Министерство образования Пензенской области
Открытый региональный конкурс исследовательских и проектных работ школьников «Высший пилотаж - Пенза» 2024
Муниципальное бюджетное общеобразовательное учреждение средняя общеобразовательная школа с. Бессоновка

Создание игрового приложения для android-устройств в среде разработки Unity

секция Computer science (информационные технологии, в рамках конференции «Авангард»)

Выполнил: ученик 10 Б класса
Кальченко Никита Сергеевич

Научный руководитель:
учитель информатики
Атаманова Татьяна Ивановна

с. Бессоновка

2023

Оглавление

Введение

Наступивший век информационных технологий диктует свои правила, увеличивая информационную и психологическую нагрузку на каждого члена общества. Это обстоятельство порождает потребность в частой смене деятельности, переключении на какой-то занимательный вид деятельности, чтобы снизить тревожность, накопившуюся за день. На помощь в решении данной проблемы пришли мобильные игровые приложения, не несущие тяжелой смысловой нагрузки, тем самым способствуя отдыху человека и переключению внимания с одной деятельности на другую.

Мобильные игры являются одним из основных продуктов на рынке информационных продуктов и услуг, так же следует отметить, что игровые приложения привносят большой вклад в развитие информационных технологий, программного и аппаратного обеспечения. Именно поэтому разработка игровых приложений на сегодняшний день является наиболее востребованной среди разработчиков мобильных приложений.

В связи с большим количеством устройств, работающих под операционной системой Android, разработка приложений под данную платформу очень актуальна.

Объект моего исследования: процесс разработки игровых приложений для операционной системы Android.

Предмет исследования: технология разработки игровых приложений для операционной системы Android при помощи инструмента Unity.

Цель исследования: создать технологию разработки игровых приложений и игровое приложение под управлением операционной системы Android при помощи платформы Unity.

Задачи:

1. Проанализировать особенности разработки приложений для различных мобильных платформ, таких как: Windows Phone, iOS, Android.
2. Рассмотреть возможности платформы Unity.
3. Раскрыть технологию разработки игровых приложений.
4. Разработать игровое приложение «Сапёр» средствами Unity для операционной системы Android.
5. Провести апробацию приложения на одноклассниках

1. Теоретические основы разработки приложений для android-устройств

1.1 Разработка приложения для android-устройств

Разработку приложений необходимо начинать с выбора подходящей платформы, для которой будет создаваться мобильное приложение. Наиболее востребованными и актуальными на сегодняшний день принято считать следующие платформы:

- Windows Phone – мобильная платформа, разработанная американской компанией Microsoft;
- iOS – мобильная платформа для смартфонов, электронных планшетов и носимых проигрывателей, разработанная американской компанией Apple;
- Android – мобильная платформа для смартфонов и множества других устройств, разработанная американской компанией Google.

На сегодняшний день Android является лидирующей мобильной операционной системой и занимает 82% на рынке мобильных устройств [9].

Разработка приложений под операционную систему Android имеет ряд особенностей. Приложения в Android используют четыре основных типа компонентов: деятельности, сервисы, слушатели сообщений, поставщики содержимого [3].

1. Деятельность (Activity) – это визуальный компонент приложения, который отвечает за интерфейс и это то, с чем можно взаимодействовать. Приложение может содержать как несколько деятельностей, так и вовсе не содержать их.
2. Сервисы (Services) – это то что выполняется, пока приложение не находится в фокусе. Сервис может запускаться вместе с системой и работать в фоновом режиме (например, музыкальный проигрыватель), выполняя некоторые действия. Взаимодействовать с сервисом можно посредством интерфейсов.
3. Слушатель событий (Listener) – один из важнейших компонентов приложения. Слушатель событий, также как и сервис, не имеет видимого интерфейса. Его задача отслеживать определенные действия или системные сообщения и реагировать на это. Так как сам по себе слушатель событий сделать ничего не может, он передает сигнал дальше (пользователю приходит уведомление, по событию вызывается деятельность и т.д.).
4. Поставщик содержимого (Content provider) предоставляет определенные данные другим приложениям. Эти данные могут храниться в файловой системе, SQLite базе данных и т.д.

Одним из уникальных качеств Android является то, что все программы имеют общий уровень. Android не проводит различия между основными системными программами и программами сторонних разработчиков. Все они пользуются равными правами доступа к возможностям мобильного устройства, предоставляя пользователям широкий спектр приложений и услуг. С устройствами, построенными на платформе Android, пользователи будут иметь возможность в полной мере адаптировать устройство

под свои интересы.

1.2 Анализ возможностей платформы UNITY

Unity – кроссплатформенный игровой движок для разработки двух- и трехмерных приложений со встроенной технологией IDE. Он является одним из наиболее популярных игровых движков для платформы Android. Он применяется для разработки видеоигр для веб-платформ, платформ настольных ПК, игровых консолей и мобильных устройств, и используется более чем миллионом разработчиков. Разработка приложений ведется для следующих платформ: Windows, MacOS, Wii, Apple iOS, Android, PS3 и XBox 360. Для Unity3D предусмотрено несколько лицензий: Unity 5 Personal Edition (бесплатная), Unity 5 Professional Edition, iOS Pro, Android Pro и групповая лицензия Team License.

Особенности Unity, которые являются причиной выбора данного движка для реализации игрового приложения [7]:

- кроссплатформенность;
- наличие бесплатной версии с некоторыми ограничениями.
- возможность создания 2D и 3D приложений;
- возможность писать на двух языках программирования: JavaScript, C#;
- возможность создания игры любого жанра;
- возможность мгновенного запуска игры через встроенный эмулятор;
- IDE: сочетание редактора сцен (в комплексе общего редактора) с ре-дактором игровых объектов и редактора скриптов;
- интуитивно понятный и полностью настраиваемый интерфейс;
- работа с ресурсами через Drag-and-Drop. Drag-and-Drop (в пер. с англ. тащи и бросай) – это способ оперирования элементами интерфейса при помощи манипулятора «мышь» или сенсорного экрана, который позволяет захватить элемент и перенести его;
- наличие магазина Asset Store, в котором можно найти: 3D модели, расширения редактора, скрипты, шейдеры (компьютерные програм- мы, предназначенные для исполнения процессорами видеокарты), звуковые файлы, анимацию и многое другое.

1.3 Техническое задание на разработку игрового приложения

1) Общие сведения.
    a. Название продукта разработки: Сапёр.
    b. Назначение продукта: игровое приложение, головоломка, развивающая память, внимание и интуицию.
2) Требования к продукту разработки.
    a. Аппаратные требования:

Таблица 1.Аппаратные требования

	Минимальные	*Рекомендуемые*
Платформа	Android 5.0	Android 5.0 и выше

Частота процессора	1 ГГц	1.3 ГГц
Количество ядер	1	2
Сенсорный экран	да	да
Разрешение экрана	480x800	1280x720
Оперативная память	512 Мб	1024 Мб
Свободное место на	30Мб	50Мб

b. Указание системного программного обеспечения:
   - Операционная система Android;

c. Указание программного обеспечения, используемого для реализации:
   - Unity 5 Personal Edition;
   - Visual Studio.

3) Требования к пользовательскому интерфейсу.

a. Общая характеристика пользовательского интерфейса: пользователю предоставляется главное меню с возможностью начать новую игру, выбор размера игровой сетки минного поля, расстановки мин. Режим игры многопользовательский.

b. Особенности ввода информации пользователем, представление выходных данных: входные данные передаются с сенсорного экрана и сохраняются на время игры.

2. Разработка игрового приложения на UNITY под Android

2.1 Технология разработки игрового приложения для операционной системы android

Разработка приложения осуществляется по следующему алгоритму [8]:

1. *Выбор жанра игры.*

Список общеизвестных жанров:

- аркадные – компьютерные игры с нарочно примитивным игровым процессом (бегает, прыгает);
- платформер – жанр компьютерных игр, в которых основной чертой игрового процесса является перемещение по платформам и лестницам, сбор предметов, обычно необходимых для завершения уровня;
- приключение – игра, обладающая целостным сюжетом, который пишут сценаристы;
- экшн – жанр компьютерных игр, в которых успех игрока в большей степени зависит от его скорости реакции и способности быстро принимать тактические решения;
- развивающие мышление и головоломки;
- казуальные игры – компьютерная игра, предназначенная для широкого круга пользователей, в которую играют от случая к случаю, чтобы как-то «убить» время;
- ролевая игра – игра, в которой пользователь управляет персонажем или

группой персонажей, обладающих определенным набором навыков и умений;

- симуляторы – игры, которые полностью или частично имитирует определенную сферу реальной жизни;
- обучающие – игры, которые обладают элементом обучающей программы;
- спортивные игры.
- Также стоит отметить, что существует несколько классификаций игр помимо вышеприведенной классификации жанров.
- Классификация игр по платформам:
- персональный компьютер (ПК, PC, ноутбук, нетбук);
- игровая консоль или приставка (PS, Xbox, Nintendo);
- мобильное устройство: телефон, планшет, карманный компьютер(КПК, PDA);
- игровой автомат;
- браузерная или флеш-игра (виртуальная интернет платформа).

Классификация игр по количеству платформ:

- мультиплатформенные игры (вышедшие на двух и более платформах);
- одноплаторменные игры (эксклюзивны в рамках одной платформы).

Классификация игр по количеству игроков:

- однопользовательская;
- многопользовательская;
- многопользовательские игры на одном компьютере;
- массовые онлайновые.

2. *Разработка сюжета и структуры проекта.*
3. *Выбор средства разработки.*
4. *Создание рабочего прототипа.*
5. *Тестирование.*
6. *Релиз.*

2.2 Разработка игрового приложения

Для игры был выбран жанр головоломки, т.к. игра предназначена для развития памяти, внимания и интуиции. Разработанное приложение создано для мобильных устройств под управлением операционной системы Android, из чего следует вывод, что игра является одноплатформенной.

Перед непосредственной разработкой игрового приложения на бесплатных ресурсах сети интернет были подобраны фон и оформление других элементов интерфейса, создана анимация взрывов.

Описание игры: игра задумывалась, как многопользовательская. В данный момент я занимаюсь разработкой серверной части, чтобы реализовать этот сценарий в полной мере.

Вызвав приложение на исполнение игрок видит главное меню (Рис.1) После нажатия кнопки Play появляется экран настроек (Рис. 2), где предлагается выбрать размер минного поля и сохранить его. Затем пользователь видит поле игры (Рис. 3), где первый игрок можно задать расположение мин и безопасный путь. После нажатия кнопки Play в левом верхнем углу другие игроки могут попытаться пройти минное поле, управляя героем

игры, воином, с помощью кнопок стрелок. Если герой наступает на мину, она взрывается с соответствующей анимацией взрыва и герой погибает (Рис. 4). В правом верхнем углу можно увидеть количество неудачных попыток.

Скриншоты игры:

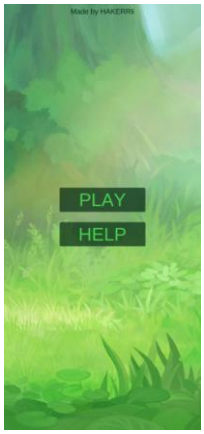

Рис. 1

Рис. 2

Рис. 3

Рис. 4

Коды игры представлены в Приложении.

2.3 Результаты апробации

Для реализации апробации было принято решение провести тестирование приложения учащимися 10Б класса МБОУ СОШ с. Бессоновка. Для выявления ошибок была создана анкета с вопросами, которая заполнялась после тестирования приложения.

Анкета «Результаты тестирования приложения»

1. Укажите модель телефона.
2. Укажите версию операционной системы Android.
   a) Android 5.0; b)Android 6.1; c)Android 9.0 и выше.
3. Укажите разрешение экрана устройства, на которое было установлено приложение.
4. Были проблемы при установке приложения?
   a) Да; b)Нет
5. Были ошибки в процессе игры, если да, то какие?
6. Удобный интерфейс?
   a) Да; b)Нет
7. Удалось ли пройти минное поле?
   a) Да; b)Нет
8. Понравилась ли игра?
   a) Да; b) Нет; c)Затрудняюсь ответить
9. Оцените приложение по шкале от 1 до 5.
10. Пожелания для улучшения игрового приложения.

Результаты тестирования показали следующее:

1. Игра была установлена и работала под разными версиями операционнойсистемы Android.
2. Элементы интерфейса отражаются без критичных искажений на экранелюбого разрешения.
3. Проблем с установкой не возникло.
4. Критичных ошибок в процессе игры не было.

5. Интерфейс посчитали удобным 100% тестирующих.
6. Пройти минное поле и сохранить жизнь удалось 70,% тестирующих.
7. Средняя оценка приложения составила

Заключение

Исследование показало, что операционная система Android на сегодняшний день является наиболее востребованной, как у разработчиков, так и пользователей. Также было доказано, что Unity обладает разнообразным набором ресурсов позволяющих упростить создание игровых приложений.

В ходе исследования:

1. Проанализированы особенности процесса запуска и исполнения приложений операционной системой Android;
2. Произведено сравнение мобильных платформ.
3. Рассмотрены особенности программирования под операционную системуAndroid.
4. Рассмотрены особенности работы операционной системы Android.
5. Разработано игровое приложение «Сапёр» средствами Unity для операционной системы Android;
6. Проведена апробация разработанного игрового приложения в форме тестирования.

В планах разработка серверной части игры.

Список литературы

1. Голощапов А.Л. Google Android программирование для мобильных устройств. - СПб: 2011.
2. Майер Р. Программирование приложений для планшетных компьютеров и смартфонов. - М.: Эксмо, 2011.
3. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android. - М.: Питер, 2011.
4. Цехнер М. Программирование игр для Android . Питер, 2013.
5. Goldstone W. Unity3D Game Development Essentials. 2009.
6. Нахавандипур В. iOS. Разработка приложений для iPhone, iPad и iPod.- СПб: Питер, 2013. - 864 с.
7. Официальный сайт Unity URL: https://unity.com/ru
8. Официальный сайт Android URL: http://www.android.com
9. Статистика мобильных операционных систем за ноябрь 2023 года https://gs.statcounter.com/os-market-share/mobile/worldwide

Приложение

ComingSoonBehaviour Это код который отвечает за появление и дальнейшее поведение объявления «Coming Soon...»

```
using System.Collections;
using UnityEngine;

public class ComingSoonBehaviour : MonoBehaviour
{
   //private int c = 0;
   private RectTransform comingSoon;
   // Start is called before the first frame update
   void Start()
   {
      comingSoon = GetComponent<RectTransform>();
      comingSoon.anchoredPosition = new Vector2(0, -5270);
      StartCoroutine(ShowCoroutine());
   }

   IEnumerator ShowCoroutine()
   {
      for (int i = 0; i < 100; i++) {
         comingSoon.anchoredPosition = new Vector2(comingSoon.anchoredPosition.x, comingSoon.anchoredPosition.y +
100 - i);
         yield return new WaitForSeconds(0.03f);
      }
      Destroy(gameObject, 1f);
   }
}
```

GridScript Это код который отвечает за поведение сетки игрового поля «Coming Soon...»

```
using UnityEngine;
using UnityEngine.UI;
using Vector2 = UnityEngine.Vector2;
using Vector3 = UnityEngine.Vector3;
using Quaternion = UnityEngine.Quaternion;

public class GridScript : MonoBehaviour
{
   [SerializeField]
   GameObject planePrefab;

   [SerializeField]
   private RectTransform gridTransform;

   public int sizeXY = 12;
   public RectTransform playerTransform;
   public RectTransform[] anchors;
   public Text fieldSize;

   public void setfieldsize(int n)
   {
      if (sizeXY > 5 && n < 0)
      {
         sizeXY += n;
      }
      if (sizeXY < 15 && n > 0)
      {
         sizeXY += n;
      }
      fieldSize.text = "size: " + sizeXY;
   }

   public void generateField()
   {
```

```
        gridTransform = GetComponent<RectTransform>();
        var glg = gridTransform.GetComponent<GridLayoutGroup>();
        int sz = 690 / sizeXY - 10;
        glg.cellSize = new Vector2(sz, sz);
        for (int i = 0; i < sizeXY; i++)
        for (int j = 0; j < sizeXY; j++)
        {
            GameObject obj = Instantiate(planePrefab, Vector2.zero, Quaternion.identity, transform);
            RectTransform rt = obj.GetComponent<RectTransform>();
            rt.anchorMin = new Vector2 (i*1.0f/sizeXY, j*1.0f/sizeXY);
            rt.anchorMax = new Vector2((i+1)*1.0f/sizeXY, (j+1)*1.0f/sizeXY);
            rt.offsetMin = Vector2.zero; rt.offsetMax = Vector2.zero;
        }
        playerTransform.SetAsLastSibling();
    }

    public Vector2 getSize() {
        var size = new Vector2();
        size.x = Vector2.Distance(anchors[0].anchoredPosition, anchors[1].anchoredPosition);
        size.y = Vector3.Distance(anchors[1].anchoredPosition, anchors[2].anchoredPosition);
        Debug.Log(size.x + " / " + size.y);
        return size;
    }
}
```

PlaneScript Это код который отвечает за поведение плиток игрового поля

```
using UnityEngine;
using UnityEngine.UI;

public class PlaneScript : MonoBehaviour
{
    public bool Enable;
    public bool Game;
    public RectTransform player;
    public RectTransform plane;

    // Start is called before the first frame update
    void Start() {
        plane = GetComponent<RectTransform>();
        player = plane.parent.GetComponent<GridScript>().playerTransform;
        Enable = true;
        Game = false;
    }

    // Update is called once per frame
    void Update() {
        if (Vector3.Distance(plane.anchoredPosition, player.anchoredPosition) <= 4) {
            if (Game && Enable) {
                player.GetComponent<PlayerScript>().Boom();
            }
        }
        Game = player.GetComponent<PlayerScript>().Game;

        if (Game) {
            transform.GetComponent<Image>().color = Color.white;
        }
    }

    public void win() {
        if (!Enable) {
            player.GetComponent<PlayerScript>().Game = false;
            transform.GetComponent<Image>().color = Color.black;
        }
    }
```

```
    public void StartGame() {
        Game = player.GetComponent<PlayerScript>().Game;
        if (Game) {
            transform.GetComponent<Image>().color = Color.white;
        }
    }

    public void ClickUIButton() {
        Game = player.GetComponent<PlayerScript>().Game;
        if (!Game) {
            Enable = !Enable;
            if (!Enable) {
                transform.GetComponent<Image>().color = Color.black;
            } else {
                transform.GetComponent<Image>().color = Color.white;
            }
        }
    }

    public void ClickUIPlayer() {
        Game = player.GetComponent<PlayerScript>().Game;
        if (!Game) {
            if (Vector3.Distance(plane.anchoredPosition, player.anchoredPosition) <= 4) {
                Enable = !Enable;
                if (!Enable) {
                    transform.GetComponent<Image>().color = Color.black;
                } else {
                    transform.GetComponent<Image>().color = Color.white;
                }
            }
        }
    }
}
```

PlayerScript Это код который отвечает за поведение объекта игрок

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Vector2 = UnityEngine.Vector2;

public class PlayerScript : MonoBehaviour
{
    public RectTransform player;
    public Transform gridTransform;
    public RectTransform[] planes;

    public bool Game;
    public bool Buttons;
    public int playerClick;
    public Text playButtonText;
    public Text dethCounter;

    private Animator CharacterController;
    private int deth;


    // Start is called before the first frame update
    void Start() {
//      Debug.Log("Start Boom: " + BOOM);
        CharacterController = GetComponent<Animator>();
        player = GetComponent<RectTransform>();
```

```
        gridTransform = player.parent;
        dethCounter.text = "Deth: " + deth;

        Game = false;
        Buttons = true;
        deth = 0;

        StartCoroutine(MoveArrowCoroutine());
    }

    // Update is called once per frame
    //void Update() {
    //    player.GetComponent<Button>().Select();

    //    player.SetAsLastSibling();

    //    if (BOOM) {
    //        respawn();
    //    }
    //}

    public void up() {
        if (player.anchoredPosition.y < 0 && Buttons) {
            player.anchoredPosition = new Vector2(player.anchoredPosition.x, player.anchoredPosition.y + 690 /
gridTransform.GetComponent<GridScript>().sizeXY);
            if (player.anchoredPosition.y > 0) {
                Buttons = false;
                for (int i = 0; i < gridTransform.childCount - 1; i++) {
                    RectTransform plane = gridTransform.GetChild(i).GetComponent<RectTransform>();
                    plane.GetComponent<PlaneScript>().win();
                }
            }
        }
```

```
    public void down() {
        if (player.anchoredPosition.y > -700 && Buttons) {
            player.anchoredPosition = new Vector2(player.anchoredPosition.x, player.anchoredPosition.y - 690 /
gridTransform.GetComponent<GridScript>().sizeXY);
        }
    }

    public void right() {
        if (player.anchoredPosition.x < 690 - ((690 / gridTransform.GetComponent<GridScript>().sizeXY - 10)) && Buttons) {
            player.anchoredPosition = new Vector2(player.anchoredPosition.x + 690 /
gridTransform.GetComponent<GridScript>().sizeXY, player.anchoredPosition.y);
        }
    }

    public void left() {
        if (player.anchoredPosition.x > ((690 / gridTransform.GetComponent<GridScript>().sizeXY - 10) / 2 + 5) + 1 &&
Buttons) {
            player.anchoredPosition = new Vector2(player.anchoredPosition.x - 690 /
gridTransform.GetComponent<GridScript>().sizeXY, player.anchoredPosition.y);
        }
    }

    public void Boom() {
        Buttons = false;
        CharacterController.SetBool("BOOM", true);
    }

    public void respawn() {
        Buttons = true;
        deth++;
        dethCounter.text = "Deth: " + deth;
```

```
        CharacterController.SetBool("BOOM", false);
    }

    public void GameVoid() {
        if (!Game & playButtonText.text == "Play") {
            playButtonText.text = "Replay";
            Game = true;
        } else {
            Reload();
        }
    }

    public void ClickPlayer() {
        foreach (Transform code in planes)
            code.GetComponent<PlaneScript>().ClickUIPlayer();
    }

    public void Reload() {
        SceneManager.LoadScene(Application.loadedLevel);
    }

    IEnumerator MoveArrowCoroutine() {
        ulong state = 0;
        while (true) {
            if (Input.GetKey(KeyCode.UpArrow) && (player.anchoredPosition.y < 0) && Buttons) {
                player.anchoredPosition = new Vector2(player.anchoredPosition.x, player.anchoredPosition.y + 690 /
gridTransform.GetComponent<GridScript>().sizeXY);
                if (player.anchoredPosition.y >= 0){
                    Buttons = false;
                    for (int i = 0; i < gridTransform.childCount - 1; i++)
                    {
                        RectTransform plane = gridTransform.GetChild(i).GetComponent<RectTransform>();
                        plane.GetComponent<PlaneScript>().win();
                    }
                }
                ++state;
            } else if (Input.GetKey(KeyCode.DownArrow) && (player.anchoredPosition.y > -700) && Buttons) {
                player.anchoredPosition = new Vector2(player.anchoredPosition.x, player.anchoredPosition.y - 690 /
gridTransform.GetComponent<GridScript>().sizeXY);
                ++state;
            } else if (Input.GetKey(KeyCode.RightArrow) && (player.anchoredPosition.x < 690 - ((690 /
gridTransform.GetComponent<GridScript>().sizeXY - 10) / 2 + 5) - 3) && Buttons) {
                player.anchoredPosition = new Vector2(player.anchoredPosition.x + 690 /
gridTransform.GetComponent<GridScript>().sizeXY, player.anchoredPosition.y);
                ++state;
            } else if (Input.GetKey(KeyCode.LeftArrow) && (player.anchoredPosition.x > ((690 /
gridTransform.GetComponent<GridScript>().sizeXY - 10) / 2 + 5)) && Buttons) {
                player.anchoredPosition = new Vector2(player.anchoredPosition.x - 690 /
gridTransform.GetComponent<GridScript>().sizeXY, player.anchoredPosition.y);
                ++state;
            } else {
                state = 0;
            }
            //Debug.Log("state: " + state);
            if (state == 1) {
                yield return new WaitForSeconds(0.5f);
            } else if (state > 1) {
                yield return new WaitForSeconds(0.2f);
            } else {
                yield return new WaitForSeconds(0.1f);
            }
        }
    }
}
```

PlayModeBehaviourScript Это код который отвечает за поведение игровых меню

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayModeBehaviourScript : MonoBehaviour
{
    private RectTransform trnsfrm;
    public GameObject comingSoonPrefab;

    void Start() {
        trnsfrm = GetComponent<RectTransform>();
    }

    public void on() {
        StartCoroutine(OnCoroutine());
    }

    public void off() {
        StartCoroutine(OffCoroutine());
    }

    public void comingSoon() {
        GameObject obj = Instantiate(comingSoonPrefab, Vector3.zero, Quaternion.identity, transform);
        RectTransform rt = obj.GetComponent<RectTransform>();
    }

    IEnumerator OnCoroutine() {
        for (int i = 0; i < 100; i++) {
            trnsfrm.anchoredPosition = new Vector2(trnsfrm.anchoredPosition.x - 100 + i, trnsfrm.anchoredPosition.y);
            yield return new WaitForSeconds(0.03f);
        }
```

Рецензия на практико-ориентированный исследовательский проект

«Создание игрового приложения для android-устройств в среде разработки Unity» ученика 10 класса МБОУ СОШ с. Бессоновка Бессоновского района Пензенской области Кальченко Никиты

Общая оценка работы:

Практико-ориентированная исследовательская работа выполнена на тему «Создание игрового приложения для android-устройств в среде разработки Unity». Тема выбрана не случайно, ведь спрос на мобильные компьютерные игры постоянно растёт, что открывает широкие перспективы для применения данного проекта. В работе выдержаны все части: введение, теоретическая часть, основная (практическая часть), заключение и список используемых источников информации. Практическая часть преобладает над теоретической. В ней представлена действующая компьютерная игра, полностью созданная учащимся.

Очень толково и подробно составлена технологическая карта проекта с поставленной целью и задачами и обоснованием актуальности работы.

Теоретическая часть содержит информацию о мобильных операционных системах и приложениях, классификациях игр, основных моментов и правил создания компьютерных игр, освоенностях платформы Unity . Теоретическая часть соответствует выбранной теме.

Практическая часть описана логично, подробно. Представлены необходимые пояснения и скриншоты. В приложении отражён основной программные коды игры. Представлены так же и сами файлы программного продукта, представляющего собой успешно работающую компьютерную игру с удобным меню и приятным интерфейсом, в которой прослеживается указанный в работе сценарий.

В заключение по работе сделаны разноплановые, обоснованные выводы.

Оформление работы соответствует предъявляемым критериям.

С данной работой ученик направляется для участия в Региональном конкурсе исследовательских и проектных работ школьников «Высший пилотаж - Пенза»

Рекомендации:

продолжить работу по данной теме, усовершенствовав уже созданный продукт.

Заключение

Работа соответствует требованиям, предъявляемым к исследованиям подобного рода и заслуживает высокой оценки.

Рецензент
учитель информатики
МБОУ СОШ с. Бессоновка  Т.И. Атаманова