Региональный конкурс исследовательских и проектных работ школьников
«Высший пилотаж - Пенза»
Computer science (информационные технологии, в рамках конференции «Авангард)
Муниципальное бюджетное общеобразовательное учреждение
средняя общеобразовательная школа с. Бессоновка

Проект по информатике
на тему:
«Разработка компьютерной игры на платформе Unity»

Проект ученика 9 «В» класса
Кальченко Никиты Сергеевича
Руководитель проекта: Атаманова Татьяна Ивановна, учитель информатики

с.Бессоновка
2022 г.

Оглавление

ВВЕДЕНИЕ 3
1. ПОНЯТИЕ «КОМПЬЮТЕРНАЯ ИГРА» . ТЕХНОЛОГИИ РАЗРАБОТКИ КОМПЬЮТЕРНЫХ ИГР 4
1.1. Понятие и классификация компьютерных игр 4
1.2. Алгоритм реализации проекта 4
1.3. Средства создания компьютерных игр 5
2. РАЗРАБОТКА ПРОЕКТА 7
2.1 Постановка задачи проекта 7
2.2.1 Актуальность, цель и назначение 7
2.1.2 Суть проекта 7
2.1.3 Сценарий 7
2.2 Реализация проекта 7
2.2 1 Этап разработки игры 7
2.2.3 Тестирование 10
2.3.4 Технические характеристики 10
ЗАКЛЮЧЕНИЕ 11
СПИСОК ИСПОЛЬЗОВАННЫХ РЕСУРСОВ 12
Приложение 1 13

ВВЕДЕНИЕ

Рынок компьютерных игр в России постоянно растёт, но тем не менее развит достаточно слабо, по сравнению с этим сегментом в других странах. При высоком уровне спроса (по данным статистики по состоянию на конец 2022 года в нашей стране более 60 % населения играют в различные компьютерные игры, и это число имеет устойчивую тенденцию к росту, особенно в части выбирающих 3D-игры), при этом у нас мало компаний-разработчиков, которые могли бы конкурировать с зарубежными.

В связи с вышесказанным развитие технологий в указанном направлении представляется одним из наиболее перспективных и актуальных, особенно в нашей стране.

Объектом исследования в моей работе является процесс разработки компьютерных игр.

Предмет исследования: технологии разработки компьютерной 3D игры.

Цель проекта – разработать прототип компьютерной 3D игры средствами среды Unity.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить особенности спроса на рынке компьютерных игр в России;
- выбрать жанр, вид и платформу для собственной компьютерной игры;
- разработать сценарий, концепцию основных элементов;
- выбрать и изучить средство реализации;
- подготовить необходимые для игры анимации;
- реализовать прототип игры.

1. ПОНЯТИЕ «КОМПЬЮТЕРНАЯ ИГРА» . ТЕХНОЛОГИИ РАЗРАБОТКИ КОМПЬЮТЕРНЫХ ИГР

1.1. Понятие и классификация компьютерных игр

Компьютерная игра – компьютерная программа, которая служит для организации игрового процесса[1]. Компьютерные игры могут быть созданы по сюжетам известных фильмов или книг, но в возможны и обратные случаи, когда по известной серии игр начинают создавать дополнительные материалы, расширяющие вселенную игры. Кроме всего прочего специально разработанные игры могут выступать как обучающий ресурс или создают условия для использования в научно - исследовательских целях.

Таким образом, компьютерные игры плотно влились в современную жизнь и поле их применения постоянно расширяется.

Компьютерные игры классифицируют по нескольким основным признакам[2]:

- жанр;
- количество игроков;
- визуальное представление;
- платформа.

По количеству игроков игры разделяются на два вида:

- однопользовательские;
- многопользовательские.

По визуальному представлению компьютерные игры можно разделить на следующие виды:

- текстовые – минимальное графическое представление, общение с игроком проходит с помощью текста;
- 2D – все элементы отрисованы в виде двумерной графики.
- 3D – все элементы отрисованы в виде трехмерной графики.

По типу платформы:

- персональные компьютеры;
- игровые приставки/консоли;
- мобильные телефоны.

Игра может соответствовать сразу нескольким жанрам, выйти сразу на нескольких платформах или иметь как однопользовательский режим игры, так и многопользовательский. Данной классификации достаточно, для того что бы определить большинство компьютерных игр существующих на данный момент.

Основываясь на данной классификации компьютерных игр я принял решение разработать трехмерную однопользовательскую игру для персональных компьютеров. Данное решение было принято потому, что многопользовательские игры сложны в реализации.

1.2. Алгоритм реализации проекта

Алгоритм разработки любой компьютерной игры включает в себя три важных этапа:

- концептирование и проектирование;

[1] Компьютерная игра — Википедия (wikipedia.org)
[2] Классификация компьютерных игр — Википедия (wikipedia.org)

— собственно разработка;

— релиз и поддержка.

Средствами разработки являются программный код и игровой движок. От их продуманного выбора зависит как скорость создания продукта, так и его работоспособность. Программный код зависит от платформы, для которой будет создаваться компьютерная игра. Для персонального компьютера, оптимальным выбором является язык программирования C#.

Игровой движок отвечает за описание физических характеристик объектов игры, особенностей рендеринга графики и т.д.. При выборе игрового движка я руководствовался такими составляющими как доступность и язык программирования. Игровой движок Unity позволяет разрабатывать игры на языке C# и при этом является бесплатным.

После концептирования и проектирования, начинается второй этап реализации проекта – разработка. Во-первых, нужно определиться с сюжетом и игровой механикой – набором правил и способов, реализующим определённым образом некоторую часть интерактивного взаимодействия игрока и игры[3]. Игровая механика основывается на цели игры. Параллельно с разработкой игровой механики идет написание сюжета игры и ранняя проработка графической составляющей и дизайна.

Затем начинается самая важная часть – разработка самой игры. После создания основ идет сборка первого прототипа игры. Прототип – это быстрая, черновая реализация интерактивной системы для упрощения исследования самой игры или поведения пользователя в ней. Если первая версия игры успешно проходит тестирование, наступает следующий этап разработки – проработка механики и объектов игры, где исправляются первые ошибки и неисправности в коде игры, которые были обнаружены при тестировании.

После этого наступает этап создания второго прототипа игры. Его тестирование. Далее третий прототип, снова тестирование, и так далее. Если игра проходит все тесты, идет сборка финальной версии игры и релиз. После релиза игры происходит последующая её поддержка – выпуск патчей (файлов исправлений ошибок в готовом продукте).

1.3. Средства создания компьютерных игр

Обзор игровых движков

Основными средствами разработки игр являются игровые движки и графические редакторы для отрисовки графики. В настоящее время существует огромное количество различных игровых движков. Основные их различия заключаются в поддерживаемых языках программирования и функциональности. Для анализа выбрал следующие игровые движки: Unreal Engine и Unity самые популярные платформы для разработки трехмерных игр.

Unreal Engine[4] – игровой движок разрабатываемый и поддерживаемый компанией Epic Games. Написанный на языке C++, движок позволяет создавать игры для большинства операционных систем и платформ. Поддерживает различные системы рендеринга, средства голосового воспроизведения текста, распознавание речи, модули для работы с сетью и поддержки различных устройств ввода.

Unity[5] – средство для разработки двух- и трехмерных игр, являющееся одним из наиболее популярных на сегодняшний день систем разработки. Позволяет создавать приложения,

[3] Игровая механика — Википедия (wikipedia.org)
[4] Unreal Engine — Википедия (wikipedia.org)
[5] Unity (игровой движок) — Википедия (wikipedia.org)

работающие под большинством современными операционными системами, а также на игровых приставках и Motion Parallax 3D дисплеях (устройства для воспроизведения виртуальных голограмм), например, Nettlebox. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL. Последние версии Unity позволяют создавать приложения для шлема виртуальной реальности.

Редактор Unity имеет простой интерфейс, который легко настраивать. Движок поддерживает два сценарных языка: C#, JavaScript. Расчёты физики производит физический движок PhysX от NVIDIA.

Проект в Unity делится на сцены (уровни) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Unity распространяется бесплатно, но помимо бесплатной, существуют четыре сборки - стандартная Unity, Unity iOS Pro, Android Pro и командная лицензия. Они отличаются стоимостью и функциональностью. Бесплатная версия имеет некоторые ограничения, однако возможность распространять игры имеется. С выходом Unity 5 многие ограничения бесплатной версии были убраны.

Все это делает Unity одним из наиболее приоритетных движков для начинающих игровых разработчиков.

2. РАЗРАБОТКА ПРОЕКТА

2.1 Постановка задачи проекта

2.2.1 Актуальность, цель и назначение

На сегодняшний день рынок компьютерных игр развивается очень быстро, количество активных игроков растет с каждым годом, как и прибыль с продаж. Это очень актуальное направление. Стоит отметить, что общество устало от излишне масштабных проектов, требующими от игрока глубокого понимая механики игрового процесса, долгого никания в правила и тонкости, к тому же прохождение таких игр требует больших временных затрат. На этом фоне игра с простым и понятным геймплеем будет смотреться предпочтительнее для тех, кому занятость или лень не позволяет увлечься монументальными проектами..

В свете сказанного, формулируется цель проекта: предоставить людям средство для приятного времяпрепровождения. Назначение проекта заключается в развлечении потенциального потребителя, снятия стресса.

2.1.2 Суть проекта

Проект представляет собой компьютерную 3D игру, основная цель которой заключается в развлечении и предоставлении средства для отдыха и приятного времяпрепровождения.

Прежде всего, проект должен отвечать следующим требованиям:

1. Простой и понятный геймплей.

2. Простое управление главным героем.

3. Враги обитают в определенных местах и, при обнаружении игрока, они должны сражаться с ним, до тех, пор пока игрок не сбежит или не будет повержен, или не победит.

4. За победу будет начисляться игровая валюта.

5. При проигрыше игрока, должно выводиться меню, предлагающее потратить игровую валюту, либо выйти в главное меню, либо повторить игру.

6. Меню не должно быть перегружено, но в тоже самое время должно

быть информативным(см. Рис. 5).

2.1.3 Сценарий

Перед началом разработки проекта, всегда пишется сценарий игры, представляющий собой краткий синопсис сюжета с описанием основных персонажей. На его основе разрабатываются первые концепты персонажей.

Главный герой моей игры – субстанция души на пороге новой реинкарнации. Она имеет облик мужчины, но это не важно, главное – смысл существования. Задача души – возродиться в новом мире, предварительно преодолев все преграды и проблемы в старом. Мужчина проходит через дверь, и видит что стоит на том же месте, но монстры прошлого, так и не побежденные им в прошлой карме, возродились. С этого и начинается игра. Название очень символично «Next Door», что значит «следующая дверь» (Рис. 4).

2.2 Реализация проекта

2.2 1 Этап разработки игры

Для реализации компьютерной игры были выбраны игровой движок Unity и Microsoft Visual Studio 2017 для описания программного кода. При запуске Unity мы увидим окно проекта (см. Рис.1), посередине находятся окно Сцены, окно Анимации, а так же окно Игры. Первое служит для создания общей композиции уровня и добавления новых объектов, второе представляет возможность для анимирования объектов, третье окно представляет вид из камеры показывая, каким образом будет выглядеть игра на данный момент.

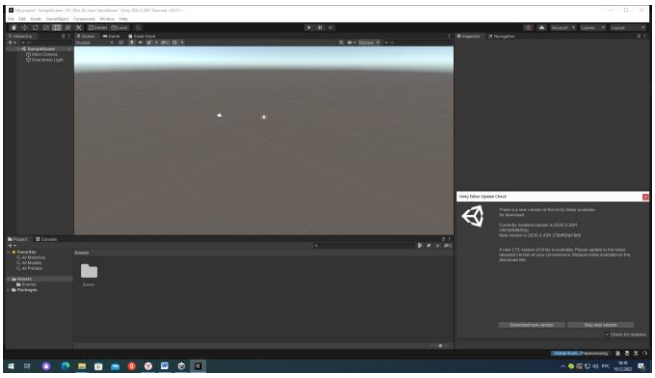

Рис. 1

Слева находится окно Иерархии, здесь показываются все объекты, участвующие в данной сцене. С самого начала здесь находится только камера. Справа окно Инспектора, здесь отображаются все текущие свойства выбранного объекта. Снизу находятся три окна: окно Проекта, окно Аниматора и Консоль. В окне Проекта отображаются все объекты, добавленные в текущую игру, в том числе скрипты, анимации и прочее. В окне Аниматора создаются связи и правила перехода между различными анимациями. Консоль служит для отображения ошибок и исключений возникающих во время работы игры.

Необходимо добавить в сцену платформу, по которой будет перемещаться герой и его враги. Без неё игрок просто будет бесконечно падать в пространстве. Сначала нам необходимо добавить сам спрайт платформы, который был создан ранее, для этого достаточно просто перетащить его из папки в окно проекта, после чего он должен появиться среди всех объектов уже добавленных в игру. После этого нам необходимо добавить нашу платформу в текущую сцену. Это можно сделать 2-мя способами: перетащить из окна Проекта в окно Сцены, либо сначала в окне Иерархии создать отдельный пустой объект, к которому впоследствии и применить спрайт платформы.

После этого нам необходимо добавить к объекту коллайдер. Он необходим для того, что бы движок понимал границы объекта и игрок мог с ним взаимодействовать. В Unity существует два вида коллайдеров: для трехмерных и двумерных объектов. Так как создаваемая игра трехмерная, то и коллайдер нужно добавлять трехмерный. Что бы добавить коллайдер к объекту, необходимо выделить его в окне Иерархии, а затем добавить компонент коллайдера в окне Инспектора. Для настройки размеров коллайдера нужно воспользоваться кнопкой Edit Collider.

Теперь создаем героя игры. Для этого так же добавляем трехмерный объект, и добавляем к нему спрайт героя (см. Рис. 2). Для реализации героя нам понадобится коллайдер, а так же компонент Rigidbody 3D, через него настраиваться физическая модель объектов. Так как главный герой должен двигаться и совершать различные действия нам необходимо создать скрипт на языке C#. Это можно сделать несколькими способами: создать в окне Проекта, или создать сразу на необходимом объекте в окне Инспектора.

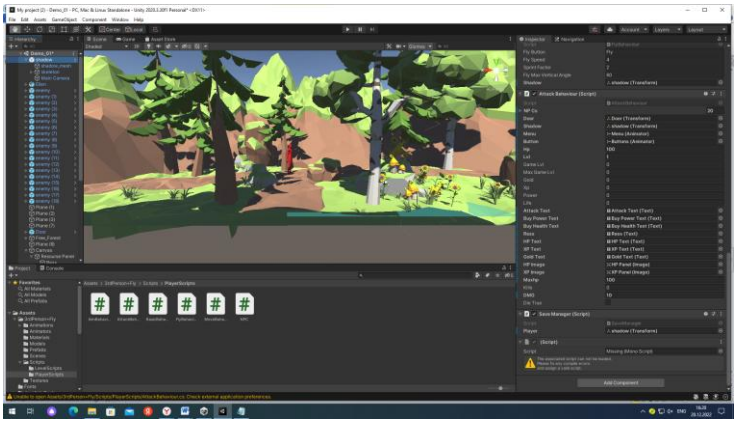

Рис. 2

Когда скрипт будет создан, открываем его и после этого должен запуститься Microsoft Visual Studio. В коде уже будут подключены основные библиотеки, а так же создан стандартный метод Update. Записываем код для движения персонажем (см. Приложение 1). После чего добавляем его к нашему герою.

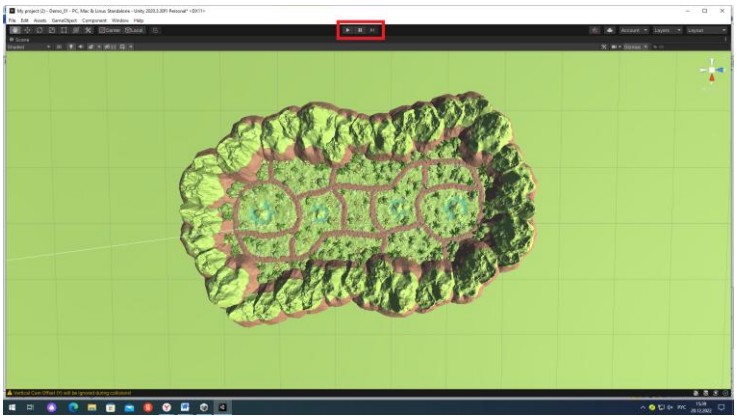

Рис. 3

Теперь нам необходимо создать врагов. Враги создаются схожим способом, за тем исключением, что игрок ими управлять не может. Так же отличительно особенностью врагов является то, что в зависимости от ситуации, они сами выбирают необходимое действие доступное им по правилам игры.

Затем создаем Animator Controller и добавляем к нашему персонажу. Он определяет правила и связи для анимаций определённого объекта. Для создания самих анимаций, нужно в окне Animation создать новый клип анимации, перед этим выбрав нужный объект, а затем перенести необходимую анимацию в данное окно в привальном порядке. Здесь настраиваем скорость анимации.

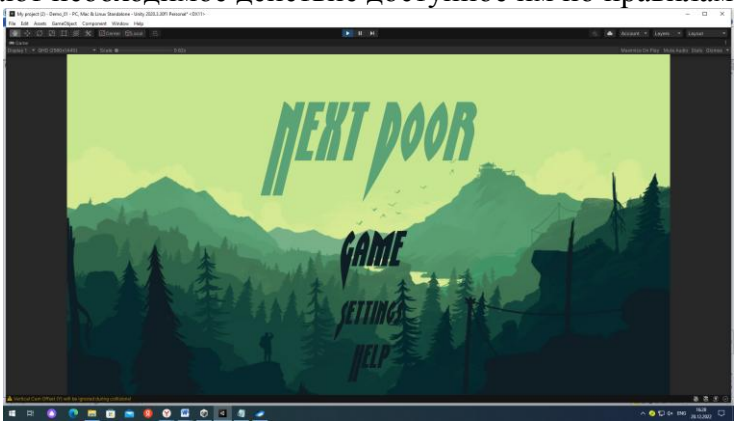

Рис. 4

После создания всех необходимых клипов анимаций, переходим в окно Аниматора. Здесь нам необходимо создать связи и настроить условия переходов между различными анимациями, что мы уже создали ранее. Для врагов делаем тоже самое. Если запустить уровень, мы сможем побегать в разные стороны, попрыгать, и совершать остальные доступные игроку действия.

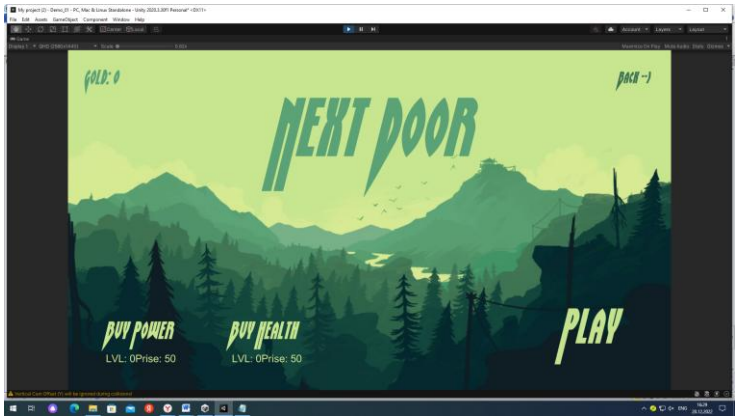

Рис. 5

Теперь нам необходимо создать уровень, после чего расставить на нем различные объекты и врагов. Первым делом нужно прикинуть примерный план уровня, а так же расположение врагов и объектов. После того как общий план уровня будет достаточно проработан, можно приступать к реализации самого уровня. Уровень можно собрать прямо в окне сцены Unity.

Проверить работоспособность проекта можно прямо в окне Unity, для этого необходимо нажать кнопку запуска проекта в верхней части экрана. Рядом с ней находятся кнопки паузы и

окончания теста игры. Они имеют такой же вид, как кнопки управления в любом другом аудио- или видеоплеере. (см. Рис. 3)

2.2.3 Тестирование

Я думаю, что на данном этапе реализации проекта проводить какое-либо крупное тестирование нет смысла, так как в игре реализована лишь малая толика возможностей, что предполагалось реализовать. Именно по этой причине данное тестирование, является лишь небольшой проверкой на соответствие основным функциональным требованиям.

Это тестирование показало, что на данном этапе реализации проекта, игра соответствует большинству функциональных требований выделенных на этапе постановки задачи. Но все же стоит уделить особое внимание программному коду и исправить ошибку, из-за которой ситуация проигрыша игрока отрабатывается неверно.

2.3.4 Технические характеристики

Мой проект является прототипом, работащим на уровне ранней альфа-версии. Вызвано это было тем, что создание качественного продукта занимает большое количество времени. Так, к примеру, разработка небольшой игры для мобильных телефонов с процедурно генерируемым уровнем может занимать более года, при условии, что над проектом работает не более трех человек. Увеличение количества человек может ускорить разработку, но не сильно, а я создавал проект в одиночку.

На данный момент сложно определить какими системными требованиями будет обладать проект в будущем, так как многие элементы я планирую усовершенствовать. На данный момент мы имеем лишь элементы, отвечающие за основную игровую механику, а значит, в будущем технические характеристики для разрабатываемой компьютерной игры могут очень сильно измениться. Сейчас можно с уверенностью утверждать лишь то, что для запуска данного прототипа потребуется компьютер, обладающий системными характеристиками не ниже минимально допустимых для корректной работы игрового движка Unity, для которого наиболее важным аспектом является только видеокарта. Это означает, что в данный момент созданный прототип с большой вероятностью будет работать на большинстве компьютеров.

ЗАКЛЮЧЕНИЕ

Основываясь на всей полученной в ходе исследования информации, было решено разработать прототип трехмерной игры для одного игрока на игровом движке Unity. Такое решение было принято по нескольким причинам:

— однопользовательская игра, в отличие от многопользовательская легче в создании;
— игровой движок Unity распространяется бесплатно и позволяет разрабатывать приложения на языке программирования C#.

После выбора средств разработки было начато изучение Unity, а так же разработка самого проекта. В ходе разработки был изучен игровой движок Unity и были приобретены необходимые знания и умения, а именно: создание сцен, создание анимаций, создание и написание скриптов, настройка объектов, создание UI, компиляция проекта.

В ходе реализации проекта были выполнены следующие задачи:

— изучены особенности и состояние рынка компьютерных игр России;
— выбраны жанр, вид и платформа для компьютерной игры;
— разработаны сценарий и концепция основных элементов;
— выбрано и изучено средство реализации;
— подготовлены необходимые для игры анимации;
— реализован прототип игры.

Освоение среды разработки Unity несет не маловажный характер, так как в этой среде возможно создание и более сложных игр, которые могут составить конкуренцию популярным игровым сериям.

СПИСОК ИСПОЛЬЗОВАННЫХ РЕСУРСОВ

1) Википедия: https://ru.wikipedia.org/wiki
2) Официальный сайт Unity: **https://unity.com/ru**
3) Официальный сайт Миксамо: https://www.mixamo.com/#/
4) Среда разработки игр Unity

Приложение 1

Программный код, отвечающий за работу главного персонажа

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class AttackBehaviour : MonoBehaviour
{
    public Transform[] NPCs;
    public Transform Door;
    public Transform Shadow;
    public Animator Menu;
    public Animator Button;
    public float hp;
    public int lvl;
    public int GameLvl;
    public int MaxGameLvl;
    public float gold;
    public float xp;
    public int power;
    public int life;
    public Text attackText;
    public Text BuyPowerText;
    public Text BuyHealthText;
    public Text Ress;
    public Text HPText;
    public Text XPText;
    public Text GoldText;
    public Image HPImage;
    public Image XPImage;
    public float maxhp;
    public int kills;
    public float DMG;
    public bool dieTrue;
    private bool attackTrue;
    private bool keyQPress;
    private int randomAttack;
    private Animator CharacterController;

    void Start()
    {
        attackTrue = false;
```

```
    keyQPress = false;
    randomAttack = 0;
    CharacterController = GetComponent<Animator>();
    Shadow = GetComponent<Transform>();
    Button.SetBool("Enable", false);
    StartCoroutine(AttackCoroutine());
}
void Update()
{
    BuyPowerText.text = "LVL: " + power + "Prise: " + (50 * Mathf.Pow(1.5f, power));
    BuyHealthText.text = "LVL: " + life + "Prise: " + (50 * Mathf.Pow(1.5f, life));
    HPImage.fillAmount = hp / maxhp;
    XPImage.fillAmount = xp / (Mathf.Pow(1.5f, lvl) * 60f);
    if (Input.GetKey(KeyCode.Q))
    {
        if (!keyQPress)
        {
            attackTrue = !attackTrue;
            keyQPress = true;
        }
    }
    else
    {
        keyQPress = false;
    }
    Attack();
    if (hp > 0)
    {
        CharacterController.SetBool("Die", false);
    }
    if (hp <= 0)
    {
        Shadow.GetComponent<Rigidbody>().useGravity = false;
        Shadow.GetComponent<CapsuleCollider>().enabled = false;
        die();
        Menu.SetBool("Enable", true);
        Button.SetBool("Enable", false);
    }
    if (kills >= 10)
    {
        Door.GetComponent<DoorManager>().OpenDoor();
        kills = 0;
    }
    if (xp >= Mathf.Pow(1.5f, lvl) * 60f)
    {
        float ratio = hp / maxhp;
```

```
        lvl++;
        maxhp = Mathf.Pow(1.2f, lvl) * Mathf.Pow(1.2f, life) * 100f;
        hp = maxhp * ratio;
        DMG = Mathf.Pow(1.2f, lvl) * 10f;
        xp = 0;
    }
    Ress.text = "  LVL: " + lvl + "   Gold: " + gold;
    XPText.text = "XP: " + xp + "/" + (Mathf.Pow(1.5f, lvl) * 60f);
    HPText.text = "HP: " + hp + "/" + maxhp;
    GoldText.text = "Gold: " + gold;
}
public void LoadGame()
{
}
public void OnGravity()
{
    Shadow.GetComponent<Rigidbody>().useGravity = true;
    Shadow.GetComponent<CapsuleCollider>().enabled = true;
    Debug.Log("popal restore: hp=");
}
public void Buttons()
{
    if (Shadow.GetComponent<BasicBehaviour>().mobile)
        Button.SetBool("Enable", true);
}
public void AttackForUIButton()
{
    if (!keyQPress)
    {
        attackTrue = !attackTrue;
        keyQPress = true;
    }
    else
    {
        keyQPress = false;
    }
}
public void LVLUpNPC()
{
    GameLvl++;
    foreach (Transform code in NPCs)
        code.GetComponent<NPC>().LVLUp();
}
public void Borning()
{
    if (hp <= 0)
```

```
    {
        if (MaxGameLvl < GameLvl)
            MaxGameLvl = GameLvl;
        lvl = 0;
        maxhp = 100 * Mathf.Pow(1.2f, life);
        hp = maxhp;
        xp = 0;
        kills = 0;
        DMG = 10 * Mathf.Pow(1.2f, power);
        GameLvl = MaxGameLvl - 10;
        if (GameLvl < 0)
            GameLvl = 0;
        foreach (Transform code in NPCs)
            code.GetComponent<NPC>().LVLUp();
    }
}
public void LifeUp()
{
    if (gold >= 50 * Mathf.Pow(1.5f, life))
    {
        life++;
        gold -= 50 * Mathf.Pow(1.5f, life - 1);
    }
}
public void PowerUp()
{
    if (gold >= 50 * Mathf.Pow(1.5f, power))
    {
        power++;
        gold -= 50 * Mathf.Pow(1.5f, power - 1);
    }
}
public void TakeAweyHealth(float damage)
{
    if (hp > 0)
    {
        hp -= damage;
    }
    damage = 0;
}
public void die()
{
    CharacterController.SetBool("Die", true);
    transform.position += new Vector3(0, 4, 0);
}
public void Damage()
```

```
    {
        foreach (Transform code in NPCs)
            code.GetComponent<NPC>().TakeAweyHealth(DMG);
    }
    private void Attack()
    {
        if (attackTrue)
        {
            if (randomAttack == 0)
                randomAttack = -1;
            attackText.text = "Attack: V";
        }
        else
        {
            randomAttack = 0;
            CharacterController.SetInteger("Attack", 0);
            attackText.text = "Attack: X";
        }
    }
    IEnumerator AttackCoroutine()
    {
        while (true)
        {
            if (randomAttack != 0 && hp > 0)
            {
                randomAttack = Random.Range(1, 4);
                CharacterController.SetInteger("Attack", randomAttack);
                yield return new WaitForSeconds(1);
            }
            else
            {
                CharacterController.SetInteger("Attack", 0);
                yield return new WaitForSeconds(0.1f);
            }
        }
    }
}
```

Рецензия
на практико-ориентированный исследовательский проект

«Разработка компьютерной игры на платформе Unity» ученика 9 класса МБОУ СОШ с. Бессоновка Бессоновского района Пензенской области Кальченко Никиты

Общая оценка работы:

Практико-ориентированная исследовательская работа выполнена на тему «Разработка компьютерной игры на платформе Unity». Тема выбрана не случайно, ведь спрос на 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, что такое компьютерная игра и её прототип, какие классификации компьютерных игр существуют, в чём заключаются основные моменты и правила создания компьютерных игр. Теоретическая часть соответствует выбранной теме.

Практическая часть описана логично, подробно. Представлены необходимые пояснения и скриншоты. В приложении отражён основной программный код для главного героя игры, остальные коды не приводились в связи с их масштабностью. 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