МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент

Ведущий разработчик

ООО «ВОРТЕКСКОД»

_____ П.А. Михайлов

“___”_____ 2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2018 г.

РАЗРАБОТКА КОМПЬЮТЕРНОЙ ИГРЫ С ИСПОЛЬЗОВАНИЕМ ИГРОВОГО ДВИЖКА UNITY

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2018. 308-06.ВКР

Научный руководитель

старший преподаватель кафедры СП

_____К.Ю. Никольская

Автор работы,
студент группы КЭ-402

_____В.С. Ковалев

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2018 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Человек всегда уделял большое внимание развлечениям. Постоянно совершенствуются старые и появляются новые способы развлечений. Компьютерные игры и видеоигры в наше время являются одним из самых популярных методов развлечения и могут потеснить по прибыли даже индустрии кино и музыки. Также, компьютерные игры двигают прогресс в области разработок нового компьютерного оборудования и обеспечивают его продажи.

В 2016 году мобильные игры впервые вышли на второе место по объему рынка в России, обогнав сегмент социальных игр. В 2015 году социальные и мобильные игры занимали примерно равные доли: 24% и 20% соответственно. В 2016 году «мобилки» заполучили уже треть рынка — их доля составила 29%, что на 9% больше, чем в предыдущем периоде. При этом «социалки» сместились на третье место с долей в 18%. В денежном выражении за последние пять лет рынок мобильных игр увеличился в 10 раз и достиг отметки в 16,3 миллиарда рублей [3].

Кроме того, уже продолжительное время киберспорт воспринимают всерьез. Россия Стала первой страной, признавшей киберспорт как один из официальных видов спорта. А с 5 июля 2017 года приказом Министерства спорта Российской федерации была основана Федерация компьютерного спорта России [14].

Цель и задачи

Целью работы является разработка компьютерной игры с использованием движка Unity.

Для достижения данной цели должны быть решены следующие задачи.

1. Провести исследование компьютерных игр схожего жанра, изучить разработку игр на движке Unity.

2. Спроектировать игровое приложение.

3. Реализовать игровое приложение с помощью игрового движка Unity.

4. Провести тестирование игрового приложения.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, библиографического списка и двух приложений. Объем работы без приложений составляет 34 страниц, объем приложений составляет 6 страниц, объем библиографии – 15 источников.

Содержание работы

В первой главе «Анализ предметной области» проведен обзор программных средств разработки и обзор существующих игр схожего жанра.

Вторая глава «Проектирование игрового приложения» посвящена определению требований к разрабатываемому игровому приложению. Также в этой главе рассматривается диаграмма вариантов использования и диаграмма компонентов.

В третьей главе «Реализация игрового приложения» описаны методы реализации, приведены скрипты приложения. В этой же главе рассмотрена файловая структура игрового приложения.

В четвертой главе «Тестирование игрового приложения» представлены результаты тестирования игрового приложения.

В заключении описываются основные результаты, полученные при выполнении работы.

В приложении 1 представлены скриншоты работающего игрового приложения.

В приложении 2 представлен отчет юзабилити-тестирования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор программных средств разработки

Unity – игровой движок и среда разработки трехмерных и двухмерных игр, а также различных приложений, поддерживающий свыше двадцати 25 платформ, среди которых Windows, Android, OS X, IOS, PlayStation 4, Xbox One, Nintendo Switch. Unity также имеет поддержку технологий VR и AR для использования в играх и приложениях. На базе Unity сделано больше игр, чем с любой другой игровой технологией, в них играют все больше и больше игроков, а инструменты и сервисы Unity предпочитают все больше и больше разработчиков: 34 % из топ 1000 бесплатных мобильных игр разработаны на Unity, среди клиентов Unity есть Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square Enix, Ubisoft, Obsidian, Insomniac и Warner Bros. В игры, созданные с использованием движка Unity играют более 770 млн. пользователей по всему миру [6].

Unity обладает интуитивным интерфейсом, поддерживающим Drag&Drop и его можно гибко располагать в рамках окна приложения [5]. Любое приложение на Unity состоит из сцен, которые в свою очередь содержат различные игровые объекты, обладающие набором поведенческих скриптов и других компонентов [11, 15].

В качестве языков программирования Unity позволяет выбрать между языками C# и JavaScript. Удобство и строгая типизация языка C# сподвигло выбрать его в качестве языка программирования для написания пользовательских скриптов [1, 12].

1.2. Обзор существующих решений

Nebulus – игра в жанре платформер, разработанный Triffix Entertaiment Inc и изданный HewSon Consultants в 1987 для таких платформ как ZX Spectrum, Commodore 64 и Commodore Amiga. В ней игрок управляет маленьким существом, которое зовут Pogo. Его задачей стоит уничтожить восемь башен и для этого ему нужно взобраться на каждую из

них. К плюсам можно отнести инновационную графику – когда игрок передвигается влево и вправо, то он остается на месте в то время, как башня двигается, создавая эффект объема. К минусам можно отнести устаревшую графику и примитивное звуковое сопровождение.

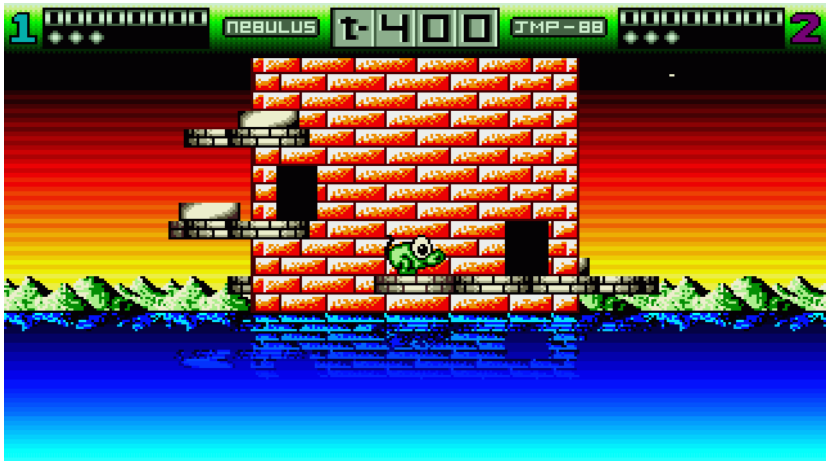


Рис. 1. Скриншот из игры Nebulus

Super Kong Climb – аркада для IOS. Её суть заключается в том, что в роли Кинг Конга необходимо взбираться по бесконечному небоскребу [9]. Сверх на игрока скидываются бомбы и в его задачу входит уклонение от них. Плюсами данной игры можно назвать приятное визуальное оформление, соревновательный элемент – каждая бомба, от которой увернулся игрок дает ему очки. Минусами этой игры является малое количество игровых возможностей и дополнительная внутриигровая монетизация.

Рис. 2. Скриншот из игры Super Kong Climb

Upwards, Lonely Robot – игра в жанре аркада, где в роли робота необходимо путешествовать по башням в поисках своих создателей, решая различного рода головоломки [7]. Плюсами игры является наличие многопользовательского режима. Минусом можно назвать излишнюю затянутость. На рис. 3 представлен скриншот из игры Upwards, Lonely Robot.

Рис. 3. Скриншот из игры Upwards, Lonely Robot

2. ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ

2.1. Эскизный проект игрового приложения

Концепция игры

Rapunz Tower – представленное в работе игровое приложение, которое представляет из себя аркадную игру для персональных компьютеров. В задачу игрока входит в роли рыцаря взобраться по башне к принцессе. Игроку нужно быть внимательным и активным, поскольку у него есть временное ограничение.

Цели игры

Цель игры заключается в прохождении трех уровней игры и наборе очков. Задача каждого уровня состоит в том, чтобы добраться до принцессы по башне.

Правила игры

Каждый уровень начинается с того, что игрок находится в самом низу башни и ему необходимо карабкаться вверх и вокруг башни. Ему необходимо собирать бонусы для пополнения здоровья и набора очков. В игре есть окна с гоблинами, которые кидаются в игрока камнями, попадания которых отнимает у игрока здоровье. Если уровень здоровья персонажа игрока достигает нуля, то уровень считается проигранным. Также снизу игрока преследует терновый куст, прикосновение с которым приводит к незамедлительному проигрышу уровня. При достижении принцессы на вершине башни уровень считается пройденным

Игровые возможности

Игрок может передвигаться по башне в любую сторону, а также собирать различные бонусы.

Игроки

Игра является однопользовательской и в ней нет режима для нескольких пользователей.

Атмосфера игры

Игровое поле является трехмерным и содержит некоторое количество блоков башни, количество которых может различаться на разных уровнях. Камера находится за спиной игрока и передвигается вместе с ним. Графика реализована с использованием трехмерных моделей и текстур.

Интерфейс

Интерфейс приложения состоит из главного меню, внутриигрового меню и игровой панели.

В главном меню находятся кнопки «Start game» и «Exit from game». При нажатии кнопки «Start game» происходит открытие первого уровня. Нажатие клавиши «Exit from game» приводит к выходу из приложения.

Внутриигровое меню открывается нажатием на кнопку игровой панели, находящейся в правом верхнем углу. Внутриигровое меню может быть вызвано на только на игровых уровнях. При вызове внутриигрового игровой процесс приостанавливается до выхода из него. Внутриигровое меню состоит из кнопок «Return», «Restart», «Exit to main menu», «Exit from game». Нажав кнопку «Return», или повторно на кнопку игровой панели, в правом верхнем углу, игра будет возобновлена на том моменте, на котором она была остановлена. При нажатии кнопки «Restart» произойдет перезапуск уровня, при этом все набранные на этом уровне очки будут сброшены. Кнопки «Exit to main menu» и «Exit from game» прекращают игровой процесс, но в первом случае происходит возврат в главное меню игры, а во втором игровое приложение прекращает работу.

Игровая панель находится на верху экрана и состоит из нескольких компонентов: счетчик количества жизней игрока, счетчик очков и кнопки вызова внутриигрового меню.

Объекты игры

К объектам игры относятся игрок, гоблины в окошках, каменные блоки, булыжники, которыми кидаются гоблины, принцесса на балконе и различные бонусы. Существует три вида бонусов. Один восполняет игроку

здоровье и дает небольшое количество очков, другой дает больше очков, но не восполняет здоровье, а последний бонус делает игрока временно неуязвимым к булыжникам.

Игровые уровни

Игровые уровни отличаются размером башни, а также местом расположения и количеством бонусов и препятствий.

2.2. Диаграмма вариантов

На рис. 4 приведена диаграмма вариантов использования.

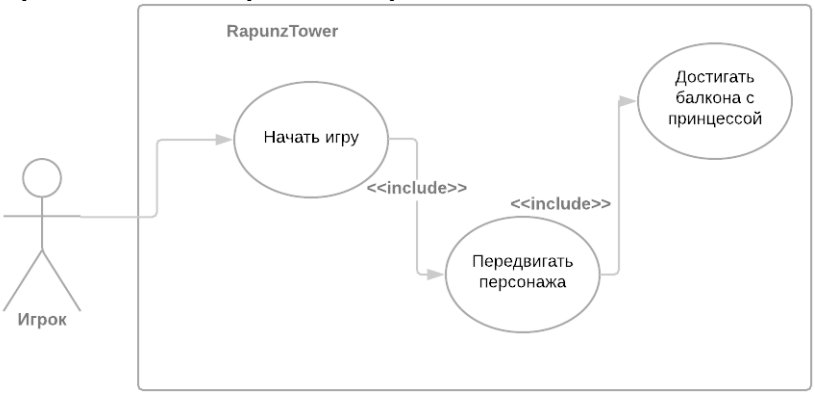


Рис. 4. Диаграмма вариантов использования

Для проектирования приложения был использован язык графического описания для объектного моделирования UML [2, 13]. Была построена модель взаимодействия внешнего актера с игровым приложением в виде диаграммы вариантов использования.

В ходе проектирования был выделен актер ***игрок***. Игрок – неавторизованный пользователь приложения.

Начать игру – приступить к игровому процессу. Реализуется с помощью кнопки «Start game» главного меню, «Return» внутриигрового меню и «Restart» внутриигровых меню.

Передвигать персонажа – нажать и клавиши-стрелки на клавиатуре.

Собрать бонусы – Передвинуть персонажа игрока к месту расположения бонуса.

Поставить на паузу – нажать клавишу на кнопку в правом верхнем углу на сценах «level1», «level2», «level3».

Закончить игру – закончить игровой процесс. Реализуется с помощью кнопки «Exit from game» главного меню кнопки «Exit from game» внутриигрового меню.

2.3. Диаграмма компонентов

Диаграмма компонентов отображена на рисунке 5.Приложение состоит из четырех основных логических блоков:

1. User Scripts.
2. Menu.
3. Resources.
4. Levels.

Компонент User Scripts представлен в виде ряда артефактов – файлов с расширением *.cs, основными из которых являются: «Balcony.cs», «BonusCollect.cs», «DeadlyGrass.cs», «UIControl.cs», «Pebel.cs», «PlayerMove.cs» и «WindowFrame».

Компонент Menu содержит элементы главного меню игры и игровых меню.

Компонент Resources содержит файлы со шрифтами и графическим оформлением сцен.

Компонент Levels материализован в форме ряда артефактов – систем объектов, каждая из которых представляет собой отдельный уровень

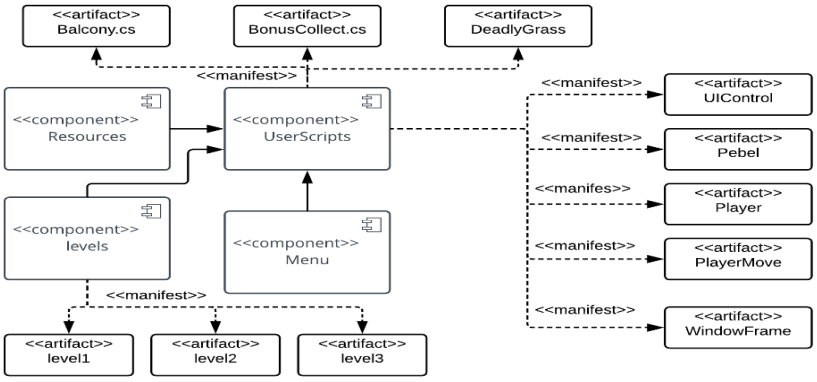


Рис. 5. Диаграмма компонентов

3. РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ

3.1. Файловая структура приложения

Проект разработанного игрового приложения состоит из списка каталогов, содержащих Материалы; 3D-моделей и их анимации; готовых компонентов, которыми являются игровые объекты с заданными свойствами для повторного использования; сцен; игровых скриптов; текстуры моделей и неба. Файловая структура приложения представлена на рис. 6.

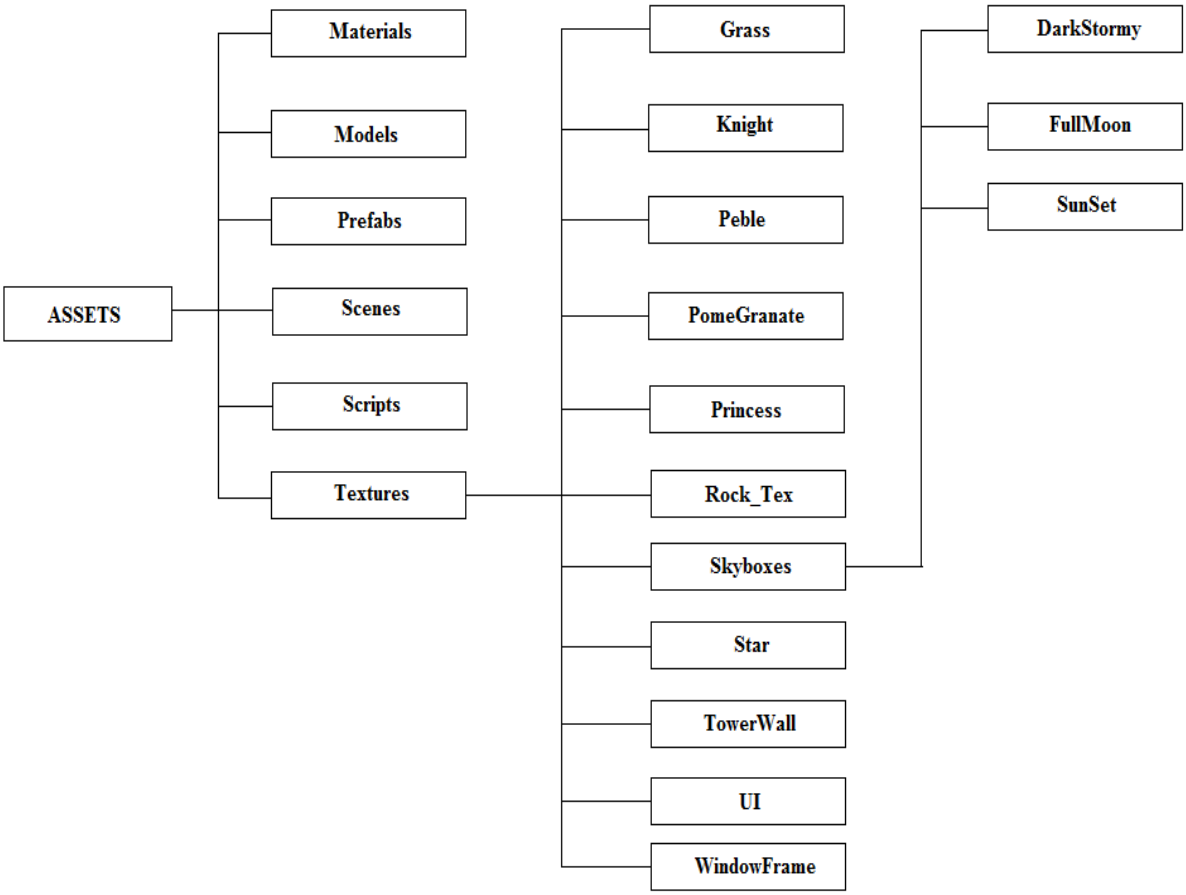


Рис. 6. Файловая структура приложения

3.2. Интерфейс

Главное меню

Главное меню является отдельной сценой и содержит кнопки «Start game» и «Exit from game» (рис. 1 приложения 1). При нажатии на кнопку

«Start game» загружается сцена с первым уровнем, а при нажатии на кнопку «Exit from game» происходит выход из игры.

Внутриигровое меню

Внутриигровое меню — это объект сцены, отвечающий за прорисовку и взаимодействия с интерфейсом игры. В Unity все объекты, относящиеся к пользовательскому интерфейсу, являются дочерними к объекту, который по умолчанию называется Canvas.

Игровой интерфейс приложения состоит из двух окошек, кнопки, показывающей панель, с кнопками взаимодействия, панели с кнопками, позволяющими выйти из игры, продолжить начать уровень сначала и вернуться на начальный экран игры. Код скрипта UIControl привиден в листинге 1.

Листинг 1 UIControl

```
using UnityEngine.UI;
using UnityEngine;
using UnityEngine.SceneManagement;

public class UIControl : MonoBehaviour {

    // Use this for initialization
    [SerializeField] private GameObject menu;
    public GameObject resumeButton;
    public Text HealthField;
    public Text ScoreField;
    public Text gameOverText;
    private bool isMenuOpen;
    private GameController gc;
    [SerializeField] private Player player;

      void Start () {
        isMenuOpen = false;
        if(menu!= null)
        {
            menu.SetActive(false);
        }
        if (HealthField != null)
        {
            HealthField.text = "HEALTH: " + player.GetHealth().ToString();
        }
        if (ScoreField != null)
        {
            ScoreField.text = "SCORE: " + player.GetPoints().ToString();
        }
        if (gameOverText != null)
        {
            gameOverText.gameObject.SetActive(false);
        }
        gc = FindObjectOfType<GameController>();
    }
```

```
    // Update is called once per frame
      void Update () {
        if (HealthField != null)
        {
            HealthField.text = "HEALTH: " + player.GetHealth().ToString();
        }
        if (ScoreField != null)
        {
            ScoreField.text = "SCORE: " + player.GetPoints().ToString();

        }

    }


    public void OnOpenMenu()
    {

        if (menu != null)
        {
            if (player.GetHealth() <= 0)
            {
                if ((resumeButton != null) && (player.GetHealth() <= 0))
                    resumeButton.SetActive(false);
                if (gameOverText != null)
                    gameOverText.gameObject.SetActive(true);
            }
            isMenuOpen = !isMenuOpen;
            menu.SetActive(isMenuOpen);
            GameController.SetPause(isMenuOpen);
           // player.gameObject.SetActive(!isMenuOpen);
        }

    }

    public void OnRestartLevel()
    {
        GameController.SetPause(false);
        player.SetIsRestart(true);
        SceneManager.LoadScene(SceneManager.GetActiveScene().name,
LoadSceneMode.Single);
    }

    public void OnExitToMainMenu()
    {
        Destroy(gc);
        SceneManager.LoadScene("GameMenu", LoadSceneMode.Single);
    }

    public void OnExitFromGame()
    {
        Application.Quit();
    }

    public void OnStartGame()
    {
        SceneManager.LoadScene("Level1", LoadSceneMode.Single);
    }
      }
```

Персонаж Игрока

Игрок представляет собой игровой объект, который предоставляется для управления. Он состоит из 3D модели, коллайдера(Collider), компонента Rigidbody, наличие которого подразумевает, что этот объект подчиняется законам физики. Помимо этого, данный игровой объект содержит в себе два поведенческих скрипта Player и PlayerMove.

Скрипт Player хранит в себе значение очков здоровья и количество набранных очков в течении уровня очков. Во время выполнения процедуры Start, которая вызывается после появления объекта на сцене, мы обращаемся к объекту GameController, который хранит общее количество очков и присваиваем это значение полю levelPoints, а также поле isInvisible, отвечающее за неуязвимость игрока становится false. Процедуры changeHealth и changeScore отвечают соответственно за изменение количества здоровья и очков. Во время выполнения change-Health, если мы хотим отнять у игрока здоровье, то проверяем является ли он бессмертным(isInvisible) и неравенство количество отнимаемого здоровье 100, и в этом случае прерываем выполнение процедуры. Далее, если мы прибавляем игроку здоровье и сумма имеющегося и прибавляемого здоровья больше 100, то мы также прерываем выполнение. После этого, мы производим изменения очков здоровья и если после этого изменения очков меньше или равно 0, то вызываем процедуру OnO-penMenu компонента UIControl, поскольку количество здоровья меньше или равно 0 является условием поражения. Листинг 2 отображает программный код скрипта Player.

Листинг 2. Player

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

public class Player: MonoBehaviour {

    // Use this for initialization
    public bool isInvisible;
    public bool isRestart;
    [SerializeField] private int health=100;
    [SerializeField] private int levelPoints=0;
    [SerializeField] private float invisTimeInSec = 5.0f;
```

```
    [SerializeField] private UIControl control;

      void Start () {
        isInvisible = false;
        levelPoints = GameController.points;
    }

      // Update is called once per frame
      void Update () {

      }

    public void changeHealth(int value)
    {
        if ((value < 0) && isInvisible && (value!=-100))
            return;
        if (health + value > 100)
            return;
        health += value;
        if (health <= 0)
        {
            if (control != null)
            {
                control.OnOpenMenu();
            }
        }
    }

    public void SetIsRestart(bool value)
    {
        isRestart = value;
    }
    public int GetHealth()
    {
        return health;
    }

    public int GetPoints()
    {
        return levelPoints;
    }
    public void changePoints(int value)
    {
        levelPoints += value;
    }
    public IEnumerator MakeInvisible()
    {
        isInvisible = true;
        yield return new WaitForSeconds(invisTimeInSec);
        isInvisible = false;
    }

    void OnDestroy()
    {
        if (isRestart)
            return;
        GameController.points = levelPoints;
    }
}
```

Скрипт PlayerMove отвечает за управлением персонажем игрока и его анимацией.В процедуре Start мы получаем ссылку на компоненты типа

Animator, который управляет выполнением анимации 3D-моделью и CapsuleCollider, позволяющий взаимодействовать с другими объектами, которые обладают компонентом категории Collider. Процедура Update выполняется каждый цикл выполнения игрового положения. Во время её выполнения, мы сначала получаем значения движения вертикальной оси. В зависимости от результата, мы меняем значение переменных аниматора vertimotion и down, которые отвечают за клипы анимации вертикального движения модели персонажа игрока. Затем, при помощи функции translate если мы производим движение игрового персонажа по оси Y, который равен значению ввода на вертикальной оси, помноженной на вертикальную скорость, значение которой мы можем задавать через редактор. Аналогично поступаем с горизонтальной осью. С горизонтальной анимацией движения связаны переменные аниматора Horimotion и Reverseside [8]. В отличии от вертикальной оси, для горизонтального движения необходимо поворачивать персонажа вокруг башни. Для этого мы используем процедуру RotateAround, которая позволяет двигать объекты вокруг определенной точки, с указанным направлением и углом. На рисунке 7 представлена схема схема анимации персонажа игрока. В листинге 3 содержится код скрипта PlayerMove.

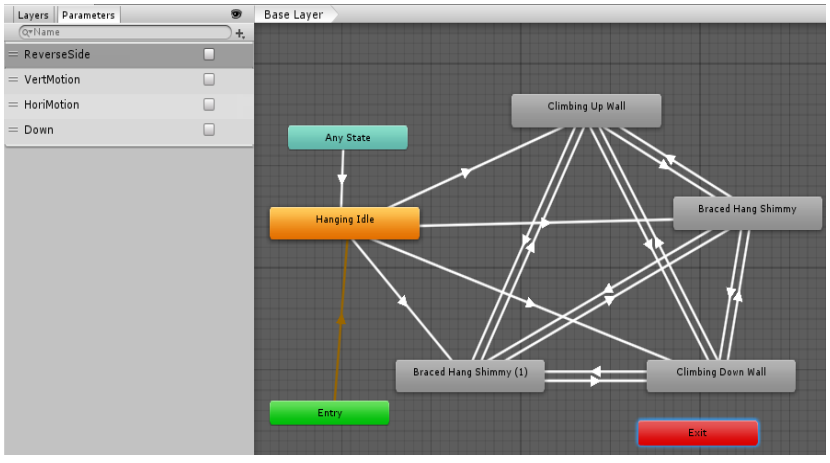


Рис. 7. Схема анимации персонажа игрока

Листинг 3. PlayerMove

```
using System.Collections;
```

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

public class PlayerMove: MonoBehaviour {

    // Use this for initialization
    [SerializeField] private float vertSpeed = 0.1f;
    [SerializeField] private float horiSpeed = 0.1f;
    [SerializeField] private Vector3 Rotatepoint = new Vector3(0f,0f,0f);
    [SerializeField] private float angle = 1f;
    private Animator animator;
    private CapsuleCollider coll;

    void Start () {
        animator = GetComponent<Animator>();
        coll = GetComponent<CapsuleCollider>();
      }

      // Update is called once per frame
      void Update () {
        float yAxis = Input.GetAxis("Vertical");
        if (yAxis != 0)
        {
            animator.SetBool("VertMotion", true);
            if (yAxis>0)
            {
                animator.SetBool("Down", false);
            }
            else
            {
                animator.SetBool("Down", true);
            }
        }
        else
        {
            animator.SetBool("VertMotion", false);
        }
        transform.Translate(0, yAxis*vertSpeed, 0);

        float xAxis = Input.GetAxis("Horizontal");
        if (xAxis != 0)
        {
            animator.SetBool("HoriMotion", true);
            if (xAxis>0)
            {
                animator.SetBool("ReverseSide", false);
            }
            else
            {
                animator.SetBool("ReverseSide", true);
            }
        }
        else
        {
            animator.SetBool("HoriMotion", false);
        }

        if ((yAxis == 0) && (xAxis == 0))
        {
            coll.center = Vector3.MoveTowards(coll.center, new Vector3(0,
1.44f, 0), 0.1f);
        }
        else
```

```
        {
            if(yAxis == 0)
            {
                coll.center = Vector3.MoveTowards(coll.center, new Vec-
tor3(0, 2.0f, 0), 0.1f);
            }
            else if (yAxis>0)
            {
                coll.center = Vector3.MoveTowards(coll.center, new Vec-
tor3(0, 1.70f, 0), 0.1f);
            }
            else
            {
                coll.center = Vector3.MoveTowards(coll.center, new Vec-
tor3(0, 2.85f, 0), 0.1f);
            }
        }
        transform.RotateAround(Rotatepoint, new Vector3(0f, -
horiSpeed*xAxis, 0f), angle);
    }
}
```

Прочие объекты

В категорию препятствий и бонусов входит окно с гоблином, сам булыжник, различные виды бонусов, трава, прикосновение к которой мгновенно убивает игрока и балкон с принцессой. Скрипт WindowFrame, код которого приведен в листинге 4, отвечает за поведение окна с гоблином. Во момент выполнения процедуры Start мы начинаем сопроцесс(Courutine), который выполняет появление новых булыжников из окна. В каждое выполнение Update мы выполняем переключение отображаемой в данный момент текстуры. При значении true переменной isGobl мы используем текстуру с изображением гоблина в окне. Процедура Shoot выполняется всё время, пока на сцене существует объект, к которому привязан данный скрипт. Он создает новый объект, указанный в переменной projectile на расстоянии, которое берется из поля offset. Также, именно в ней происходит переключения состояния переменной isGobl. Конструкция yield return new WaitForSeconds отвечает за паузу в выполнении сопроцесса на указанное количество секунд.

Листинг 4. WindowFrame

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

public class WindowFrame : MonoBehaviour {
```

```
    // Use this for initialization
    [SerializeField] private GameObject projectile;
    [SerializeField] private Vector3 offset = new Vector3(0,-0.5f,0.5f);


    [SerializeField] private Texture frameWithoutGobl;
    [SerializeField] private Texture frameGobl;
    private Renderer rend;
    private bool isGobl =false;

    void Start () {
        StartCoroutine(Shoot());
        rend = GetComponent<Renderer>();
    }

      // Update is called once per frame
      void Update () {
        if(isGobl)
        {
            rend.material.mainTexture = frameGobl;
        }
        else
        {
            rend.material.mainTexture = frameWithoutGobl;
        }

    }

    private IEnumerator Shoot()
    {
        if (projectile != null)
        {
            while (true)
            {
                isGobl = false;
                yield return new WaitForSeconds(1.5f);
                Instantiate(projectile, new Vec-
tor3(transform.position.x+offset.x,transform.position.y + off-
set.y,transform.position.z + offset.z),Quaternion.identity);

                isGobl = true;
                yield return new WaitForSeconds(1.5f);
            }
        }
    }}
```

Скрипт Balcony прикрепляется к объекту, который является балконом с принцессой и выполняет переход на сцену, указанную в nextSceneName. В момент столкновения с другим игровым объектом, который содержит компонент Collider, происходит проверка на то, содержит ли столкнувшийся объект скрипт Player. Если это так, то, при наличии заполненного названия следующей сцены, происходит переход на неё. Листинг 5 содержит код скрипта Balcony.

Листинг 5. Balcony

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

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class BalCony : MonoBehaviour
{

    // Use this for initialization
    [SerializeField] private string nextSceneName;

    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnTriggerEnter(Collider other)
    {
        Player player = other.GetComponent<Player>();
        if (player != null)
        {
            if (nextSceneName != null)
                SceneManager.LoadScene(nextSceneName, LoadScene-
Mode.Single);
        }
    }
}
```

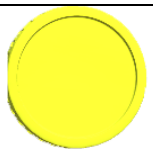
Скрипт BonusCollect используется для объектов, которы являются бонусами. Тип эффекта определяется выбором варианта в редакторе для поля effect. При столкновении с игроком, в зависимости от типа бонуса, происходит вызов разных процедур у объекта-персонажа игрока. Если тип бонуса Score, то мы меняем количество очков на указанное в поле points. Тип Invisibility, то мы делаем игрока временно неуязвимым вызвав у него процедуру MakeInvisible. При типе бонуса Life игроку добавляется здоровье из поля life. В таблице 1 представлены имеющиеся в игре виды бонусов. В листинге 6 приведен программный код скрипта BonusCollect.

Табл. 1. Игровые бонусы

№	Изображение	Очки	Получаемое здоровье	Описание
1		100	10	Pomegranate восстанавливает игроку 10

				единиц здоровья и дает 100 очков

Продолжение табл. 1

№	Изображение	Очки	Получаемое здоровье	Описание
2		200	0	Coin дает игроку 200 очков
3		0	0	Star делает игрока бессмертным на 5 секунд (кроме смертельной травы)

Листинг 6. BonusCollect

```
using UnityEngine;

public class BonusCollect : MonoBehaviour {


    public enum EffectType {Score = 1,
                            Invisibility = 2,
                            Life = 3
                           }
    [SerializeField] private EffectType effect= EffectType.Score;
    [SerializeField] private int points = 100;
    [SerializeField] private int life = 10;
    // Use this for initialization
    void Start () {

      }

      // Update is called once per frame
      void Update () {

      }

    private void OnTriggerEnter(Collider other)
    {
        Player player = other.GetComponent<Player>();
        if (player != null)
        {
            switch (effect)
            {
                case EffectType.Score:
                    Debug.Log("score");
                    player.changePoints(points);
                    break;
                case EffectType.Invisibility:
                    Debug.Log("Invisiability");
```

```
                    player.MakeInvisible();
                    break;
                case EffectType.Life:
                    Debug.Log("Life");
                    player.changePoints(points);
                    player.changeHealth(life);
                    break;
            }
            Destroy(gameObject);
        }
    }
}
```

Скрипт DeadlyGrass отвечает за работу ползущей снизу травы, которая мгновенно убивает игрока при соприкосновении с ним. Также, если происходит столкновение с окном или булыжником, то эти объекты уничтожаются, чтобы освободить используемую ими память. Код этого скрипта отражен в листинге 7.

Листинг 7. DeadlyGrass

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

public class DeadlyGrass : MonoBehaviour {

      // Use this for initialization
    [SerializeField] float vertSpeed = 1.0f;
      void Start () {

      }

      // Update is called once per frame
      void Update () {
        transform.Translate(0, vertSpeed, 0);
    }
    void OnTriggerEnter(Collider other)
    {
        Player player = other.GetComponent<Player>();
        WindowFrame wf = other.GetComponent<WindowFrame>();
        Pebel pb = other.GetComponent<Pebel>();
        if (player != null)
        {
            player.changeHealth(-100);
            vertSpeed = 0;
        }
        if(wf != null)
        {
            Destroy(wf.gameObject);
        }
        if (pb != null)
        {
            Destroy(pb.gameObject);
        }
    }
}
```

Скрипт Pebel отвечает за поведение объектов-булыжников. Процедура Vanish является сопроцессом и выполняет уничтожение объекта, к которому прикреплен скрипт после указанного времени. В момент столкновения с другим объектом, мы проверяем у него наличие скрипта Player, и, если это игрок, отнимаем здоровье в количестве, указанном в поле damage и вызывает уничтожение через три секунды, чтобы визуально булыжник успел отскочить от персонажа игрока. В листинге 8 приведен программный код скрипта Pebel.

Листинг 8 Pebel

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

public class Pebel : MonoBehaviour {

    // Use this for initialization
    [SerializeField] private int damage=10;
    [SerializeField] private float lifeTimeOverall = 10.0f;
    [SerializeField] private float lifeTimeAfterCollide = 3.0f;
      void Start () {
        StartCoroutine(Vanish(lifeTimeOverall));
    }

      // Update is called once per frame
      void Update () {

      }

    private void OnCollisionEnter(Collision collision)
    {
        Player player = collision.gameObject.GetComponent<Player>();
        if (player != null)
        {
            player.changeHealth(-damage);
            StartCoroutine(Vanish(lifeTimeAfterCollide));
        }
    }

    private IEnumerator Vanish(float sec = 3.0f)
    {
        yield return new WaitForSeconds(sec);
        Destroy(gameObject);
    }
}
```

4. ТЕСТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ

4.1. Функциональное тестирование

Тестирование приложения производилось проверкой работы всех функций системы. Данный вид тестирования можно назвать функциональным тестированием. Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает [10]. Результаты тестирования представлены в таблице 2.

Табл. 2. Функциональное тестирование

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
1	Отображение главного меню игры	На экране отображено главное меню игры с кнопками «Start game» и «Exit from game»	На экране отобразилось главное меню игры с кнопками «Start game» и «Exit from game»	Да
2	Работа кнопки «Start game»	При нажатии кнопки «Start game» открывается сцена «level1»	После нажатия кнопки «Start game» открывается сцена «level1»	Да
3	Работа кнопки «Exit from game»	При нажатии кнопки «Exit from game» игровое приложение заканчивает свою работу	После нажатия кнопки «Exit from game» игровое приложение заканчивает свою работу	Да

Продолжение табл. 2.

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
4	Перемещение игрока	Нажатие клавиш-стрелок персонаж игрока начинает двигаться в указанном направлении.	Нажатие клавиш-стрелок персонаж игрока начинает двигаться в указанном направлении.	Да
5	Подбор бонуса здоровья	Количество здоровья и очков увеличивается, при количестве здоровья равном 100 увеличения не происходит	Количество здоровья и очков увеличивается, при количестве здоровья равном 100 увеличения не происходит	Да
6	Подбор бонуса очков	При подборе бонуса очков количество очков увеличивается	При подборе бонуса очков количество очков увеличивается	Да
7	Подбор бонуса неуязвимости	При подборе бонуса неуязвимости, на 5 секунд булыжники не наносят урон персонажу игрока	При подборе бонуса неуязвимости, на 5 секунд булыжники не наносят урон персонажу игрока	Да
8	Подбор любого бонуса	После столкновения с персонажем игрока объект удаляется	После столкновения с персонажем игрока объект удаляется	Да

Продолжение табл. 2.

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
9	Работа окошка с гоблином	Окошки с гоблинами в заданный интервал времени создают новые булыжники, меняется текстура окошка	Окошки с гоблинами в заданный интервал времени создают новые булыжники, меняется текстура окошка	Да
10	Нанесение урона булыжниками	При столкновении булыжника с игроком, игроку наносится урон.	При столкновении булыжника с игроком, игроку наносится урон.	Да
11	Падение булыжников	Булыжники падают в течение 10 секунд, а затем уничтожаются	Булыжники падают в течение 10 секунд, а затем уничтожаются	Да
12	Перемещение травы	Терновник передвигается вертикально вверх с заданной скоростью	Терновник передвигается вертикально вверх с заданной скоростью	Да
13	Смерть игрока	При достижении количества здоровья игрока уровня равного нулю. Открывается внутриигровое	При достижении количества здоровья игрока уровня равного нулю. Открывается внутриигровое	Да

		меню, отображается надпись «Game over».	меню, отображается надпись «Game over».	

Продолжение табл. 2.

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
14	Столкновение игрока со смертельной травой	При столкновении игрока со смертельной травой происходит незамедлительная смерть игрока	При столкновении игрока со смертельной травой происходит незамедлительная смерть игрока	Да
15	Вызов внутриигрового меню кнопкой в верхнем правом углу	При нажатии кнопки в верхнем правом углу, игровой процесс приостанавливается, отображается внутриигровое меню	При нажатии кнопки в верхнем правом углу, игровой процесс приостанавливается, отображается внутриигровое меню	Да
16	Нажатие клавиши «Re-start level»	Нажатие клавиши «Restart level» приводит к повторной загрузке текущего уровня.	Нажатие клавиши «Restart level» приводит к повторной загрузке текущего уровня.	Да
17	Прикосновение с балконом с принцессой	Прикосновение с балконом с принцессой приводит к вызову	Прикосновение с балконом с принцессой приводит к вызову	Да

		следующего уровня, при этом набранные на уровне очки сохраняются	следующего уровня, при этом набранные на уровне очки сохраняются	

Продолжение табл.2.

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
18	Столкновение булыжников и окон с гоблинами с смертельной травой	Столкновение со смертельной травой приводит к уничтожению объектов-булыжников и объектов-окон	Столкновение со смертельной травой приводит к уничтожению объектов-булыжников и объектов-окон	Да
19	Прохождение первого уровня	Первый уровень проходим- на уровне есть балкон с принцессой, и игрок может до него добраться. При достижении происходит переход на второй уровень	Первый уровень проходим- на уровне есть балкон с принцессой, и игрок может до него добраться. При достижении происходит переход на второй уровень	Да
20	Прохождение второго уровня	Первый уровень проходим- на уровне есть балкон с принцессой, и игрок может до него добраться. При достижении происходит переход на третий уровень	Первый уровень проходим- на уровне есть балкон с принцессой, и игрок может до него добраться. При достижении происходит переход на третий уровень	Да

Окончание табл.2

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
21	Экран поздравления	Кнопки экрана поздравления работают, отображается текст поздравления с количеством набранных игроком очков	Кнопки экрана поздравления работают, отображается текст поздравления с количеством набранных игроком очков	Да
22	Анимация персонажей	Принцесса имеет зацикленную анимацию танца, персонаж игрока имеет анимации, зависящие от направления движения	Принцесса имеет зацикленную анимацию танца, персонаж игрока имеет анимации, зависящие от направления движения	Да

4.2. Юзабилити-тестирование

Юзабилити-тестирование – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий [4].

Все задачи были решены всеми респондентами. Выявленные проблемы устранены. Отчет о результатах юзабилити-тестирования приведен в приложении 2.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы бакалавра мною было реализовано игровое приложение с непрямым управлением в жанре тактическая стратегия и решены следующие задачи.

1. Проведено исследование компьютерных игр схожего жанра, изучена разработка игр на движке Unity.
2. Спроектировано игровое приложение.
3. Реализовано игровое приложение с помощью игрового движка Unity.
4. Проведено тестирование игрового приложения.

СПИСОК ЛИТЕРАТУРА

1. Албахари Д., Албахари Б. C# 6.0. Справочник. Полное описание языка. – М.: Вильямс, 2017. – 1040 с.

2. Буч Г., Рамбо Д., Якобсон А. Введение в UML от создателей языка. – М.: ДМК Пресс, 2015. – 496 с.

3. Исследование игровой индустрии от Mail.ru Group. [Электронный ресурс] URL: https://gamestats.mail.ru/article/rossijskij_igrovoj_rynok_v_2016_godu_56_7_mlrd_rublej/ (дата обращения: 10.04.2018).

4. Майерс Г., Баджетт Т., Сандлер К. Искусство тестирования программ. – М.: Вильямс, 2016. – 495 с.

5. Мэннинг Д., Батфилд-Эддисон П. Unity для разработчика. Мобильные мультиплафор-менные игры. – СПб.: Питер, 2018. – 352 с.

6. Официальный сайт Unity. [Электронный ресурс] URL: https://unity3d.com (дата обращения: 28.04.2018).

7. Официальный сайт игры Upwards, lonesome robot. [Электронный ресурс] URL: https://www.kasedogames.com/upwards-lr (дата обращения: 14.04.2018)

8. Пэрент Р. Компьютерная анимация. Теория и алгоритмы. – М.: КУДИЦ-ОБРАЗ, 2004. – 560 с.

9. Страница игры Super Kong Climb в AppStore. [Электронный ресурс] URL: https://itunes.apple.com/ru/app/super-kong-climb-endless-pixel-arcade-climbing-game/id948714582?mt=8 (дата обращения: 14.04.2018)

10. Тамре Л. Введение в тестирование программного обеспечения. – М.: Вильямс, 2003. – 368 с.

11. Торн А. Искусство создания сценариев в Unity. – М.: ДМК-Пресс, 2016. – 360 с.

12. Троэлсэн Э., Джепикс Ф. Язык программирования C# 6.0 и платформа .NET 4.6. – М.: Вильямс, 2016. – 1440 с.

13.Фаулер М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. – СПб.: Символ плюс, 2018. – 192 с.

14.Федерация компьютерного спорта России. [Электронный ресурс] URL: https://resf.ru/about/resf/ (дата обращения: 30.05.2018).

15.Хокинг Д. Unity в Действии. Мультиплатформенная разработка на C#. – СПб.: Питер, 2017. – 336 с.

ПРИЛОЖЕНИЯ

Приложение 1

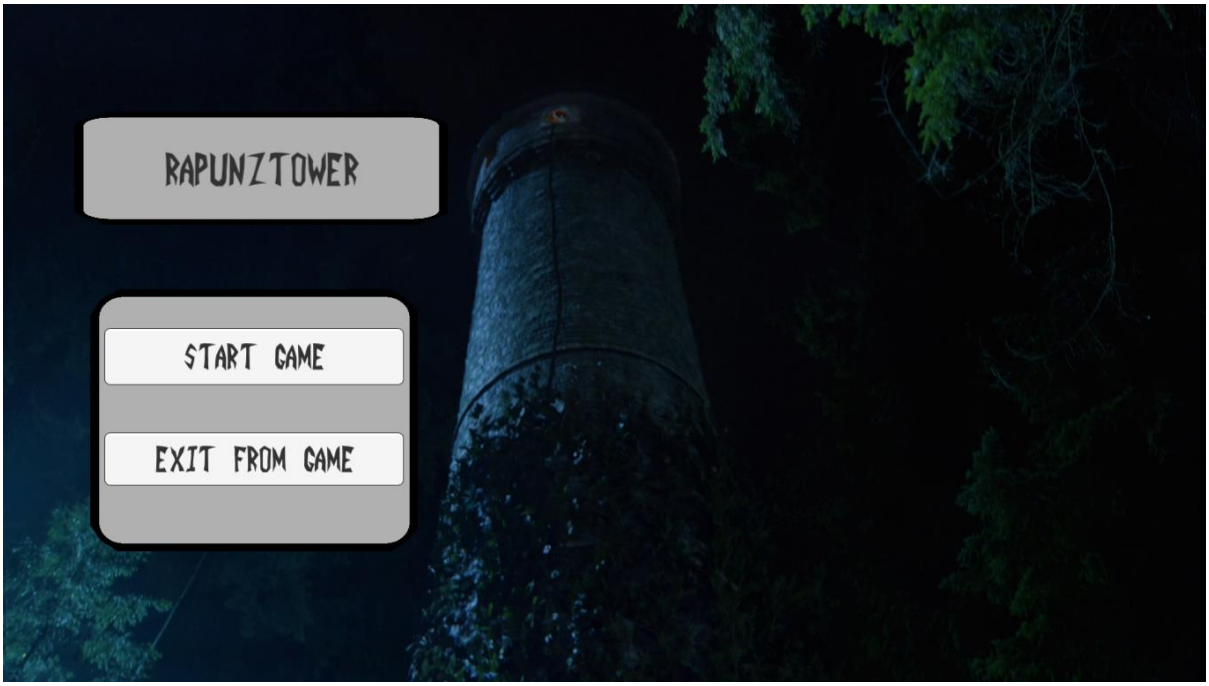


Рис. 1. Главное меню

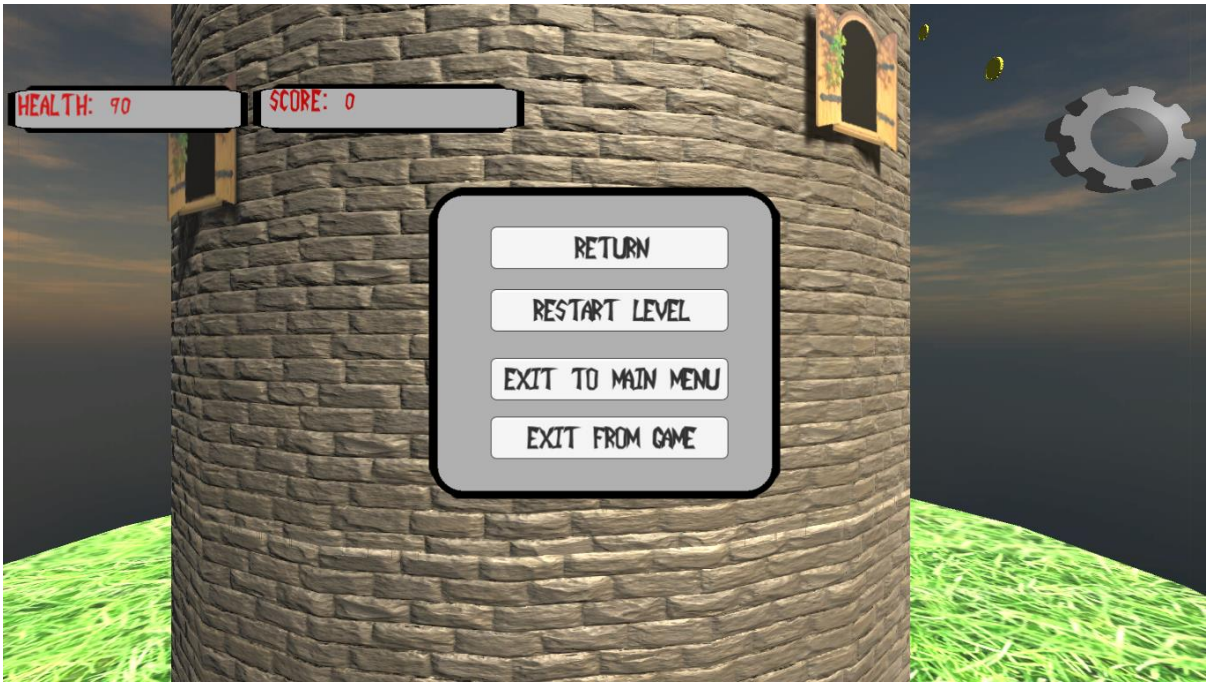


Рис. 2. Внутриигровое меню 1

Рис. 3. Игровой уровень

Рис. 4. Игровой уровень

Рис. 5. Игровой уровень

Рис. 6 Экран поздравления

Приложение 2

Отчет о юзабилити-тестировании

1. Объект исследования

Объектом исследования является игровое приложение RapunzTower.

2. Метод исследования

В качестве метода исследования был выбран метод юзабилити-тестирования «Мысли вслух».

3. План проведения тестирования

1. Планирование:
   1.1. Разработка заданий.
   1.2. Набор участников тестирования.
2. Проведение тестирования.
3. Анализ полученных данных.

4. Методика проведения тестирования

Для проведения исследования использовалось следующее оборудование и ПО:

1. ПК с операционной системой Windows не младше 7;
2. Клавиатура и компьютерная мышь;
3. Встроенный в ПК микрофон.

Наблюдатель выдает список заданий участникам тестирования. Респонденты выполняют их последовательно и самостоятельно, при этом высказывая свои мысли и мнения, возникающие в процессе взаимодействия с приложением. Комментарии записываются через микрофон и затем представляются модератором в виде протокола тестирования.

5. Протокол заданий

1. Отправная точка: главное меню.
   Задание: начать новую игру.

2. Отправная точка: игровая сцена «Уровень 1».

   Задание: собрать «Health Bonus».

3. Отправная точка: игровая сцена «Уровень 1».

   Задание: накопить 300 «Очков».

4. Отправная точка: игровая сцена «Уровень 2».

   Задание: собрать «Score Bonus».

5. Отправная точка: внутриигровое меню.

   Задание: начать уровень заново.

6. Отправная точка: игровая сцена «level1».

   Задание: пройти первый уровень.

7. Отправная точка: игровая сцена «level1».

   Задание: проиграть.

8. Отправная точка: внутриигровое меню.

   Задание: выйти из игры.

9. Отправная точка: игровая сцена «level1»

   Задание: пройти игру

6. Респонденты

1. Веретенников Артур Альбертович, студент ЮУрГУ.
2. Ковалев Вячеслав Сергеевич, студент ЮУрГУ.
3. Елисеев Роман Сергеевич, студент ЮУрГУ.

7. Результаты тестирования

7.1. Метрики

Табл. 1. Результаты юзабилити-тестирования

Респонденты	**Доля выполненных задач**	**Доля правильно выполненных задач**	**Время на выполнение задач**	**Кол-во ошибок**
Веретенников Артур Альбертович	100%	100%	8 мин.	0
Чулкевич Роман Андреевич	100%	100%	7 мин.	0

Елисеев Роман Сергеевич	100%	100%	10 мин.	3

7.2. Выявленные недостатки

Игровой процесс

Каждый следующий уровень короче предыдущего.

Дизайн

Окно с гоблином нарисовано слишком примитивно.