Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина: Объектно-ориентированное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

#### Игра «SpaceWar» для платформы под управлением Windows

БГУИР КП I – 40 04 01

Выполнил

студент гр. 053501 Селивестров В.А.

Проверил: Владымцев В.Д.

Минск 2022

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой Информатика

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Волорова Н. А. 2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Селивестрову Вадиму Александровичу*

1. Тема работы Игра «SpaceWar» для платформы под управлением Windows *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе

|  |
| --- |
| Тип операционной системы – ОС Windows; |
| Языки программирования – C#; Библиотеки –MonoGameLibrary | |
| Фреймворки –MonoGame. | |
| Цель проекта: разработка игры для Windows | |

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| 1 Анализ аналогов и формирование требований к игре |
| 2 Обзор используемых технологий |
| 3 Проектирование игры |
| 4 Программная реализация |
| 5 Тестирование игры |
| Заключение |
| Список использованных источников |
| Приложение А - Текст программы |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А4*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе Владымцев В. Д.

7.Дата выдачи задания *16.02.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

*Раздел 1. Введение к 28.02.2022г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.03.2022г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2022г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.05.2022г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5. Заключение. Приложения к 30.05.2022г. - 90% готовности работы;*

*оформление пояснительной записки и графического материала к 07.06.2022г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 30.05.2022г. по 09.06.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Владымцев В. Д.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Селивестров В. А. 16.02.2022г.*

*(дата и подпись студента)*

Содержание

[ВВЕДЕНИЕ 4](#_Toc105505252)

[1. Общие сведения 6](#_Toc105505253)

[**1.1** **Классификация однопользовательских игр** 6](#_Toc105505254)

[1.1.1 Экшен 6](#_Toc105505255)

[**1.1.2** **Аркада** 7](#_Toc105505256)

[**1.1.3** **Шутер** 7](#_Toc105505257)

[**1.1.4** **Файтинг** 8](#_Toc105505258)

[**1.1.5** **Итог** 8](#_Toc105505259)

[1.2 Инструментальные средства 9](#_Toc105505260)

[**1.2.1** **Microsoft Paint** 9](#_Toc105505261)

[**1.2.2** **SketchBook Autodesk** 9](#_Toc105505262)

[**1.2.3** **Photoshop** 9](#_Toc105505263)

[**1.2.4** **Язык программирования C#** 10](#_Toc105505264)

[**1.2.5** **Monogame** 13](#_Toc105505265)

[**2.** **Анализ предметной области** 14](#_Toc105505266)

[2.1 Описание главного меню приложения 15](#_Toc105505267)

[2.2 Описание игры 15](#_Toc105505268)

[3. Проектирование игры 18](#_Toc105505269)

[3.1 Концептуальная модель 18](#_Toc105505270)

[3.2 Графическая составляющая 19](#_Toc105505271)

[4. Программная реализация 31](#_Toc105505272)

[5. Тестирование приложения 33](#_Toc105505273)

[ЗАКЛЮЧЕНИЕ 35](#_Toc105505274)

[СПИСОК ЛИТЕРАТУРЫ 36](#_Toc105505275)

# ВВЕДЕНИЕ

Рынок компьютерной индустрии является самой масштабной частью мирового рынка.

Компью́терная игра́ — [компьютерная программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), служащая для организации [игрового](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0) процесса ([геймплея](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%B9%D0%BC%D0%BF%D0%BB%D0%B5%D0%B9)), связи с партнёрами по игре, или сама выступающая в качестве партнёра.

В настоящее время, в ряде случаев, вместо термина компьютерная игра может использоваться [видеоигра](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D0%B0), то есть данные термины могут употребляться как синонимы и быть взаимозаменяемыми. В компьютерных играх, как правило, игровая ситуация воспроизводится на экране дисплея или обычного телевизора (в этом случае компьютерные игры одновременно являются и [видеоиграми](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE%D0%B8%D0%B3%D1%80%D0%B0)), но в то же время компьютерная игра может быть [звуковой](https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), [телетайповой](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D1%82%D0%B0%D0%B9%D0%BF" \o "Телетайп) и другой.

Компьютерные игры могут создаваться на основе [фильмов](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%BE%D1%84%D0%B8%D0%BB%D1%8C%D0%BC) и [книг](https://ru.wikipedia.org/wiki/%D0%A5%D1%83%D0%B4%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5); есть и обратные случаи. С 2011 года компьютерные игры официально признаны в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90) отдельным [видом искусства](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%BE).

Компьютерные игры являются одной из [драматических форм](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%BC%D0%B0_(%D0%B6%D0%B0%D0%BD%D1%80)), а их [интерактивность](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) — это вопрос степени участия, но не формы. Поэтому, как и другие формы, компьютерная игра имеет пять ключевых элементов: стиль, [фабула](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%83%D0%BB%D0%B0), герой, декорации и тема. Все хорошие игры должны обладать некоторым развлекательным потенциалом, и в большинстве их он основан на классических законах [драмы](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%BC%D0%B0_(%D0%B6%D0%B0%D0%BD%D1%80)).

По сообщению сайта 3DNews, в 2022 году компьютерные игры были официально признаны [правительством США](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D1%88%D0%B8%D0%B5_%D1%84%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D1%8B_%D0%B3%D0%BE%D1%81%D1%83%D0%B4%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9_%D0%B2%D0%BB%D0%B0%D1%81%D1%82%D0%B8_%D0%A1%D0%A8%D0%90) и американским [Национальным фондом искусств](https://ru.wikipedia.org/w/index.php?title=%D0%9D%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B4_%D0%B8%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2&action=edit&redlink=1) отдельным [видом искусства](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%BE), наряду с театром, кино. После этого разработчики получили право, наравне с представителями кинематографа, музыки, живописи и литературы, рассчитывать на государственные [гранты](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BD%D1%82) в размере от 10 до 200 тыс. долларов. Данная финансовая поддержка позволит независимым специалистам и компаниям значительно активней реализовать концептуальные проекты.

За предыдущий год сегмент видеоигр достиг объёма в 180 миллиардов долларов.  
 Одну из крупнейших ниш, занимает ниша однопользовательских игр, которые на сегодняшний день насчитывают огромное количество жанров. Среди них перспективным направлением считается разработка простых игр в совместном жанре Аркада, Шутер, Экшен.

Таким образом была выявлена цель курсовой работы, а именно: создать игровой проект.  
 Для достижения этой цели требуются:

* Провести анализ предметной области;
* Освоить инструментальные средства;
* Создать концептуальную модель;
* Осуществить отрисовку дизайна;
* Совершить программную реализацию игрового проекта.

## Общие сведения

* 1. **Классификация однопользовательских игр**

Однопользовательская игра или одиночная игра (англ. single-player — «один игрок») — режим компьютерной игры, во время которого с ней через устройства ввода-вывода взаимодействует один человек.

Альтернативным режимом к однопользовательскому является [многопользовательский](https://www.wikiwand.com/ru/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0). В игре может быть как выбор между однопользовательским и многопользовательским режимами, так и предоставление доступа к одному из них.

Однопользовательская игра организована как интерактивный диалог пользователя (автора по сути) с игрой, в котором раскрывается содержание. В отличие от неё, многопользовательские игры обеспечивают объединение игроков в совместное сотворчество или соревнование.

Компьютерная игра — компьютерная программа, служащая для организации игрового процесса (геймплея), связи с партнёрами по игре, или сама выступающая в качестве партнёра.

Игровой процесс, или геймплей (англ. gameplay), — компонент игры, отвечающий за взаимодействие игры и игрока. Геймплей описывает, как игрок взаимодействует с игровым миром, как игровой мир реагирует на действия игрока и как определяется набор действий, который предлагает игроку игра.

### 1.1.1 Экшен

Экшен (англ. action — «действие») — жанр компьютерных игр, в котором делается упор на эксплуатацию физических возможностей игрока, в том числе координации глаз и рук и скорости реакции. В экшен-играх обычно игрок управляет протагонистом или аватаром. Этот персонаж должен найти выход из уровня, собрать предметы, избежать препятствий и сразиться с врагами разными способами. Действие таких игр развивается очень динамично и требует высокой концентрации внимания и быстрой реакции на происходящие в игре события. В конце уровня или ряда уровней игрок обычно сражается с боссом, битва с которым более требовательна к игроку, а сам босс, зачастую, крупнее обычных врагов. Препятствия и вражеские атаки истощают здоровье и запас жизней аватара. При отсутствии у него жизней, игрок получает сообщение «Game over» (англ. Игра окончена). В другом случае, когда серия уровней успешно пройдена, игрок побеждает. Тем не менее, в ряде игр, особенно в аркадах, количество уровней может быть бесконечно, и победить в таких играх невозможно. В этом случае целью игрока становится получение как можно большего количества очков, собирая предметы и уничтожая врагов.

* + 1. **Аркада**

Аркада (англ. Arcade) — Игры, в которых игроку приходится действовать быстро, полагаясь в первую очередь на свои рефлексы и реакцию. Аркады характеризуются развитой системой бонусов: начисление очков, бесконечная игра, отсутствие сюжета/истории, простой игровой процесс, игра на одном экране.

В мировой практике, а также в прессе, аркадами называются игры для aркaдных игровых автоматов. Это не отдельный жанр игр, а скорее игровое направление. Компьютерная игра называется «аркадной» в том случае, если она напрямую портирована с автомата или же схожа по концепции с играми для автоматов. Например, к аркадным играм относятся все проекты жанров «файтинг» (fighting), часть игр жанра «шутер» (shooter).

* + 1. **Шутер**

Шутер (Стрелялка, англ. shooter — «стрелок») — жанр компьютерных игр. На момент зарождения жанра за рубежом укрепилось слово «шутер», как вариант описания игрового процесса и перевод для слова shooter, в странах постсоветского пространства жанр изначально был назван как «стрелялка». Игрок находится в трёхмерном/двумерном пространстве и имеет некоторую свободу передвижения. Уровни, как правило, являют собой ограниченный лабиринт, в котором расположены враги, союзники и нейтрально настроенные NPC.

* + 1. **Файтинг**

Файтинг (англ. Fighting — бой, драка, поединок, борьба) — жанр компьютерных игр, имитирующих рукопашный бой малого числа персонажей в пределах ограниченного пространства, называемого ареной (часть игровой вселенной, не управляемая участником игры, на которой происходят основные действия игры). Важной особенностью файтингов является их нацеленность на соревнование, а не на сотрудничество игроков, что делает игры этого жанра подходящими для киберспортивных чемпионатов. Обычно файтинги предоставляют игроку возможность вести бой в режиме «один на один» против компьютерного противника или другого игрока, реже — позволяют сражаться одновременно трём или четырём противникам на одной арене.

* + 1. **Итог**

Однопользовательские игры – перспективный жанр. Их популярность обуславливается неограниченной свободой действий и отсутствием токсичного комьюнити. Для начинающего разработчика однопользовательские игры зачастую наилучший вариант.

## Инструментальные средства

Чтобы сделать проект более уникальным, я обратился за помощью к знакомой иллюстраторше, которая работает над созданием уникальных графических изображений, все изображения защищены авторскими правами и являются уникальными

* + 1. **Microsoft Paint**

Microsoft Paint — многофункциональный, но в то же время довольно простой в использовании [растровый графический редактор](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80) компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), входящий в состав всех операционных систем [Windows](https://ru.wikipedia.org/wiki/Windows), начиная с первых версий.

Имеет следующие возможности:

* Выбор третьего цвета для рисования;
* Пипетка;
* Заливка;
* Замена цвета или ластик;
* Десятикратное увеличение;
* Выделение;
* Пользовательская кисть.
  + 1. **SketchBook Autodesk**

Это программное обеспечение для растровой графики, предназначенное для выразительного рисования и создания эскизов концепций, а также для создания анимации. Первоначально разработанное как коммерческое программное обеспечение, оно превратилось в модель подписки, прежде чем в конечном итоге стало бесплатным для личного пользования.

* + 1. **Photoshop**

Многофункциональный графический редактор, разрабатываемый и распространяемый компанией Adobe Systems. В основном работает с растровыми изображениями, однако имеет некоторые векторные инструменты. Продукт является лидером рынка в области коммерческих средств редактирования растровых изображений и наиболее известной программой разработчика.

* + 1. **Язык программирования C#**

C# (произносится как "си шарп") — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# — объектно-ориентированный, **ориентированный на компоненты** язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — **объектно-ориентированный** язык. Вы определяете типы и их поведение.

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. [**Сборка мусора**](https://docs.microsoft.com/ru-ru/dotnet/standard/garbage-collection/) автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. [**Типы, допускающие значение null,**](https://docs.microsoft.com/ru-ru/dotnet/csharp/nullable-references) обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. [**Обработка исключений**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/exceptions/) предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. [**Лямбда-выражения**](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/lambda-expressions) поддерживают приемы функционального программирования. [**Синтаксис LINQ**](https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/) создает общий шаблон для работы с данными из любого источника. Поддержка языков для [**асинхронных операций**](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/) предоставляет синтаксис для создания распределенных систем. В C# имеется [**Единая система типов**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/types/). Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями [ссылочные типы](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/reference-types), так и [типы значений](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-types). C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

C# подчеркивает **Управление версиями** , чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Для первого знакомства с языком программирования традиционно используется программа "Hello, World". Вот ее пример на C#:

using System;

class Hello

{

static void Main()

{

Console.WriteLine("Hello, World");

}

}

Программа "Hello, World" начинается с директивы using, которая ссылается на пространство имен System. Пространства имен позволяют иерархически упорядочивать программы и библиотеки C#. Пространства имен содержат типы и другие пространства имен. Например, пространство имен System содержит несколько типов (в том числе используемый в нашей программе класс Console) и несколько других пространств имен, таких как IO и Collections. Директива using, которая ссылается на пространство имен, позволяет использовать типы из этого пространства имен без указания полного имени. Благодаря директиве using в коде программы можно использовать сокращенное имя Console.WriteLine вместо полного варианта System.Console.WriteLine.

Класс Hello, объявленный в программе "Hello, World", имеет только один член — это метод с именем Main. Метод Main объявлен с модификатором static. Методы экземпляра могут ссылаться на конкретный экземпляр объекта, используя ключевое слово this, а статические методы работают без ссылки на конкретный объект. По стандартному соглашению точкой входа программы C# является статический метод с именем Main.

Выходные данные программы создаются в методе WriteLine класса Console из пространства имен System. Этот класс предоставляется библиотеками стандартных классов, ссылки на которые компилятор по умолчанию добавляет автоматически.

Основными понятиями Организации в C# являются [**программы**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/program-structure/), [**пространства имен**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/types/namespaces), [**типы**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/types/), [**члены**](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/members) и [**сборки**](https://docs.microsoft.com/ru-ru/dotnet/standard/assembly/). В программе объявляются типы, которые содержат члены. Эти типы можно организовать в пространства имен. Примерами типов являются классы, структуры и интерфейсы. К членам относятся поля, методы, свойства и события. При компиляции программы на C# упаковываются в сборки. Сборки обычно имеют расширение .exe файла или .dll , в зависимости от того, реализуют ли они .exe или **библиотеки** соответственно.

Данный язык был выбран в качестве основного, так как соответствует всем стандартам качества, имеет обширную библиотеку классов, и очень удобен в разработке.

* + 1. **Monogame**

Для реализации технических возможностей, использовалась библиотека MonoGame

MonoGame представляет кроссплатформенную OpenSource-реализацию популярного фреймворка Microsoft XNA 4, который предназначен для работы с графикой и прежде всего для создания игр. Microsoft XNA 4 позволял с помощью стандартных для платформы .NET языков C# и VB.NET относительно быстро и легко создавать сложные визуальные сцены вплоть до трехмерных игр с богатой графикой и изощренными игровыми сценариями.

Платформа имеет открытый код, что позволяет ей расти и расширяться. Так же данная платформа имеет кроссплатформенность, которая на данный момент поддерживает iOS, Android, MacOS, Linux, Windows, PS4, Xbox One, Switch а так же много других

1. **Анализ предметной области**

Игра Space War представляет собой следующий набор сущностей:

* Сущность **Player** (Игрок), сущность player будет под управлением Пользователя и будет иметь следующие поля: поле, отвечающее за текущее здоровье, поле, отвечающее за количество убитых **Enemy** (враг), поле, отвечающее за скорость **Speed** (скорость).
* Сущность **Enemy** (Враг), сущность враг будет под управлением компьютера и будет иметь свойство **Attack.**
* Сущность **Bullet** (Пуля), сущность пуля будет создаваться при вызове метода **Attack,** и будет иметь поле **Speed** (скорость). Так же эта сущность будет в себе хранить поле **Damage** который при попадании в сущность враг или игрок, будет наносить урон, уменьшающий количество здоровья.
* Сущность **World** (игровой мир), будет представлять собой фоновое изображение, поверх которого будут отображаться все перечисленные сущности.
* Сущность **Benefits** («плюшки»), «плюшки» будут появляться на месте убитых врагов, и представляют собой усиления полей сущности игрок или ослабление полей сущности враг.
* Сущность **Score** (счёт), счёт представляет собой набор очков, набранных сущностью, игрок, за количество уничтоженных врагов и подобранных плюшек.

Сущности Игрок и враг, будут наследоваться от общей сущности, **SpaceShip** (корабль), имеющий общие параметры.

Цель игры набрать наибольшее количество очков, и побить результат из таблицы лидеров(которая будет храниться в базе данных)

## 2.1 Описание главного меню приложения

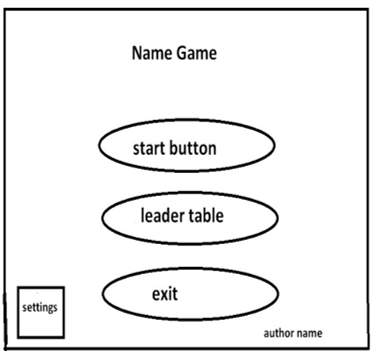
****

Рисунок 1 – Главное меню приложения

Само меню будет запускаться при запуске игры, у меня будет 4 интерактивные кнопки.

Кнопка «start button» непосредственно будет запускать саму игру.

Кнопка «leader table» откроет новое окно с таблицей лидеров, которая будет выгружаться из базы данных.

Кнопка «exit» будет завершать жизненный цикл программы.

Кнопка «settings» откроет новое окно в котором будут доступны базовые настройки для комфортной игры, будет возможность ввода «чит кода», который при запуске следующей игры даст весомые преимущества.

## 2.2 Описание игры

В самой игре у игрока будет под контролем сущность, которая может перемещаться по игровому полю и стрелять

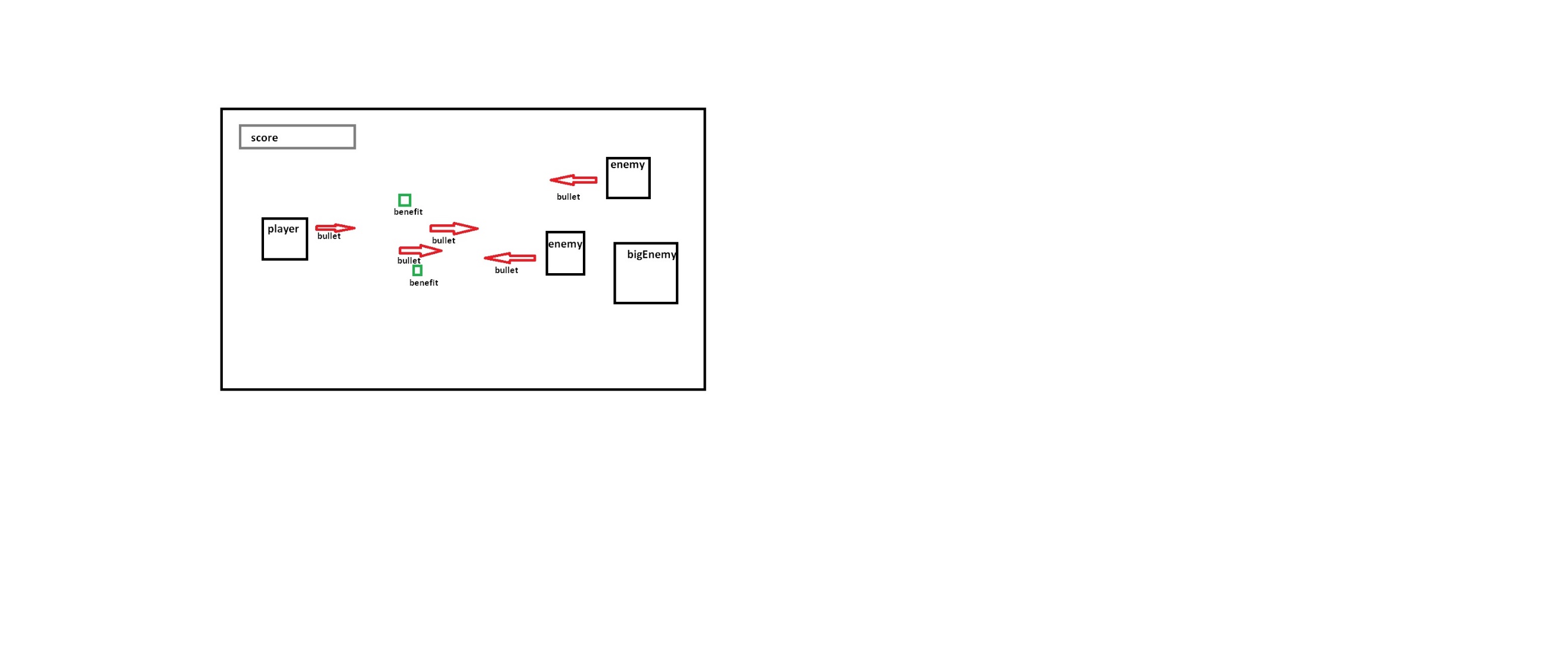


Рисунок 2 - Схематическое отображение игрового мира

При достижении игроком счёта, который больше чем «рекорд», игроку предлагается ввести имя, для занесения его в таблицу лидеров.

Суть игры будет заключаться в том, чтобы «стать лучшим» в таблице лидеров(аркада) или хотя бы попытаться.

По сути игра будет представлять собой следующий набор требований:

1. Игра должна быть бесконечной, до того момента пока игрок не сделает n-ое количество ошибок (где n – это количество жизней).
2. Игра должна со временем «усложняться», чтобы игроку не казалось, что его прогресс застыл на месте.
3. Игра на одном экране. Игрок в любой момент времени должен видеть весь игровой мир и принимать решения.
4. Интуитивный игровой процесс, игроку легко научиться геймплею, но становится практически невозможным стать мастером в игре из-за её сложности. Вместе с тем, если игрок погибает в игре, то это практически всегда происходит по его вине.
5. Нет сюжета или истории. игрок должен понять, что происходит — и сразу влиться в игру. В классических аркадах сеттинг обычно следующий — это научная фантастика, война, спорт или что-то ещё. Для этой аркады сеттинг находится пока на уровне засекречено.
6. Игровой счёт. Когда игрок получает очки за выполнение различных целей или задач.
7. Множество жизней. Такой подход позволяет новому игроку получить большую возможность изучить игровые механики до того, как игра заканчивается.

# Проектирование игры

## 3.1 Концептуальная модель

Концептуальная модель нужна для того, чтобы определить cтруктурные элементы предметной области и обозначить связи между ними. В итоге концептуальная модель представляет собой логическую структуру рассматриваемой области.

Концептуальная модель - первый и наиболее важный шаг для создания

игрового проекта. На данном этапе геймдизайнер создает и описывает свои идеи в специальном документе. На выходе должен получиться документ, описывающий игру, как конечный продукт, а также начальную проработку всех элементов игры. Далее документ используется тестировщиками, продюсерами, дизайнерами, программистами и инвесторами.

Так как данный игровой проект создается одним начинающим разработчиком, то имеет смысл описать концепт лишь в виде тезисов, и далее

изменять их по ходу решения практических задач.

При описании концептуальной модели игрового проекта были сформулированы следующие тезисы:

**жанр**: аркада;

**режим**: однопользовательская игра;

**графика**: растровая;

**пространство**: 2D;

**вид камеры**: сбоку;

**цель**: победить секретного босса набрав при этом большее количество очков;

**мир**: состоит из сущностей, где каждая сущность имеет своё поведение;

**вражеский корабль**: сущность

- необходимо уничтожить;

- появляется с некоторой периодичностью.

**Спавн поинт**: объект

- нельзя уничтожить;

- порождает вражеские корабли.

**Счёт**: объект

- показывает общий игровой счёт;

**средство**: MonoGame;

**язык программирования**: C#.

Таким образом, в данных тезисах были определены детали игры, описаны игровые объекты, а также обозначены средства программирования, необходимые для реализации проекта

# 3.2 Графическая составляющая

Для начала нужно получить исходные изображения чтобы с ними работать.

Подготовка эскизов:

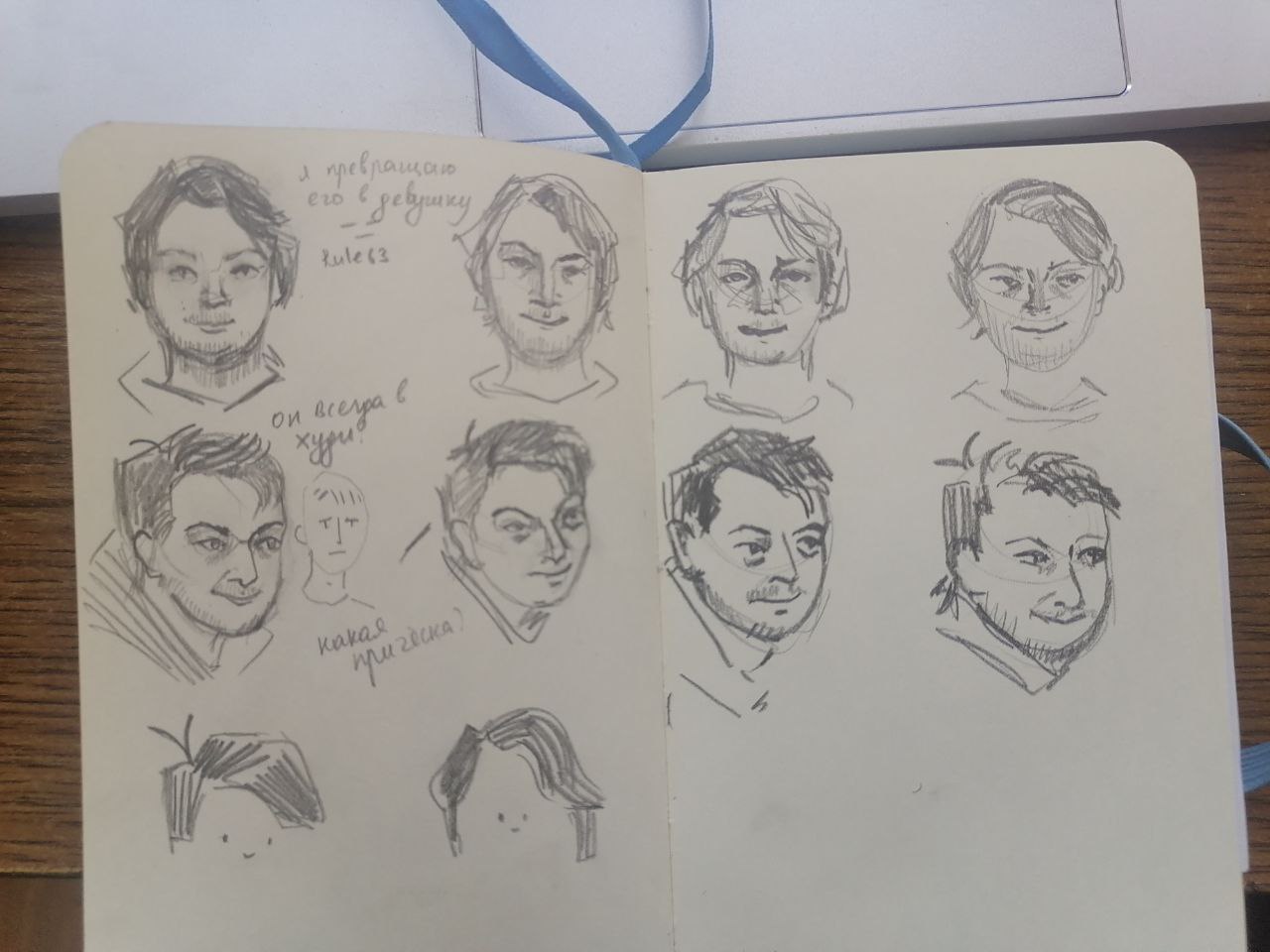


Рисунок 3 - Общие эскизы

В ходе исходных фотографий моим субъективным чувством прекрасного был выбран и реализован следующий эскиз:



Рисунок 4 - Зарисовочный эскиз

Конечный вариант выглядит так:



Рисунок 5 - Сущность Игрок

Рисование кораблей.

Тут всё просто, был нарисован обычный кораблик

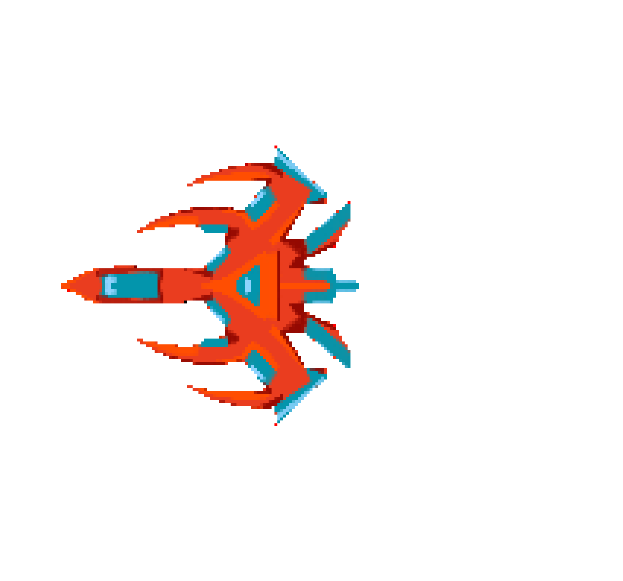


Рисунок 6 - Сущность корабля

Рисование снарядов

Т.к. этот пункт я забыл уточнить, а добавлять потом, когда поджимали сроки было неловко, пришлось выкручиваться и рисовать изображение самостоятельно, по просьбе проверяющих эту курсовую снаряд имеет следующий вид:



Рисунок 7 - Сущность Снаряда

Готовый игровой мир:

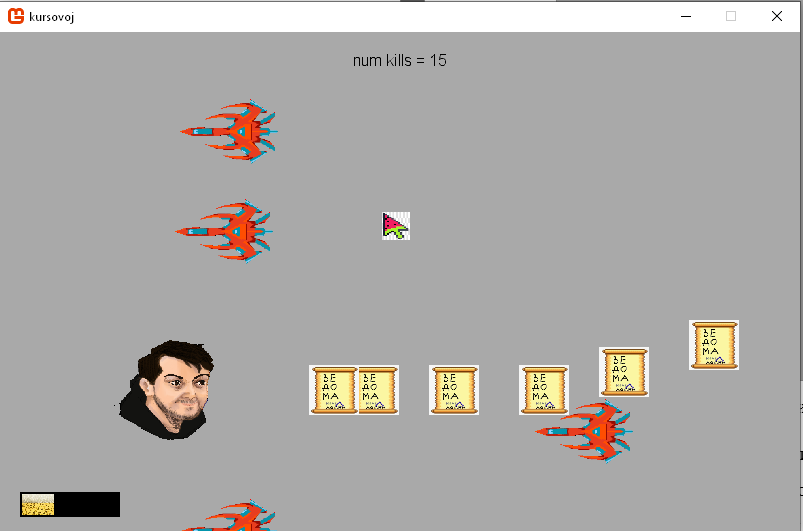


Рисунок 8 - Игровой процесс

Предварительная диаграмма классов выглядит следующим образом:

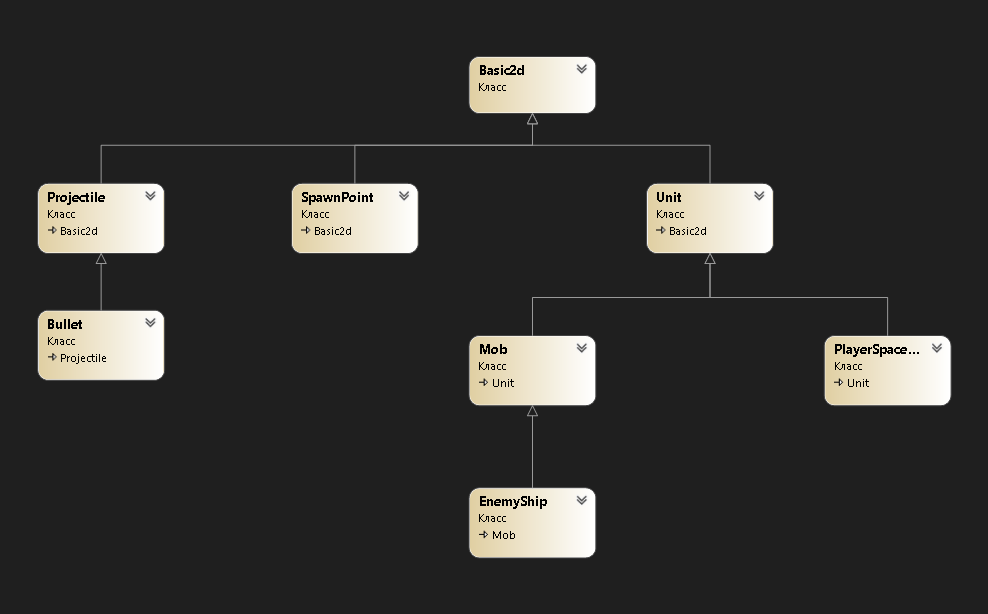


Рисунок 9 - Диаграмма классов

Для данного проекта был написан свой мини-игровой движок представляющий собой следующие возможности:

* Базовый класс для всех объектов;
* Использование элементов управления через клавиатуру и мышь;
* Собственный таймер, который производит подсчёт игрового времени;
* Музыкальный проигрыватель для воспроизведения звуков игры для погружения в аутентичность и атмосферу данной аркады.

Базовый класс для всех объектов представляет собой базовый класс который имеет координаты и текстуру, а также несколько виртуальных методов такие как Draw и Update.

Использование элементов управления через клавиатуру и мышь позволяет игроку перемещаться по игровому миру, и не вылетать за его пределы

Таймер отсчитывает игровое время, и от его показания, спавнятся определённые мобы через SpawnPoint, который реализован через паттерн проектирования фабрики. В зависимости от прошедшего времени, будут появляться всё более сильные и сильные враги, сделано это для того, чтобы игроку не надоедал монотонный игровой процесс

Немного о фабрике. Фабрика — порождающий шаблон проектирования, предоставляющий подклассам (дочерним классам) интерфейс для создания экземпляров некоторого класса. В момент создания наследники могут определить, какой класс создавать. Иными словами, данный шаблон делегирует создание объектов наследникам родительского класса. Это позволяет использовать в коде программы не конкретные классы, а манипулировать абстрактными объектами на более высоком уровне.

Целью данного паттерна является определение интерфейса для создания объекта, но за подклассами остаётся решение о том, на основании какого класса создавать объект. Фабрика позволяет классу делегировать создание подклассов.

Пример реализации фабрики на языке C#:

using System;

namespace RefactoringGuru.DesignPatterns.FactoryMethod.Conceptual

{

// Класс Создатель объявляет фабричный метод, который должен возвращать

// объект класса Продукт. Подклассы Создателя обычно предоставляют

// реализацию этого метода.

abstract class Creator

{

// Обратите внимание, что Создатель может также обеспечить реализацию

// фабричного метода по умолчанию.

public abstract IProduct FactoryMethod();

// Также заметьте, что, несмотря на название, основная обязанность

// Создателя не заключается в создании продуктов. Обычно он содержит

// некоторую базовую бизнес-логику, которая основана на объектах

// Продуктов, возвращаемых фабричным методом. Подклассы могут косвенно

// изменять эту бизнес-логику, переопределяя фабричный метод и возвращая

// из него другой тип продукта.

public string SomeOperation()

{

// Вызываем фабричный метод, чтобы получить объект-продукт.

var product = FactoryMethod();

// Далее, работаем с этим продуктом.

var result = "Creator: The same creator's code has just worked with "

+ product.Operation();

return result;

}

}

// Конкретные Создатели переопределяют фабричный метод для того, чтобы

// изменить тип результирующего продукта.

class ConcreteCreator1 : Creator

{

// Обратите внимание, что сигнатура метода по-прежнему использует тип

// абстрактного продукта, хотя фактически из метода возвращается

// конкретный продукт. Таким образом, Создатель может оставаться

// независимым от конкретных классов продуктов.

public override IProduct FactoryMethod()

{

return new ConcreteProduct1();

}

}

class ConcreteCreator2 : Creator

{

public override IProduct FactoryMethod()

{

return new ConcreteProduct2();

}

}

// Интерфейс Продукта объявляет операции, которые должны выполнять все

// конкретные продукты.

public interface IProduct

{

string Operation();

}

// Конкретные Продукты предоставляют различные реализации интерфейса

// Продукта.

class ConcreteProduct1 : IProduct

{

public string Operation()

{

return "{Result of ConcreteProduct1}";

}

}

class ConcreteProduct2 : IProduct

{

public string Operation()

{

return "{Result of ConcreteProduct2}";

}

}

class Client

{

public void Main()

{

Console.WriteLine("App: Launched with the ConcreteCreator1.");

ClientCode(new ConcreteCreator1());

Console.WriteLine("");

Console.WriteLine("App: Launched with the ConcreteCreator2.");

ClientCode(new ConcreteCreator2());

}

// Клиентский код работает с экземпляром конкретного создателя, хотя и

// через его базовый интерфейс. Пока клиент продолжает работать с

// создателем через базовый интерфейс, вы можете передать ему любой

// подкласс создателя.

public void ClientCode(Creator creator)

{

// ...

Console.WriteLine("Client: I'm not aware of the creator's class," +

"but it still works.\n" + creator.SomeOperation());

// ...

}

}

class Program

{

static void Main(string[] args)

{

new Client().Main();

}

}

}

Результат работы программы:

App: Launched with the ConcreteCreator1.

Client: I'm not aware of the creator's class, but it still works.

Creator: The same creator's code has just worked with {Result of ConcreteProduct1}

App: Launched with the ConcreteCreator2.

Client: I'm not aware of the creator's class, but it still works.

Creator: The same creator's code has just worked with {Result of ConcreteProduct2}

# Программная реализация

Базовый класс Basic2d:

#region Includes

using System;

using System.Collections.Generic;

using System.Linq;

using System.Xml.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

using Microsoft.Xna.Framework.Media;

using Microsoft.Xna.Framework.Audio;

using Microsoft.Xna.Framework.Content;

#endregion

namespace MyGame

{

class Basic2d

{

public Vector2 pos, dims;

public Texture2D myModel;

public Basic2d(string PATH, Vector2 POS, Vector2 DIMS)

{

pos = POS;

dims = DIMS;

myModel = Globals.content.Load<Texture2D>(PATH);

}

public virtual void Update(Vector2 OFFSET)

{

}

public virtual void Update(Vector2 OFFSET, Player ENEMY)

{

}

public virtual bool Hover(Vector2 OFFSET)

{

return HoverImg(OFFSET);

}

public virtual bool HoverImg(Vector2 OFFSET)

{

Vector2 mousePos = new Vector2(Globals.mouse.GetNewMousePos().X, Globals.mouse.GetNewMousePos().Y);

if (mousePos.X >= (pos.X + OFFSET.X) - dims.X/2 && mousePos.X >= (pos.X + OFFSET.X) + dims.X / 2 &&

mousePos.Y >= (pos.Y + OFFSET.Y) - dims.Y / 2 && mousePos.Y >= (pos.Y + OFFSET.Y) + dims.Y / 2)

{

return true;

}

return false;

}

public virtual void Draw()

{

if(myModel != null)

{

Globals.spriteBatch.Draw(myModel, new Rectangle((int)(pos.X), (int)(pos.Y), (int)(dims.X), (int)(dims.Y)), null, Color.White, 0.0f, new Vector2(myModel.Bounds.Width / 2, myModel.Bounds.Height / 2), new SpriteEffects(), 0);

}

}

public virtual void Draw(Vector2 OFFSET)

{

if (myModel != null)

{

Globals.spriteBatch.Draw(myModel, new Rectangle((int)(pos.X + OFFSET.X), (int)(pos.Y + OFFSET.Y), (int)(dims.X), (int)(dims.Y)), null, Color.White, 0.0f, new Vector2(myModel.Bounds.Width/2, myModel.Bounds.Height / 2), new SpriteEffects(), 0);

}

}

public virtual void Draw(Vector2 OFFSET, Vector2 ORIGIN, Color COLOR)

{

if (myModel != null)

{

Globals.spriteBatch.Draw(myModel, new Rectangle((int)(pos.X + OFFSET.X), (int)(pos.Y + OFFSET.Y), (int)(dims.X), (int)(dims.Y)), null, COLOR, 0.0f, new Vector2(ORIGIN.X, ORIGIN.Y), new SpriteEffects(), 0);

}

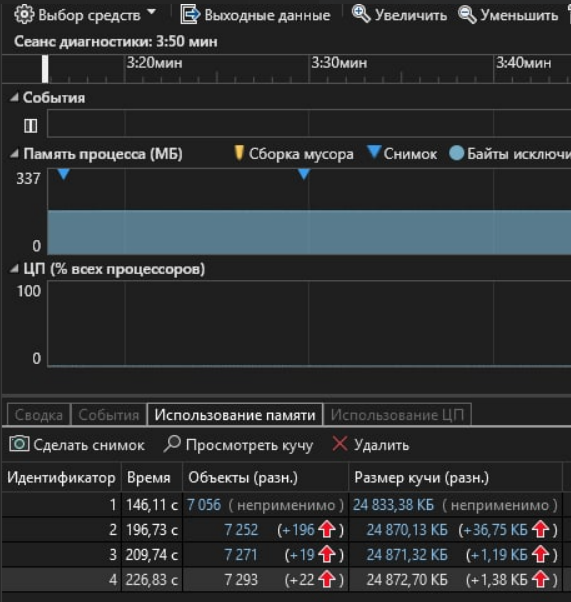
}

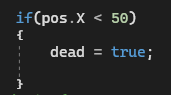
}

}

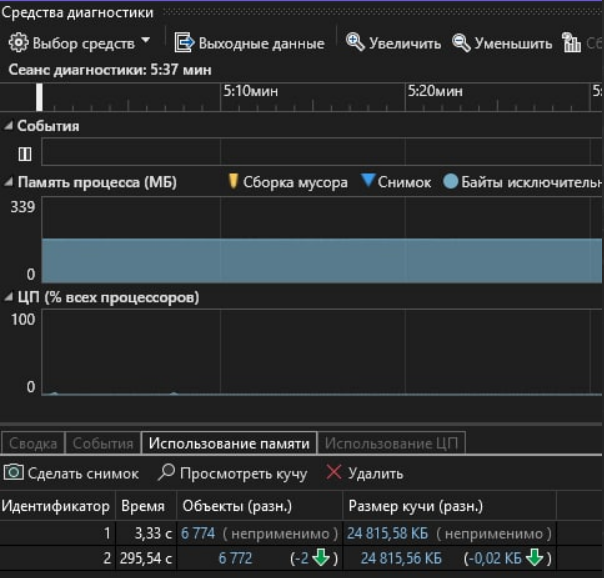
# Тестирование приложения

На этапе разработки, столкнулся с проблемой утечки ресурсов, при долгосрочном запуске приложения, потребность в ресурсах увеличивалась



Проблема была в том, что после выхода вражеского кораблся за пределы экрана, был упущен момент написания следующего кода который эту проблему решает

После добавления, средства диагностики Visual Studio показали следующее



# ЗАКЛЮЧЕНИЕ

В процессе изучения предметной области, была выявлена и описана

классификация однопользовательских игр. В рамках проекта рассмотрены 4

основных направлений однопользовательских игр: экшен, аркада, шутер, файтинг, однако, существует ещё множество различных жанров, но так как они менее распространены или являются помесями, они здесь не рассматриваются.

Также были определены и освоены инструментальные средства, при

помощи которых будет происходит реализация игрового проекта. Основной

и наиболее важной программой для разработки игр является движок, в нашем

случае он был самостоятельно написан при помощи MonoGame. MonoGame использует язык C# и имеет открытый исходный код.

При составлении концептуальной модели было принято решение

создавать игру в жанре аркада. Аркада игры - Игры, в которых игроку приходится действовать быстро, полагаясь в первую очередь на свои рефлексы и реакцию. Исходя из этого были подобраны и созданы игровые

спрайты в программах Microsoft Paint, Photoshop, Shetchbook Autodesk, , с

последующим их внедрением.

# СПИСОК ЛИТЕРАТУРЫ

1. Баженова, И.Ю. Языки программирования: Учебник / И.Ю. Баженова. - М.: Академия, 2018. - 448 c.

2. Гавриков, М.М. Теоретические основы разработки и реализации языков программирования / М.М. Гавриков, А.Н. Иванченко. - М.: КноРус, 2018. - 207 c.

3. Довек, Ж. Введение в теорию языков программирования / Ж. Довек, Ж.-Ж. Леви. - М.: ДМК, 2016. - 134 c.

4. Дорогов, В.Г. Основы программирования на языке С#: Учебное пособие / В.Г. Дорогов, Е.Г. Дорогова. - М.: Форум, 2015. - 320 c.

5. Касторнова, В.А. Структуры данных и алгоритмы их обработки на языке программирования Паскаль: Учебное пособие / В.А. Касторнова. - СПб.: BHV, 2016. - 304 c.

6. Керниган, Б. Язык программирования C#. / Б. Керниган, Д.М. Ритчи. - М.: Вильямс, 2016. - 288 c.

7. Кузин, А.В. Основы программирования на языке Objective-C для iOS: Учебное пособие / А.В. Кузин, Е.В. Чумакова. - М.: Инфра-М, 2018. - 384 c.

8. Липпман, С. Язык программирования С#. Базовый курс / С. Липпман. - М.: Вильямс И.Д., 2018. - 1120 c.

9. Макаровских, Т.А. Языки и методы программирования: Путеводитель по языку С#/ Т.А. Макаровских, А.В. Панюков. - М.: Ленанд, 2018. - 216 c.

10. Макаровских, Т.А. Языки и методы программирования: Создание простых GUI-приложений с помощью C# / Т.А. Макаровских, А.В. Панюков. - М.: Ленанд, 2018. - 144 c.

11. Орлов, С. Теория и практика языков программирования: Учебник / С. Орлов. - СПб.: Питер, 2017. - 224 c.

12. Орлов, С. Теория и практика языков программирования. Учебник для вузов Стандарт третьего поколения / С. Орлов. - СПб.: Питер, 2017. - 688 c.

13. Солдатенко, И.С. Практическое введение в язык программирования С#: Учебное пособие / И.С. Солдатенко, И.В. Попов. - СПб.: Лань, 2018. - 132 c.

14. Троелсен, Э. Язык программирования C# 7 и платформа .NET и NET Core / Э. Троелсен. - М.: Вильямс И.Д., 2018. - 1328 c.