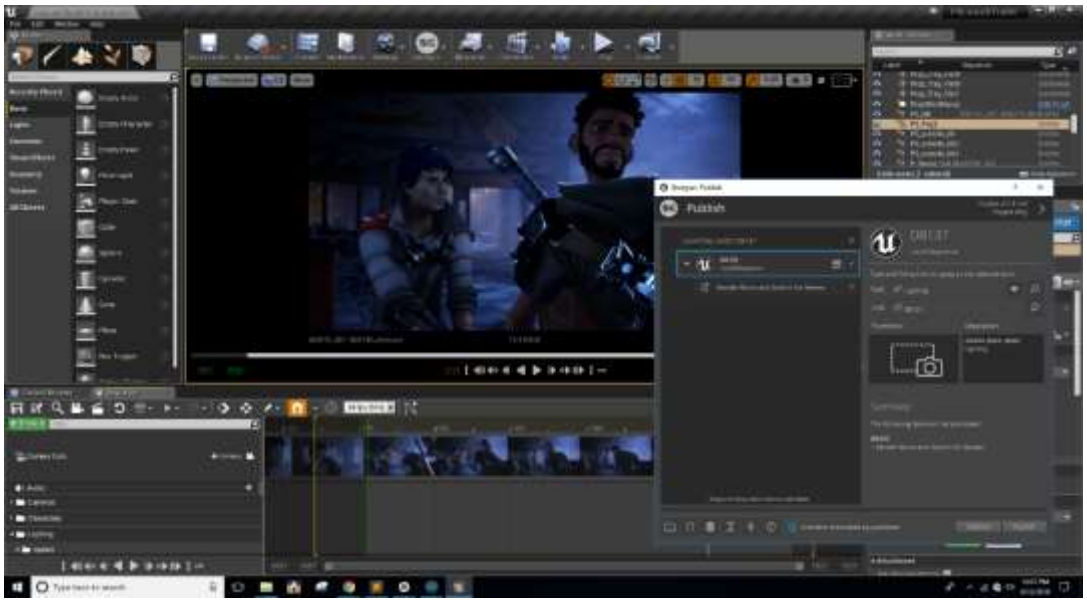

Рис.2

1.3 Game Maker: Studio

GameMaker: Studio — один из самых популярных игровых движков, позволяющий разрабатывать приложения под множество платформ. GameMaker: Studio является серьёзным развитием его предшественника GameMaker и главным отличием является добавление кроссплатформенности, благодаря которой, а также другим существенным

доработкам, GameMaker: Studio стал мощным инструментом для профессиональной разработки.

Бесплатная версия ограничена компиляцией под Windows. По сравнению с ней, Professional версия имеет множество преимуществ, включая управление ресурсами, компиляцию для macOS, Ubuntu и запуск на Android. Также, в профессиональной версии можно покупать отдельные модули, расширяющие функциональность программы. Версия Master Collection содержит все текущие модули и будущие дополнения версии 1.x. GameMaker: Studio содержит множество математических функций для работы со скалярными и векторными величинами, включая тригонометрические вычисления, нахождение степеней, логарифмов, интерполяций, нормалей векторов, скалярных произведений. Имеется встроенный физический движок Box2D, набор функций для работы с ним, демонстрационные проекты Angry Cats и Angry Cats Space.

Рис.3

2.Игровая Индустрия

2.1. Обобщение

Индустрия компьютерных игр зародилась в середине 1970-х годов как движение энтузиастов и за несколько десятилетий выросла из небольшого рынка в одно из крупнейших направлений с годовой прибылью в 9,5 миллиардов долларов в США в 2007 году и 11,7 миллиардов в 2008 году на рынке работают как крупные игроки, так и небольшие фирмы и стартапы, а также независимые разработчики и сообщества

Современные персональные компьютеры множеством новшеств обязаны игровой индустрии. К числу самых значимых относят звуковые и графические карты, CD- и DVD-приводы, и центральные процессоры.

Звуковые карты изначально были разработаны для интегрирования качественного цифрового звука в компьютерные игры, и только потом звуковое оборудование было усовершенствовано под нужды меломанов.

Графические карты, которые на заре компьютерной эпохи эволюционировали в направлении увеличения количества поддерживаемых цветов, позже стали развиваться для аппаратной поддержки графических интерфейсов пользователя и игр. Для GUI требовалось увеличение разрешения экрана, а для игр — ускорение трёхмерной графики.

Изначально CD и DVD были разработаны как недорогой и достаточно надежный способ хранения и распространения любых данных. Впоследствии, когда эти технологии стали применяться в компьютерных играх, началось их развитие в сторону увеличения скорости чтения данных.

Современные игры — одни из самых требовательных приложений на ПК. Многие мощные компьютеры покупаются геймерами, которые требуются для запуска новейших игр, в которых используются самые передовые технологии. Таким образом, игровая индустрия тесно связана с индустрией производства центральных процессоров и других компонентов ПК, так как игры зачастую требуют более высоких аппаратных мощностей, чем бизнес-приложения.

3. Операционная система Android и Google play

3.1 Определение

Android — операционная система для смартфонов, планшетов, электронных книг, цифровых проигрывателей, наручных часов, фитнес-браслетов, игровых приставок, ноутбуков, нетбуков, смартбуков, телевизоров, проекторов и других устройств

Изначально разрабатывалась компанией Android, Inc., которую затем приобрела Google. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google.

Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать библиотеки и компоненты приложений, написанные на Си и других языках.

В 86 % смартфонов, проданных во всём мире во втором квартале 2014 года, была установлена операционная система Android. На конференции разработчиков в мае 2017 года Google объявила, что за всю историю Android было активировано более 2 млрд Android-устройств.

3.2 Google play

Google Play Store — магазин приложений, а также игр, книг, музыки и фильмов от компании, позволяющий сторонним компаниям предлагать владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения.

Учётная запись разработчика, которая даёт возможность публиковать приложения, стоит $25. Платные приложения могут публиковать разработчики не из всех стран.

4. Процесс Разработки игры на Android

4.1. Формирование идеи

Мной было принято решение создавать игру на движке Unity с использованием языка C#. Проект я начал выполнять в 2D. Это упростит процесс. На этом этапе я уже понимал, что сложную игру мне не сделать, всё-таки это мой первый опыт в разработке подобных приложений. Идея была совершенно проста - персонаж должен уклоняться от летящих в него “бомбочек”. Здесь всё просто, из спрайтов (спрайт-картинка, использующаяся в игре) мне понадобится персонаж, фон и бомбочки

4.2 Создание спрайтов

Мне понравилась идея с космическим кораблём, избегающим столкновений с огненными метеоритами. (рис.4 и рис.5)

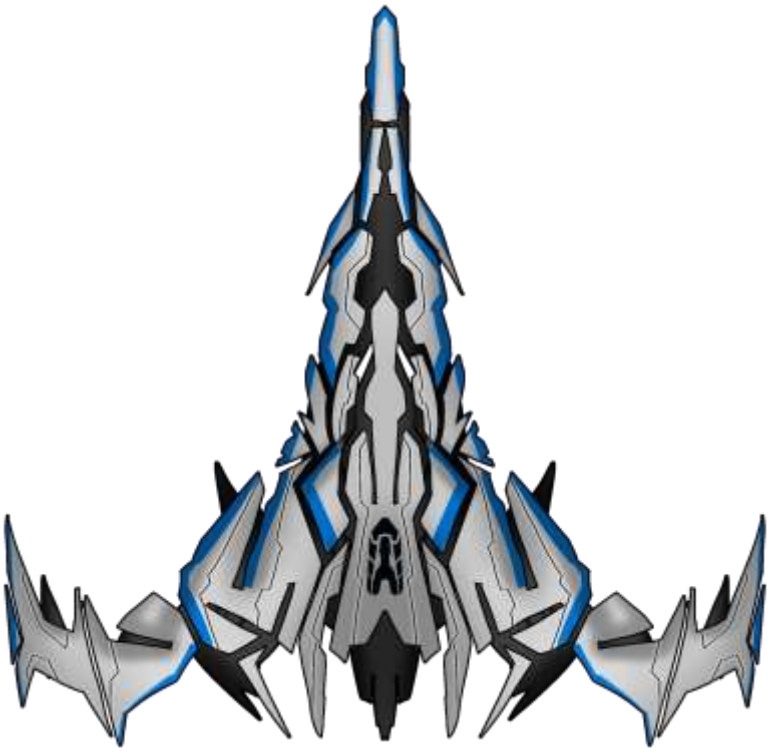

рис.4

рис.5

4.3. Знакомство с Unity

В Unity нам предстоит выбрать настройки под разработку android игры, так же выберем альбомную ориентацию экрана с разрешением 480x800 точек, мы же делаем игру под смартфон.

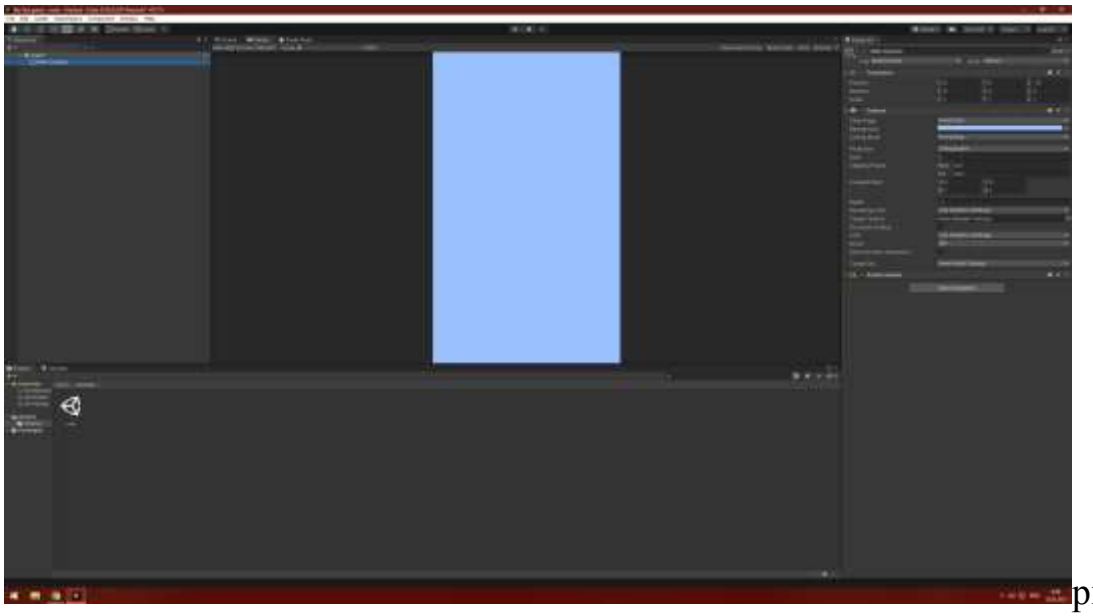
рис.6

Дальше перемещаем фон на наше рабочее пространство, на этом фоне и будет работать игра

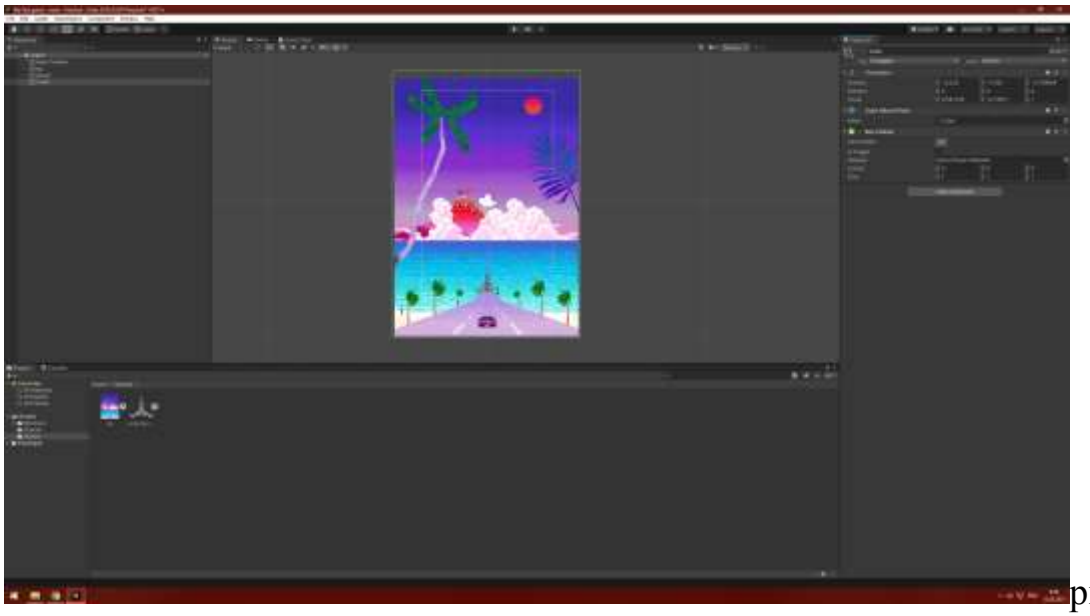
рис.7

4.4 Движение персонажа

Данный код является самым первым скриптом в моей игре, который позволит передвигать персонажа по экрану с использованием мыши или сенсора

```
using UnityEngine;


public class moveplayer : MonoBehaviour
{
  public Transform player;


  void OnMouseDrag () {


        Vector3 mousePos = Camera.main.ScreenToWorldPoint
(Input.mousePosition);
        player.position = new Vector2 (mousePos.x, player.position.y);
  }
}
```

Весь экран мы превращаем в один большой Box Collider – это такое пространство, в котором наши движения будут отслеживаться. Конечно, изначально это просто объект, но после этого кода, он превращается в так называемый сенсор.

Сразу же сталкиваюсь с проблемой и первым багом в своей игре- персонаж вылетает за пределы экрана. Надо исправлять.

```
using UnityEngine;
public class MovePlayer : MonoBehaviour
{
    public Transform player;
    [SerializeField]
    private float speed = 10f;
    void OnMouseDrag()
    {

        Vector3 mousePos =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
        mousePos.x = mousePos.x > 2.3f ? 2.3f : mousePos.x;
        mousePos.x = mousePos.x < -2.3f ? -2.3f : mousePos.x;
        player.position = Vector2.MoveTowards(player.position,
            new Vector2(mousePos.x, player.position.y),
            speed * Time.deltaTime):
    }
}
```

Я добавляю проверку позиции нашего персонажа на экране, как видите я задал это расстояние от -2.3 до 2.3 по оси абсцисс. Теперь, когда персонаж достигает этой границы он останавливается и не движется дальше.

4.5 Враги

Враги будут представлять собой падающие огни, для начала возьмём спрайт огня и занесём в его в категорию prefabs. Теперь создадим скрипт, который будет генерировать их за пределами экрана.

```
using System.Collections;
using UnityEngine;
public class SpawnBombs : MonoBehaviour
{
   public GameObject bomb;

   void Start()
   {
      StartCoroutine(Spawn());
   }

   IEnumerator Spawn()
   {
      while (!Player.lose)
      {
         Instantiate(bomb, new Vector2(Random.Range(-2.3f, 2.3f), 5.6f), Quaternion.identity);
         yield return new WaitForSeconds(0.6f);
      }
   }

}
```

Как видите я задаю определенную высоту, а также диапазон в котором у нас будут появляться огоньки. Обязательно указываю время,

через которое будут генерироваться объекты, регулируя этот параметр можно менять количество огней. Обратите внимание, что каждый раз мы вводим различные библиотеки Unity. Без этого скрипт бы не работал.

Мы видим, что огоньки появляются за экраном, но не падают. Нужно написать скрипт падения огней.

```
sing UnityEngine;
public class FallDown : MonoBehaviour
{
   [SerializeField]
   private float fallSpeed = 3f;
    void Update()
   {
      if (transform.position.y < -6f)
         Destroy (gameObject);


      transform.position -= new Vector3 (0, fallSpeed * Time.deltaTime, 0);
   }
}
```

Теперь наши враги падают вниз, я могу регулировать скорость падения меняя fallspeed

Теперь персонаж может двигаться влево-вправо, а сверху падают огни, но они не

наносят никакого вреда нашему герою, давайте сделаем, чтобы при касании с ними мы проигрывали. Для этого присвоим огню и космолёту так называемый “хит бокс” Это обозначит границы персонажа и врага

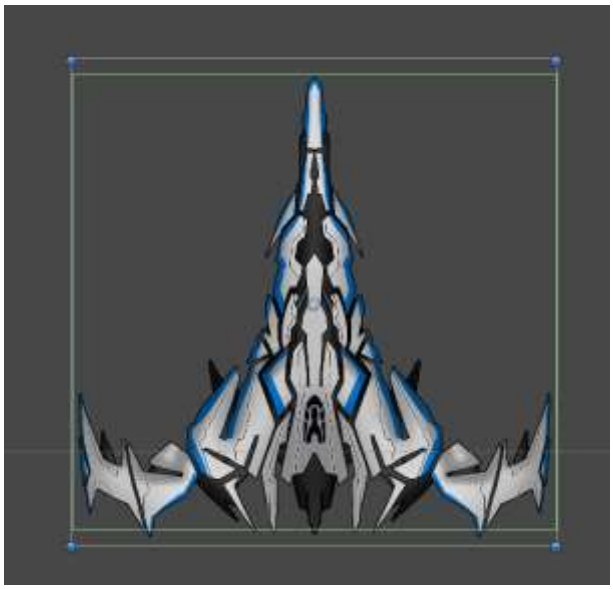

рис.8

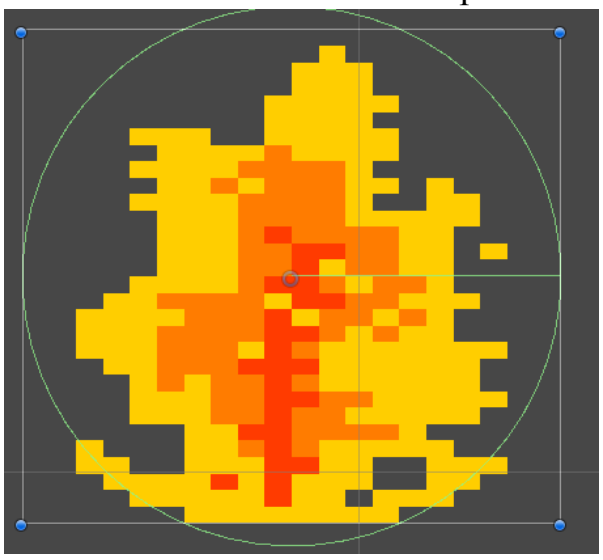

рис.10

рис.9

У огонька я сделал круглый хитбокс, а у персонажа квадратный. Так будет оптимальнее. Теперь осталось только добавить скрипт, который завершает игру при встрече этих двух объектов.

```
using UnityEngine;
public class Player : MonoBehaviour {
    public GameObject restart;
    public static bool lose = false;

    void Awake ()
    {
        lose = false;
    }
  void OnTriggerEnter2D(Collider2D other) {
        if (other.gameObject.tag == "Bomb") {
            lose = true;
            restart.SetActive(true);
        }
    }
}
```

Чтобы этот скрипт работал надо добавить тэг к огоньку. Я назвал его "Bomb". Теперь если игрок касается объекта с таким тэгом, то игра завершается.

4.6 Перезапуск игры

После встречи с врагом игра завершается, но нужно сделать кнопку, которая будет перезапускать её. Я нарисовал такой вариант кнопки

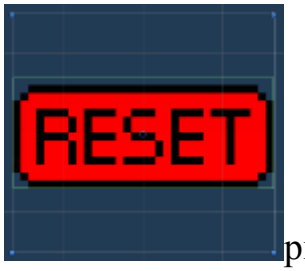

рис.11

Также задал ей хитбокс, но уже для нажатия нами. Создаём простенький скрипт для кнопки

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class Restart : MonoBehaviour
{

    void OnMouseDown()
    {
        SceneManager.LoadScene("main");
    }
}
```

4.7 Build игры

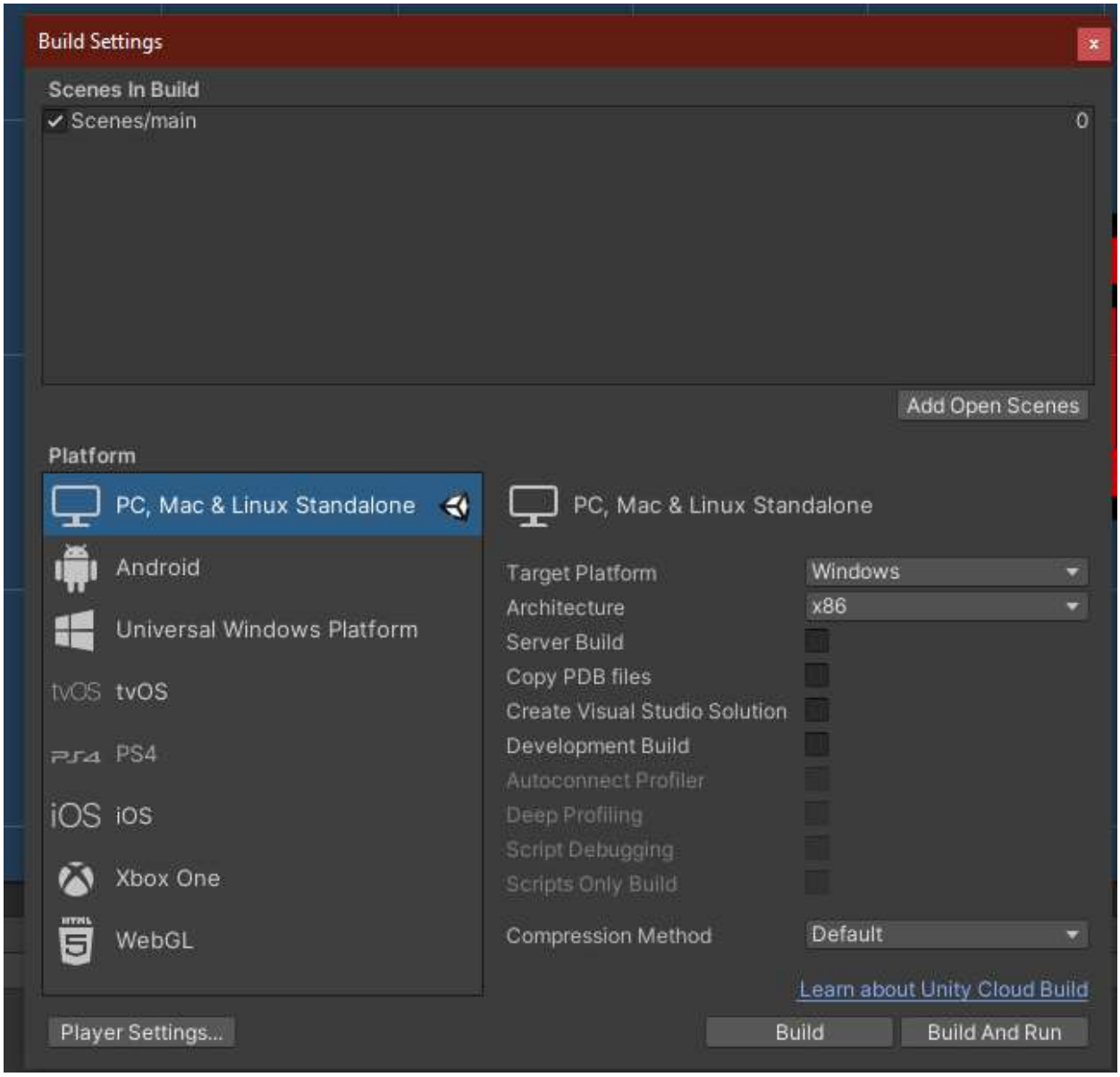

рис.12

Когда наша игра создана, нам нужно превратить её в приложение, мы разрабатывали игру для системы android. Значит нам нужно скомпилировать проект под него, однако тестировать игру на телефоне не удобно, благо в Unity простейшим способ можно сменить итоговую платформу. В связи с этим я решил всё же вывести игру на ПК.

Теперь у нас полноценная игра, в которой можно проиграть и начать заново. Да, это самый просто пример того, на что способен игровой движок Unity. Но со своими знаниями я не стал замахиваться на что-то большее. Кстати жанр такой игры называется bullet hell

рис.13

Заключение: в ходе написание этой проектной работы мною были рассмотрены различные среды разработки игр под систему Android, а также была проанализирована игровая индустрия. Под конец работы мне удалось создать собственную игру и поделится этим процессом с другими.

Устойчивость и распространение: Мой проект должен помочь людям, только начинающим разбираться в программировании и желающих создать собственное приложение на android. На примере моей работы они смогут узнать некоторые нюансы и не допустить ошибки в самом начале пути. Так же мой проект будет интересен любому человеку, который хочет узнать, как вообще работает приложение у него на телефоне, ведь сейчас у каждого есть смартфон и абсолютно каждый пользуется той или иной программой/приложением.

Использованные источники:

Википедия, Unity (игровой движок) – URL: https://ru.wikipedia.org/wiki/Unity_(%D0%B8%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA) (дата обращения 17.03.2021)

Википедия, Unreal Engine – URL: https://ru.wikipedia.org/wiki/Unreal_Engine (дата обращения 17.03.2021)

Википедия, Game Maker Studio – URL: https://ru.wikipedia.org/wiki/GameMaker:_Studio (дата обращения 17.03.2021)

Приложение 1

```
using UnityEngine;

public class MovePlayer : MonoBehaviour
{
    public Transform player;
    [SerializeField]
    private float speed = 10f;

    void OnMouseDrag()
    {

        if (!Player.lose)
        {
            Vector3 mousePos =
Camera.main.ScreenToWorldPoint(Input.mousePos
ition);
            mousePos.x = mousePos.x > 2.3f ?
2.3f : mousePos.x;
            mousePos.x = mousePos.x < -2.3f ?
-2.3f : mousePos.x;
            player.position =
Vector2.MoveTowards(player.position,
                new Vector2(mousePos.x,
player.position.y),
                speed * Time.deltaTime);
        }
    }
}
using UnityEngine;

public class Player : MonoBehaviour {
```

```
    public GameObject restart;
    public static bool lose = false;

    void Awake ()
    {
        lose = false;
    }

    void OnTriggerEnter2D(Collider2D other) {
        if (other.gameObject.tag == "Bomb") {
            lose = true;
            restart.SetActive(true);
        }
    }

}

using System.Collections;
using UnityEngine;

public class SpawnBombs : MonoBehaviour
{

    public GameObject bomb;

    void Start()
    {
        StartCoroutine(Spawn());
    }

    IEnumerator Spawn()
    {
```

```
        while (!Player.lose)
        {
            Instantiate(bomb, new
Vector2(Random.Range(-2.3f, 2.3f), 5.6f),
Quaternion.identity);
            yield return new
WaitForSeconds(0.6f);
        }
    }

}
```

```
using UnityEngine;

public class FallDown : MonoBehaviour
{
    [SerializeField]
    private float fallSpeed = 3f;




    void Update()
    {
        if (transform.position.y < -6f)
            Destroy (gameObject);


        transform.position -= new Vector3 (0,
fallSpeed * Time.deltaTime, 0);
    }
}
```

```
using UnityEngine;
```

```
using UnityEngine.SceneManagement;

public class Restart : MonoBehaviour
{

    void OnMouseDown()
    {
        SceneManager.LoadScene("main");

    }



}
```