|  |
| --- |
| Распоряжение Правительства Российской Федерации от 27 декабря 2010 г. №2387-р  Федеральное государственное бюджетное образовательное учреждение высшего образования  **«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**  КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ |

**К У Р С О В О Й П Р О Е К Т**

**Тема: Проектирование и разработка мобильной игры “Межвременные войны”**

**Специальность: 09.02.07 «Информационные системы и программирование»**

**Исполнитель: студент группы 31ИС-19 Асадов Л. И.**

**Руководитель: Наумочкин А. С.**

**Москва 2021**

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc90644979)

[1. Аналитический раздел 6](#_Toc90644980)

[1.1 Предпроектное обследование разработки игрового приложения 6](#_Toc90644981)

[1.1.1 Выбор лидирующих жанров в игровой индустрии 6](#_Toc90644982)

[1.1.2 Платформер 6](#_Toc90644983)

[1.1.3 Шутер 7](#_Toc90644984)

[1.1.4 Battle Royale 8](#_Toc90644985)

[1.2 Выбор языка программирования 8](#_Toc90644986)

[1.2.1 C# 9](#_Toc90644987)

[1.2.2 C++ 9](#_Toc90644988)

[1.2.3 Java 10](#_Toc90644989)

[1.3 Выбор IDE 11](#_Toc90644990)

[1.3.1 Visual Studio 11](#_Toc90644991)

[1.3.2 Eclipse 12](#_Toc90644992)

[1.3.3 Visual Studio Code 13](#_Toc90644993)

[2. Проектный раздел 18](#_Toc90644994)

[2.1 Разработка алгоритмов 18](#_Toc90644995)

[2.1.1 Передвижение 18](#_Toc90644996)

[2.1.2 Стрельба 19](#_Toc90644997)

[2.1.3 Прыжок 20](#_Toc90644998)

[2.2 Разработка программного кода для программного продукта 20](#_Toc90644999)

[2.3 Разработка интерфейса пользователя 27](#_Toc90645000)

[Заключение 29](#_Toc90645001)

[Список использованных источников 30](#_Toc90645002)

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ПК – Персональный компьютер;

ПО – Программное обеспечение;

ОС – Операционная система;

Шутер – англ. Shooter – “стрелялка”;

Платформер – англ. Platform;

Королевская битва – англ. Battle Royale.

Спрайт – англ. Sprite – Модель;

Геймплей – англ. Gameplay – компонент игры, отвечающий за интерактивное взаимодействие игры и игрока;

Дизайн – англ. Design – визуальное оформление уровней, окружающей игрока среды;

ИИ - игровой интеллект – англ. “Artificial Intelligence” (AI);

Visual Studio – VS;

IDE – Integrated Drive Electronic;

Баг – англ. Bug – Ошибка/Дефект в программе.

ВВЕДЕНИЕ

Изобретение компьютеров послужило переломным моментом в развитии многих отраслей. Для игровой индустрии, в которой раньше присутствовали только игровые автоматы. Данное новшество стало прорывом, затронувшее большое количество людей. Игровая индустрия почти с самого начала имела большую популярность в мире. Компьютерная игра, которая дала старт индустрии, называлась SpaceWar, благодаря ей она стала именно такой, какой она сейчас и является.

Игровая индустрия оставила след во многих других сферах:

* киноиндустрия;
* были созданы специальные встречи для фанатов игр.

Пользователей привлекали проекты, которые были созданы на основе фильмов, хотя для разработчиков это не всегда окупалось. Плохие финансовые сборы были связаны с тем, что большую часть фанатов заинтересовало в фильме то, чего в игре не было.

Людям нравилось, и нравится до сих пор, проводить время в компьютерных играх. Для кого-то это является ещё одним способом приятно провести время, для кого-то это было помощью в трудные времена жизни, а для кого-то это способ получения заработной платы.

Целью курсового проекта является выбор языка программирования, установка и настройка сопутствующего программного обеспечения для разработки игрового приложения для “Межвременные войны”.

Так же одной из целей является отрисовка графических объектов, их детализации и общего вида на фоне основных событий. Так же, наиважнейшим аспектом игрового приложения является геймплей, ведь порой, несмотря даже на неподобающую графическую составляющую, пользователей завлекает геймплей, а точнее его выделение среди остальных, менее интересных, завлекающих своим сюжетом и продуманных игровых приложений.

1. **Аналитический раздел**

# **Предпроектное обследование разработки игрового приложения**

Предпроектное обследование игрового приложения является первой стадией, с которой начинается формирование новой или усовершенствование имеющегося игрового приложения.

Стадия включает в себя:

* анализ лидирующих жанров компьютерных игровых приложений на рынке;
* анализ отличий между 2D- и 3D- компьютерными игровыми приложениями.

По итогам предпроектного обследования определяются и утверждаются основные параметры реализации проекта: постановка задачи, метод реализации проекта.

* + 1. **Выбор лидирующих жанров в игровой индустрии**

Жанры в игровой индустрии – это классификация игровых приложений, разделяющая игры по их визуальной, сюжетной, а также механической части.

Анализируя лидирующие жанры компьютерных игровых приложений были выявлены три подходящих жанра:

* Платформер;
* Шутер;
* Королевская битва.
  + 1. **Платформер**

Чаще всего жанр представляется в 2D графике (рисунок 1), нежели в 3D и подразумевает под собой прыжки по платформам, лазанье по лестницам.



Рисунок 1 – Пример платформера

Многие компьютерные игровые приложения подобного жанра характеризуются не реалистичностью, рисованной мультяшной графикой.

Достаточно часто, помимо жанра Платформер, игровому компьютерному приложению назначают жанр Инди–игра. Проекты с подобным критерием означают независимость разработчиков. Проект был создан либо одиноким разработчиком, либо коллективом, не имеющего финансовой поддержки издателя игровых приложений.

* + 1. **Шутер**

Жанр игр, который может представляться в любой графике, как 2D, так и в 3D. (рисунок 2).



Рисунок 2 – Пример Шутера

Представляет из себя стрельбу, зачастую приходится стрелять по игровому искусственному интеллекту, либо по неподвижным мишеням, что в нынешнее время редко встречается.

* + 1. **Battle Royale**

Набравший свой пик популярности за 3 года жанр, подразумевающий собой выживание среди большого количества людей (от 64 до 100 человек) при помощи оружия, с целью остаться одним единственным выжившим на больших масштабах территории, высадившись на него, используя любые для этого способы (рисунок 3).



Рисунок 3 – Пример высадки в Battle Royale

Жанр многогранен, так как, с ним могут сочетаться большинство остальных жанров, например, Шутер, Платформер, Строительство.

По итогам анализа представленных жанров игр было решено использовать для разработки игрового приложения только 2 жанра – Платформер и Шутер в 2D-графике.

# **Выбор языка программирования**

Для создания более детализированного игрового приложения следует пользоваться наиболее комфортным и быстрым языком программирования.

Под подобные требования подходят сразу 3, похожие между собой, языка программирования: C#, C++, Java. Проанализировав каждый из них будет выявлен наиболее подходящий язык программирования для разработки игрового приложения.

* + 1. **C#**

Является языком программирования, относящийся к семье с С–подобным синтаксисом (C++, Java) (рисунок 4).

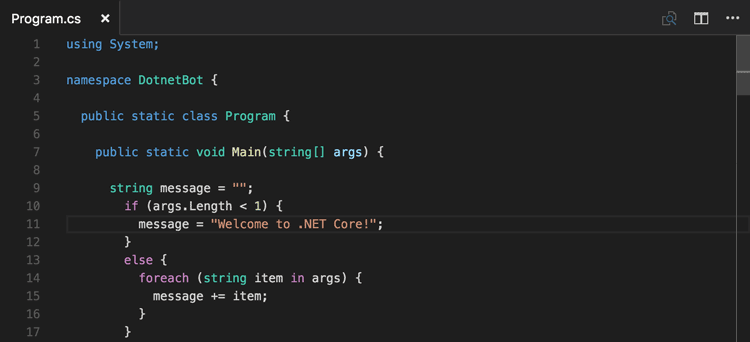


Рисунок 4 – Пример кода на C#

Получил многие функции от своих предшественников, что делает его современнее, а значит и более удобнее для программирования, учитывая все недостатки его старших собратьев.

* + 1. **C++**

В семействе является одним из самых популярных языков программирования, поскольку широко используется для разработки программного обеспечения (рисунок 5).

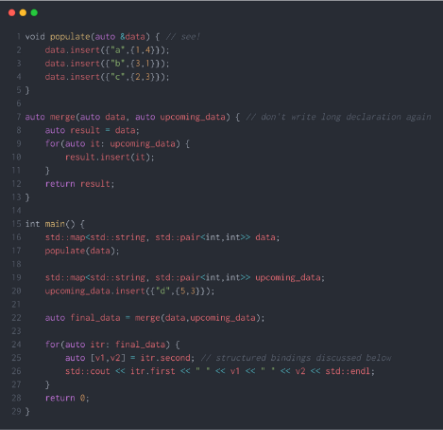


Рисунок 5 – Пример кода на C++

C++ настолько многофункционален, что область его применения включает в себя: создание ОС, разнообразных прикладных программ, драйверов устройств и так далее.

Даже учитывая свою многофункциональность, язык программирования не позволяет достичь нужных целей в “пару шагов” и является недостаточно удобным для написания больших программ, от чего требуется проверять каждую строку, для выявления проблемы.

* + 1. **Java**

Служит строго типизированным языком. Позволяет выполнять Java–приложения на любом устройстве, для которого существует виртуальная машина, что делает язык программирования особенным, относительно других.

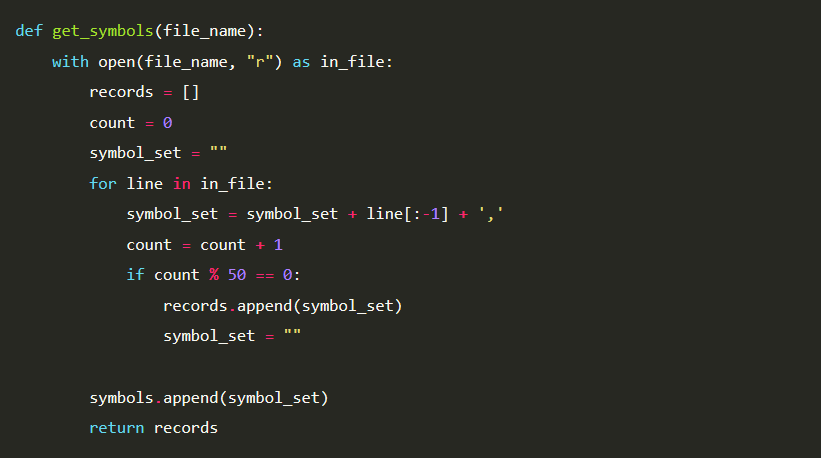


Рисунок 6 – Пример кода на Java

Так же особенностью является гибкая система безопасности, исполнение программы полностью контролируется виртуальной машиной, допустим, если будет совершаться попытка несанкционированного доступа к данным, то установленные пользователем полномочия в программе тут же вызовут немедленное ее прерывание (рисунок 6). Ниже, в виде таблицы представлено сравнение трех языков программирования для выбора одного из них (Таблица 1).

Таблица 1 - Сравнение языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики** | **C#** | **C++** | **Java** |
| Удобство в написании | + | – | + |
| Простота в освоении | + | – | – |
| Понятность кода | + | + | + |

Проанализировав каждый, выбранный язык программирования C#, в связи с его удобством в написании кодов и современности, относительно других членов семейства.

# **Выбор IDE**

Среда разработки – это комплекс программных средств для разработки программного обеспечения.

Учитывая выбор языка программирования следует выбрать и среду разработки компьютерного игрового приложения. На рынке для подобных требований имеется огромный выбор. Перед выбором следует проанализировать каждую из сред разработок для полного понимания, что из себя представляют выбранные из них.

* + 1. **Visual Studio**

С Visual Studio многие начинали работать и до сих пор работают в нем. Связано это с тем, что он является самой удобной средой разработки среди остальных, поскольку данный способ создания программного обеспечения был создан Microsoft (рисунок 7).

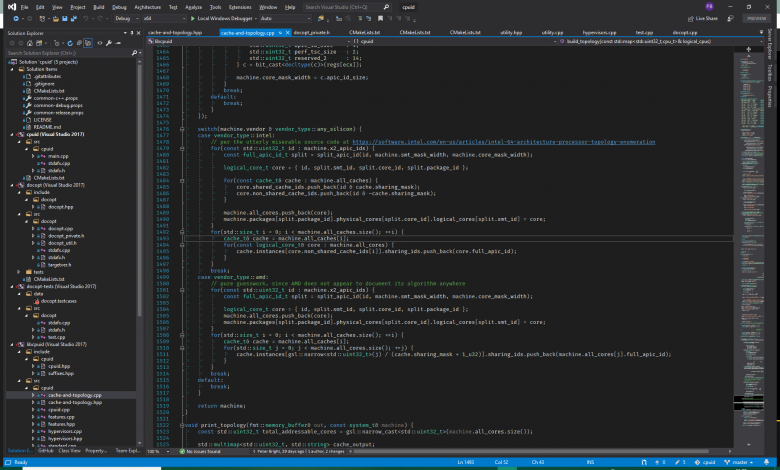


Рисунок 7 – Интерфейс и пример кода в Visual Studio

Официальная, бесплатная, многофункциональная, поддерживает платформы .NET, имеет широкие возможности по разработке приложений для Windows. Так же огромным преимуществом для VS является удобный и простой интерфейс.

* + 1. **Eclipse**

Является одной из самых популярных мультиязычных сред, которая ориентирована на разработку Java-приложений, также позволяет работать с кодами на C# (рисунок 8).

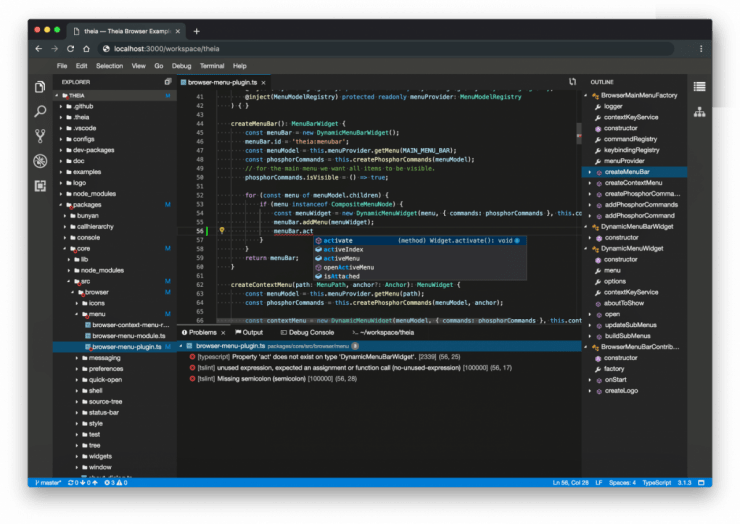


Рисунок 8 – Интерфейс и пример кода в Eclipse

Для Eclipse имеются множества плагинов способные помочь, благодаря тем же плагинам имеется возможность для визуальной, и не только, кастомизации. Данная среда так же является бесплатной, что позволяет пользоваться ей любому желающему, но при этом сложная для начинающих пользователей и не дает гарантий надежности, так как плагины создаются сообществом Eclipse.

* + 1. **Visual Studio Code**

Представляется бесплатным, легковесным, кроссплатформенным редактором кода. Для доведения его до статуса IDE пользователи используют множества плагинов, что позволяет воспользоваться программой в полной мере (рисунок 9).

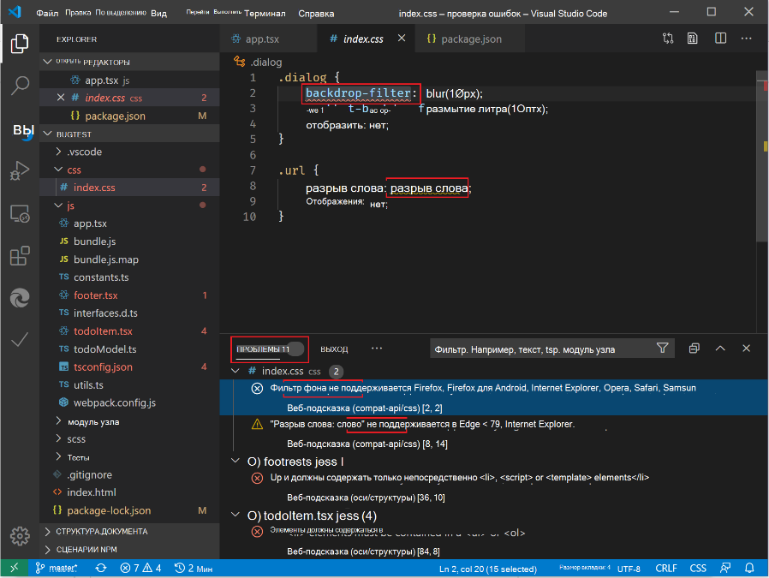


Рисунок 9 – Интерфейс и пример кода с ошибками Visual Studio Code

С учетом низкой функциональности и сомнительной надежности (многие надстройки не всегда выполняют даже основные функции), редко бывают доведены до максимальной эффективности в сфере разработки.

Ниже приведена таблица сравнения сред разработки игрового приложения (таблица 2).

Таблица 2 - Сравнение сред разработки

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики** | **Visual Studio** | **Eclipse** | **Visual Studio Code** |
| Удобство пользования | + | + | – |
| Интерфейс | + | – | + |
| Быстродействие | + | – | – |
| Обучаемость | – | + | + |
| Надежность | + | + | – |

Анализируя каждый из подходящий для разработки IDE был выявлен наиболее подходящий из них – Visual Studio

* 1. **Выбор игрового движка**

Игровым движком называется базовое программное обеспечение для компьютерной игры. Простыми словами, является основным ядром игры, на основе которого строятся остальные составляющие части. Затрагивает все компоненты игры, начиная от физики в ней и заканчивая сетевыми аспектами. При выборе игровых движков стоит учитывать тот огромный мировой ассортимент, который доступен для любого пользователя сети Интернет и не требует финансовых затрат. Наиболее подходящие движки под нынешние условия являются:

* Unity;
* Unreal Engine;
* Godot.
  + 1. **Unity**

Значится самым популярным игровым движком на сегодняшний день. На этом движке было создано огромное количество межплатформенных игровых приложений (рисунок 10).

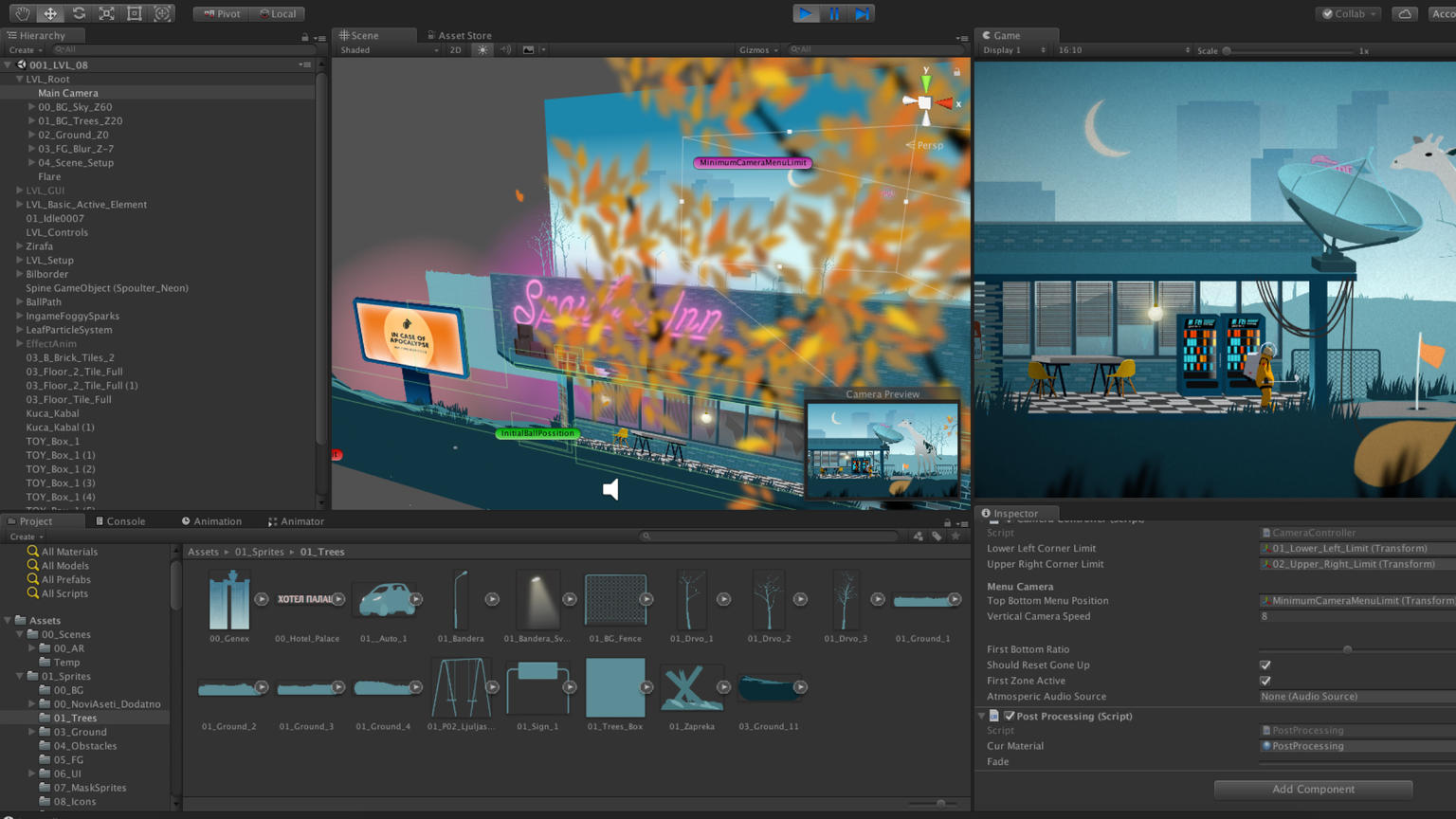


Рисунок 10 – Интерфейс и пример работы на Unity

Движок обладает кроссплатформенностью, является бесплатным для пользования физических лиц, выбор готовых шаблонов больше, чем у остальных рассматриваемых движков вместе взятых. К тому же обучаемость для работы на движке не вызывает трудностей – на просторе Интернета множество обучающих материалов, а также имеет большие возможности как в 2D, так и в 3D.

* + 1. **Unreal Engine**

Unreal Engine является движком, повлиявшим на всю игровую индустрию шутеров своей детализацией в разработке игровых приложений. Используя данный движок следует учесть, что игровое приложение будет написано на языке программирования C++, хоть этого и можно избежать благодаря особенностям данной платформы.

Движок предлагает отличную графику, хорошие инструменты для моделирования уровней, большое количество источников для обучения работы с ним (рисунок 11).

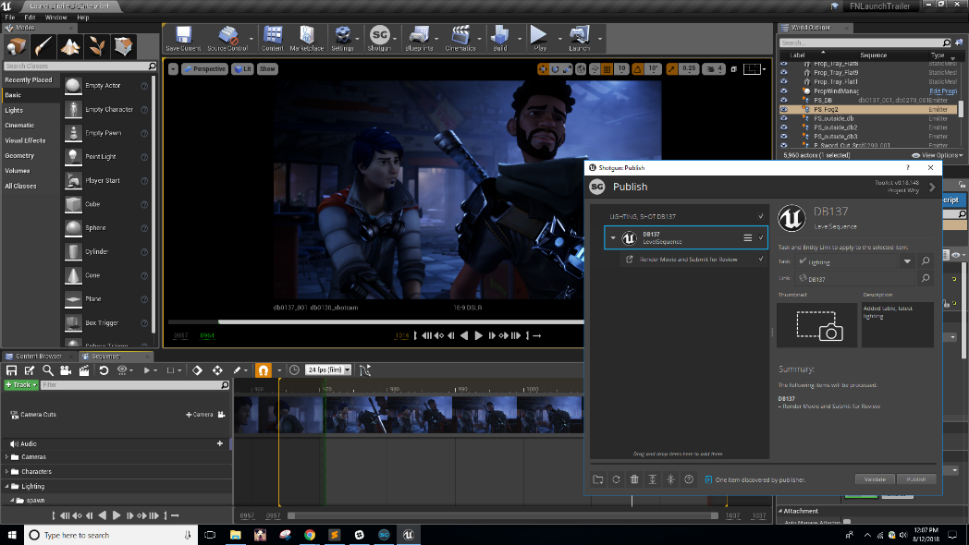


Рисунок 11 – Интерфейс и пример работы Unreal Engine

Платформа предоставляется бесплатной для физических лиц, готовых шаблонов меньше, чем у Unity.

* + 1. **Godot**

Современный движок, являющийся достойным претендентом на сравнение с Unity в разработке 2D приложений, большая часть интерфейса была русифицирована, увеличивая в разы круг людей, желающих обучиться разработке игровых кроссплатформенных приложений.

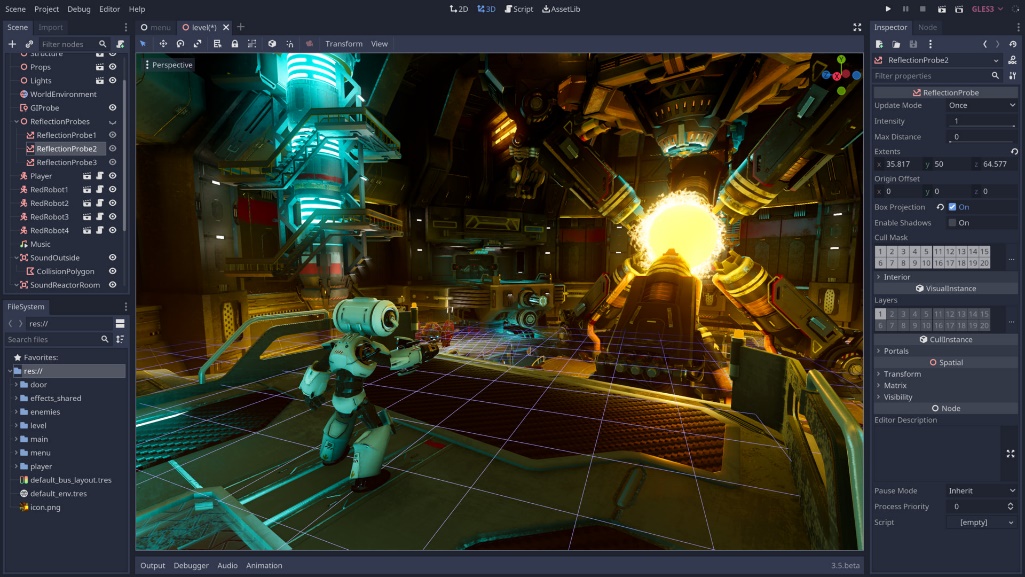


Рисунок 12 – Интерфейс и пример работы Godot

Является бесплатным, поддерживает несколько языков программирования, таких как: GDScript, C++, а C# имеет неполную поддержку. Ниже приведена таблица для сравнения игровых движков (таблица 3).

Таблица 3 – Сравнение игровых движков

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики** | **Unity** | **Unreal Engine** | **Godot** |
| Понятный интерфейс | + | - | + |
| Удобство пользования | + | + | – |
| Наличие ассетов | + | + | – |
| Кроссплатформенность | + | + | – |
| Русификация | – | – | + |

По итогам анализа был выявлен подходящий для разработки 2D приложения игровой движок – Unity. Ведь он имеет в наличии огромное количество ассетов и обладает удобным и понятным интерфейсом, что является огромным преимуществом относительно конкурентов.

* 1. **Постановка задачи**

Перед готовым приложением ставится ряд задач его функционала. Поскольку приложение является игрой в жанре Платформера и Шутера, то она должна быть в 2D–графике и с элементами стрельбы по целям/противникам.

Также в приложении должна присутствовать визуальная составляющая для объектов заднего фона, объектов препятствующих игроку. Модель игрока, как и ранее упомянутые объекты, не должны выделяться из общей концепции игрового приложения.

# **Проектный раздел**

# **Разработка алгоритмов**

При разработке игрового приложения требуется определить ряд функций, которые оно должно выполнять на протяжении активности. В игровом приложении “Межвременные войны” используются 3 функции для достижения рабочего состояния программы:

* передвижение;
* стрельба;
* прыжок.

# **Передвижение**

Для грамотного передвижения персонажа по внутриигровому миру, для назначения скорости перемещения и определения точки соприкосновения с платформой требуется применить физику игрового движка.

Используя в Unity среду разработки C# единицей измерения является – f (например, 1f). За начальное значение в переменных берется 0 для грамотной корректировки функций (рисунок 13).

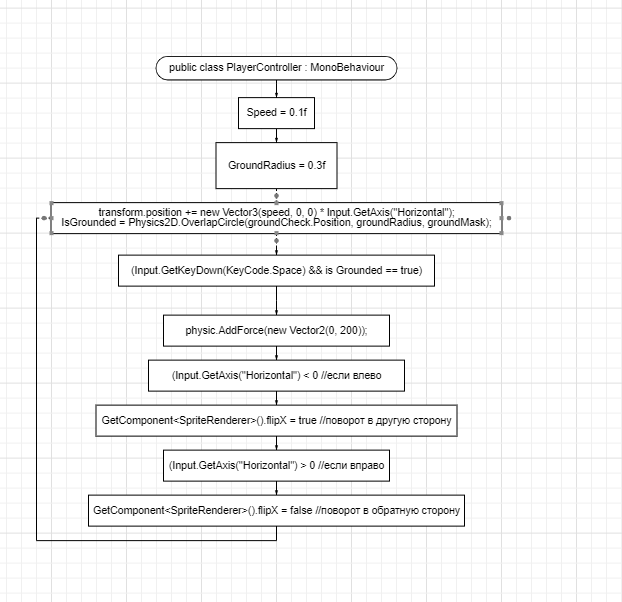


Рисунок 13 – Блок-схема алгоритмов

Так же для персонажа требуется назначить местонахождения платформ и отображение модели в зависимости от направления движения.

# **Стрельба**

Учитывая выбор жанра игрового приложения Шутер – в игре должна присутствовать функция стрельбы, полету пули и её направления. Для самого выстрела требуется назначить исходную точку, то есть, точку откуда пуля начнет свое движение (рисунок 14).

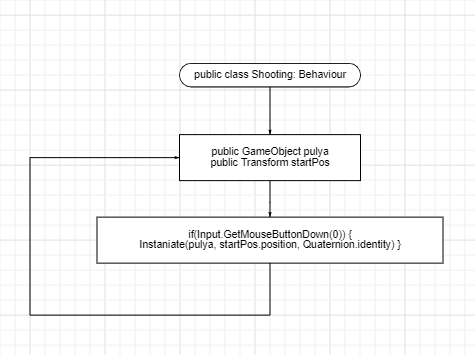


Рисунок 14 – Блок-схема алгоритма выстрела

Для самого выстрела следует назначить кнопку, при нажатии которой выстрел будет воспроизведен (рисунок 15).

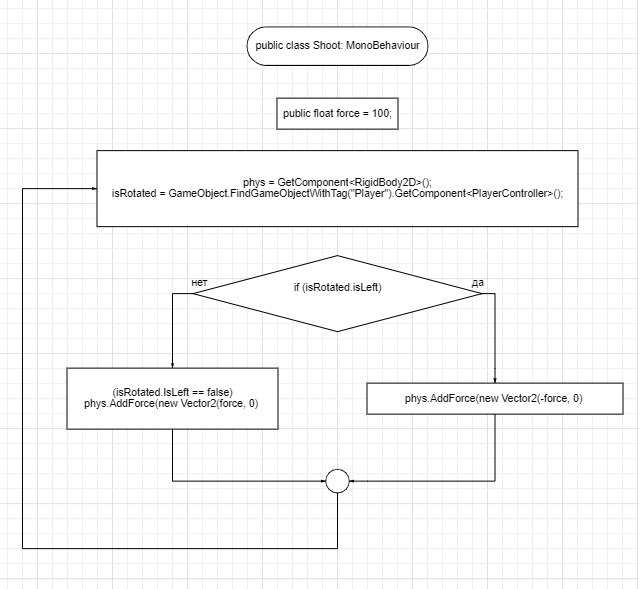


Рисунок 15 – Блок-схема алгоритма полета пули и направления

После назначения скорости полета пули требуется назначить направление полета снаряда в зависимости от стороны, в которую смотрит персонаж.

# **Прыжок**

Так же для грамотного передвижения в игровом мире потребуется настроить прыжок персонажа для преодоления вертикальных препятствий и преодоления пропастей на платформах.

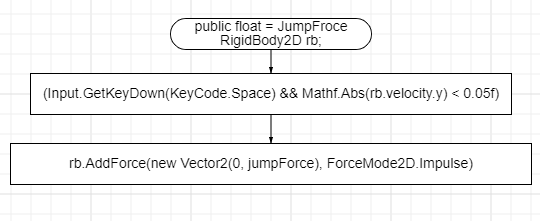


Рисунок 16 – Блок-схема алгоритма прыжка

Функция прыжка осуществляется нажатием на кнопку “Space” на клавиатуре.

# **Разработка программного кода для программного продукта**

Зачастую пользователи не знают о том, что находится в корне программы, какая использовалась среда разработки, как был написан код.

В данном разделе будет рассматриваться код и его написание. Для грамотного создания персонажа требуется два кода, для контроля им и для обозначения полного здоровья, получения им урона. Контроль персонажем (передвижение и прыжок) осуществляется при помощи данного кода:

public class PlayerController : MonoBehaviour

{

public float speed = 0.1f;

public Rigidbody2D physic;

public float jumpForce = 600;

public bool isLeft = false;

private bool isGrounded;

private float groundRadius = 0.3f;

public Transform groundCheck; //позиция ног персонажа

public LayerMask groundMask;

private void Update()

{

transform.position += new Vector3(speed, 0, 0) \* Input.GetAxis("Horizontal");

isGrounded = Physics2D.OverlapCircle(groundCheck.position, groundRadius, groundMask); //проверка, стоит ли игрок на земле

if (Input.GetKeyDown(KeyCode.Space) && isGrounded == true)

{

physic.AddForce(new Vector2(0, jumpForce));

}

if (Input.GetAxis("Horizontal") < 0) //если влево

{

GetComponent<SpriteRenderer>().flipX = true;

isLeft = true;

}

if (Input.GetAxis("Horizontal") > 0) //если вправо

{

GetComponent<SpriteRenderer>().flipX = false;

isLeft = false;

}

}

}

Обозначение здоровья и получение урона персонажу:

public class PlayerInteraction : MonoBehaviour

{

private float health = 1f;

public void TakeDamage(float damage)

{

health -= damage;

if(health <= 0f)

{

Die();

}

}

private void Die()

{

SceneManager.LoadScene(0);

}

}

Для стрельбы использовалось два кода:

Для самого выстрела, его исходящей точки и назначения клавиши для выстрела:

public class Shooting : MonoBehaviour

{

public GameObject pulya;

public Transform ShootPos;

void Update()

{

if (Input.GetMouseButtonDown(0))

{

Instantiate(pulya, ShootPos.position, Quaternion.identity);

}

}

}

И для полета пули, введения её в игровое приложение для изменения скорости полета, исчезновения после определенного времени полета, а так же нанесения урона для уничтожения противников:

public class Shoot : MonoBehaviour

{

public float force = 100;

private float damage = 0.5f;

private Rigidbody2D phys;

public PlayerController isRotated;

private void Start()

{

phys = GetComponent<Rigidbody2D>();

isRotated = GameObject.FindGameObjectWithTag("Player").GetComponent<PlayerController>();

if (isRotated.isLeft)

{

phys.AddForce(new Vector2(-force, 0));

}

else if (isRotated.isLeft == false)

{

phys.AddForce(new Vector2(force, 0));

}

Destroy(gameObject, 0.5f);

}

void OnTriggerEnter2D(Collider2D other)

{

if(other.name == "Enemy")

{

other.GetComponent<EnemyInteraction>().TakeDamage(damage);

}

}

}

Поскольку в игровом приложении имеются противники, то для них так же требуются два программных кода: первый обозначит дистанцию, с которой игровой ИИ будет передвигаться в сторону персонажа и наносить урон:

private Rigidbody2D physic;

public Transform player;

public float speed;

public float agroDistance;

void Start()

{

physic = GetComponent<Rigidbody2D>();

}

void Update()

{

float distToPlayer = Vector2.Distance(transform.position, player.position);

if(distToPlayer < agroDistance)

{

StartHunting();

}

else

{

StopHunting();

}

}

void StartHunting()

{

if(player.position.x <transform.position.x) //идет влево

{

physic.velocity = new Vector2(-speed, 0);

transform.localScale = new Vector2(6, 6);

}

else if (player.position.x > transform.position.x)

{

physic.velocity = new Vector2(speed, 0);

transform.localScale = new Vector2(-6, 6);

}

}

void StopHunting()

{

physic.velocity = new Vector2(0, 0);

}

}

Второй программный код используется для обозначения единиц здоровья и единиц наносимого урона противниками:

public class EnemyInteraction : MonoBehaviour

{

[SerializeField]

private float health = 1f;

private float damage = 0.25f;

void OnTriggerEnter2D(Collider2D other)

{

if(other.name == "Player")

{

other.GetComponent<PlayerInteraction>().TakeDamage(damage);

}

}

public void TakeDamage(float amount)

{

health = amount;

if(health <= 0f)

{

Die();

}

}

void Die()

{

Destroy(gameObject);

}

}

# **Разработка интерфейса пользователя**

В игровом приложении интерфейс пользователя исполнен в виде начального меню с двумя кнопками, одна – для запуска игрового мира с персонажем, вторая – для выхода из игрового приложения.

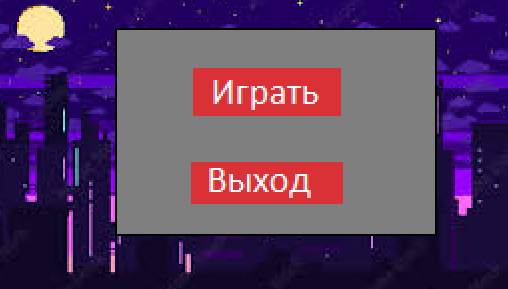


Рисунок 17 – Интерфейс (главное меню)

Цвета для панели меню, кнопок и текста кнопок были отобраны для более яркого восприятия дальнейшего геймплея и при помощи цветового круга.

* 1. **Тестирование и отладка программного продукта.**

Готовый программный продукт проверяется вручную, либо автоматизированными программами созданных специально для проверки разных типов созданных программ.

Ручная проверка игрового приложения “Межвременные войны” на работоспособность осуществляется при помощи множества использований всех функций. Для ручной проверки игрового приложения будут рассматриваться функции передвижения и стрельбы, поскольку они считаются основными в программе.

Если при передвижении персонаж не контактирует с физическим объектом и проходит сквозь него – это является багом.

Для решения подобной проблемы требуется установить на модель персонажа и физический объект “Collider2D”. После его установки баг будет исправлен и персонаж будет соприкасаться с объектом.

Багом в функции стрельбы является полет пули в обратную сторону от той, в которую смотрит модель персонажа пользователя. Для исправления данного дефекта потребовалось изменить в программном коде значение “ОсьX”. После исправления пуля персонажа полетит в ту сторону, в которую смотрит персонаж.

ЗАКЛЮЧЕНИЕ

В ходе курсового проекта на тему «Проектирование и разработка игры “Межвременные войны”» в соответствии с техническим заданием сделано следующее:

* Проведено предпроектное обследование жанров игровых приложений, языков программирования, среды разработки и игрового движка;
* Проанализированы и выбраны подходящие для разработки игрового приложения жанры, языки программирования, среды разработки и игрового движка;
* Поставлены задачи к программному продукту в виде жанра игрового приложения и 2D*–*графики.
* Разработаны алгоритмы, код программного продукта и интерфейс пользователя;
* Протестированы основные функции игрового приложения.

Благодаря игровому движку Unity игровое приложение было разработано в пиксельной 2D*–*графике с рабочим функционалом в проекте.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления: Стандартинформ, 2017
2. ГОСТ 7.1—2003 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления
3. ГОСТ 7.9—95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования
4. ГОСТ 7.11—2004 (ИСО 832:1994) Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках
5. ГОСТ 7.12—93 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила
6. ГОСТ 7.80—2000 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления
7. ГОСТ 7.82—2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления
8. Гейг, М. А. Разработка игр на Unity 2018 за 24 часа / М. А. Гейг. [перевод с английского М. А. Райтмана], М.: Эксмо, 2020. – 464 с.
9. Гвоздева, В. А. Введение в специальность программиста / В. А. Гвоздева. – М.: ИД “ФОРУМ” – ИНФРА, 2007. – 207 c.
10. Godot [Электронный ресурс]. – Режим доступа: https://docs.godotengine.org/en/stable/ – Загл. с экрана.
11. Unity [Электронный ресурс]. – Режим доступа: https://unity.com/ru/learn – Загл. с экрана.
12. Unreal Engine [Электронный ресурс]. – Режим доступа: https://www.unrealengine.com/en-US/learn – Загл. с экрана.
13. Visual Studio [Электронный ресурс]. – Режим доступа: https://visualstudio.microsoft.com/ru/ – Загл. с экрана.
14. Eclipse [Электронный ресурс]. – Режим доступа: https://www.eclipse.org/ide/ свободный. – Загл. с экрана.
15. Visual Studio Code [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com/learn свободный. – Загл. с экрана.