Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Курсовой проект по дисциплине:

«Программирование»

Пояснительная записка к курсовой работе

Тема работы:

«Игра Space Invaders»

Исполнитель

студент гр. 653502 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Галкин И.В.

(подпись дата)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Козуб В.Н.

(подпись дата)

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

(оценка)

Минск

2017 год

**Содержание**

Введение3

**1.** Постановка задачи4

**2.** Описание процесса игры5

**2.1.** MainMenu5  
**2.2.** Game6

**2.2.1.** Начало игры6

**2.2.2.** Конец игры7

**3.** Разработка программы8

**3.1.** Подготовка к созданию объекта8

**3.2.** Объекты8

**3.2.1.** Неподвижные объекты8

**3.2.2.** Подвижные объекты9

**3.2.3.** Пустые объекты9

**3.3.** Скрипты10

**3.3.1.** Главное меню10

**3.3.2.** Переход на игровое поле10

**3.3.3.** Игровой процесс.11

**3.3.4.** Завершение игры12

**4.** Планы по улучшению игры13  
**5.** Руководство пользователя13  
Заключение14

Список использованных источников15

# **Введение**

С наступлением Информационной эры появилось множество различных способов развлечений. Одним из самых популярных способов являются компьютерные игры. Компьютерные игры стремительно развиваются день за днем, улучшая систему искусственного интеллекта день за днем. Успех игр заключается в том, что они могут принести человеку те чувства, которые им было бы тяжело получить в реальной жизни.

Причиной для написания курсовой работы на тему компьютерная игра является желание изучить и попробовать себя в искусстве создания игр. Для написание этого курсового проекта был использован игровой движок Unity и язык программирования С#. Unity предоставляет огромное количество инструментов для создания игровых проектов. В нем очень удобно работать с моделями, тестировать и настраивать приложение вплоть до малейших деталей. А язык программирования C# является наиболее эффективным для написания проектов такого типа из-за своей универсальности.

Данная игра относится к не ролевым, а также к жанру «Shoot ‘em up», где основной смысл игры: защитить себя от врагов и не дать им подойти к себе. В 70-90 года прошлого века игры такого жанра имели колоссальный успех из-за своей простоты и затягивания. Быстрота реакции является ключевым моментом в достижении высоких результатов в играх такого жанра.

Целью данной курсовой работы является:

* Изучение основ создания игр.
* Изучение игрового движка Unity.
* Улучшение навыков по созданию приложений на C#.
* Закрепление уже полученных знаний по C#

**1.Постановка задачи**

Любая игра должна содержать:

* Основную задачу.
* Главное меню.
* Модели для всех персонажей.
* Декорации.
* Игровой интерфейс.
* Музыкальное сопровождение.

Игра должна быть понятна потребителю с самого запуска программы. Для этого требуется создать максимально понятный и дружелюбный интерфейс.  
Также необходимо, чтобы потребитель мог в любой момент остановить игру и выйти из программы.

Следует предусмотреть защиту от случайных нажатий и создать игру с управлением, которое используется, как стандартное, во всех уже созданных играх.

Играм такого жанра свойственно:

* Бесконечный цикл игры.
* Несколько жизней.
* Игровой счет и рекорды.
* Нет сюжета.

Данные требования должны быть соблюдены при создании подобных проектов.

# **2. Описание процесса игры**

При работе в Unity мы работает со сценами. В данном проекте присутствуют 2 сцены:

* MainMenu – главное меню.
* Game – игровое поле.

**2.1. MainMenu** Как только мы запускаем программу, мы попадаем в главное меню (Рисунок 1). На главном меню располагается 2 кнопки. Кнопка Start переносит пользователя на следующую сцену и начинает игру. Кнопка Quit позволяет выйти из игры.   
  
 Данная сцена используется только 1 раз. Она необходима для того, чтобы пользователь при входе в игру сразу же не попадал на игровое поле. Следует обратить внимание на то, что Unity при переходе на следующую сцену удаляет прошлую, если ее не сохранить. Поэтому данная сцена не будет засорять память и просто удалится.



**Рисунок 1**  — главное меню.

**2.2. Game  
  
2.2.1. Начало игры**

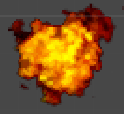
 После нажатия кнопки Start в главном меню мы попадаем на сцену Game (Рисунок 2).  
 В левом нижнем углу находится счетчик оставшегося здоровья, а в правом нижнем углу находится счетчик уничтоженных противников.   
 **Рисунок 2** — игровое поле

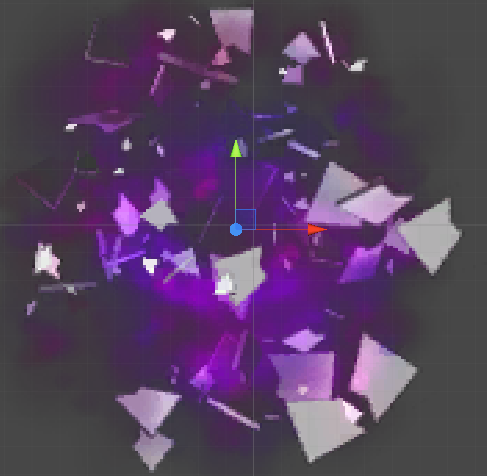
 Игрок появляется в нижней части экрана за синий космический корабль (Рисунок 3).

**Рисунок 3**  — модель игрока.

 Противники начинают появляться в верхней части экрана в виде красных космичесиких кораблей(Рисунок 4).   
Они двигаются в горизонтальном направлении , а после того как доходят до конца экрана спускаются вниз. Противники будут постоянно стрелять тем самым не позволяя игроку стоять на месте.   
  
**Рисунок 4** — модель противника.

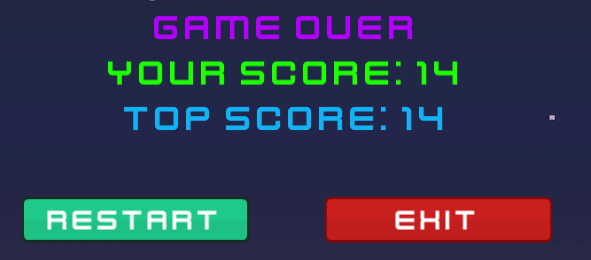
Игрок имеет бесконечное количество патронов. Уничтожая противника , игрок получает +1 очко. Следует отметить , что игра бесконечная т.к. количество противников неограничено.  
 В игре предусмотренно так, что у игрока всего 4 жизни. При попадании противника по игроку отнимается -20 здоровья.



 При уничтожении противника проигрывается специальная анимация(Рисунок 5) вместе с соответсвующим звуком.

**Рисунок 5** — взрыв противника.

То же самое происходит и при проигрыше игрока, но немного в другом внешнем виде(Рисунок 6).  
  
  
  
**Рисунок 6**  — взрыв игрока.  
**2.2.2. Конец игры** Когда у игрока заканчивается здоровье перед ним появляется результат игры (Рисунок 7).



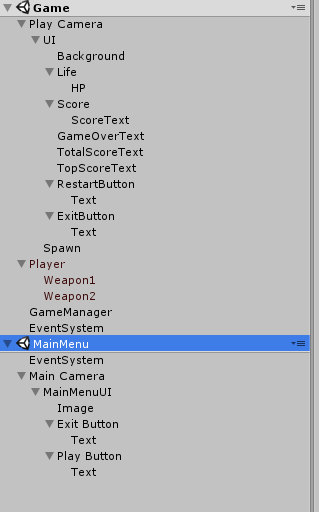
**Рисунок 7**  — конец игры.  
 Перед игроком выводится сообщение о набранном им счете в текущей игре и сообщение о максимально набранном счете за данную сессию игры.  
Игроку предлагается либо сыграть еще раз, либо выйти из игры.   
Также закончить игру можно при помощи нажатия клавиши Escape.

**3. Разработка программы  
  
3.1. Подготовка к созданию проекта**

Прежде чем начинать любой проект на Unity необходимо правильно подготовиться. Программист должен иметь четкое представление, что он собирается делать.

* Для начала следует продумать концепцию игры, придумать правила по которой игра будет создаваться.
* Необходимо продумать все классы, которые будут использоваться в проекте (хорошо помогают схемы, нарисованные на бумаге).
* Подготовить наборы спрайтов, звуковых эффектов, фоновых изображений, фоновой музыки.

После этого можно начинать делать проект.

 **3.2. Объекты**

Следующим этапом в разработке проекта будет создание объектов игры и добавление к ним свойств.

На Рисунке 1 представлена вся иерархия объектов на двух сценах, которые будут находиться на сценах при первом же запуске.  
 В целом, все их можно поделить на 3 группы:

1. Неподвижные объекты.
2. Подвижные объекты.
3. Пустые объекты.

**Рисунок 1**

**3.2.1. Неподвижные объекты** Неподвижные объекты выполняют исключительно визуальные и переключательные функции.

К ним относятся:

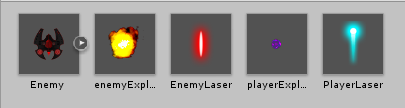
* Панели со здоровьем и текущим счетом (Life и Score).
* Фоновые изображения (Background и Image).
* Кнопки рестарта, выхода и начала игры (RestartButton, ExitButton, PlayButton).
* Сообщения выводимые в конце игры (GameOverText, TotalScoreText, TopScoreText).

**3.2.2. Подвижные объекты** Особенность объектов этого типа в том, что они могут передвигаются по всему полю при помощи скриптов.

К подвижным объектам относятся:

* Модель игрока
* Модель противника
* Атаки моделей
* Анимации разрушения моделей

Стоит заметить, что 3 последних объекта не находятся в иерархии. Они сохранены в форме префабов – уже готовых шаблонах игровых объектов (Рисунок 2). Префабы удобно использовать в случае, когда необходимо генерировать большое количество GameObject-ов одного типа.



**Рисунок 2** — префабы, используемые в игре.

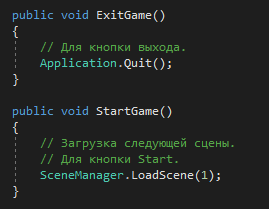
**3.2.3. Пустые объекты.** Данный тип объектов не отображается на экране, но они играют ключевые роли в проекте.

В данной игре это:

* Weapons – невидимые точки, из которых происходит генерация атак моделей
* Spawn – невидимая точка, через которую работает скрипт по генерированию противников.
* GameManager – главный объект программы, который управляет началом и концом игры.

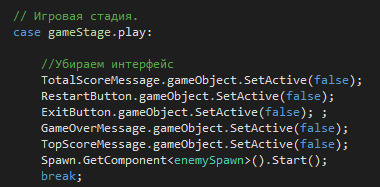
**3.3. Скрипты** Для того, чтобы игра получилась полноценной нужно написать правила, по которым объекты могут существовать и выполнять действия. Для этого программист пишет скрипты с методами инициализации и возможностями объектов.   
  
 Всего в проекте 11 скриптов. Каждый скрипт был написан для определенного типа объекта. А именно:

* MainMenu – скрипт для главного меню.
* ButtonsManager – скрипт для кнопок проекта.
* destroyObject – скрипт обрабатывающий уничтожение объекта.
* enemyScript – скрипт для моделей противников.
* enemyShot – скрипт для атак противников.
* enemySpawn – скрипт для генератора противников.
* enemyWeapon – скрипт для точек генерации атак противников.
* playerScript – скрипт для модели игрока.
* playerShot – скрипт для атак игрока.
* scoreScript – скрипт для подсчета игрового счета.
* GameManager – скрипт для управления игрой.

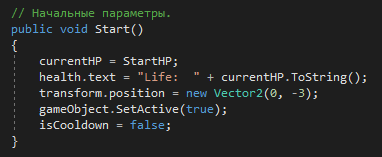
**3.3.1. Главное меню**

В главном меню используется скрипт MainMenu, который позволяет кнопкам выполнять переключательную функцию. Это реализовано при помощи методов StartGame и ExitGame (Рисунок 3).

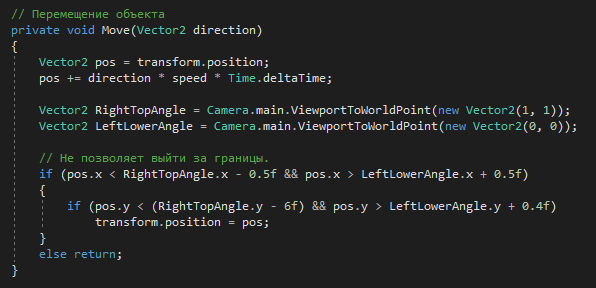
**Рисунок 3** — скрипт MainMenu.

**3.3.2. Переход на игровое поле**

При переходе на игровое поле объект GameManager при помощи своего скрипта моментально переводит игру в режим игры. Это значит, что появляется моделька игрока, обнуляется счетчик очков и начинается генерация противников (Рисунок 4). **Рисунок 4** — переход в режим игры.

 Стоит отметить, что для любого объекта мы можем задать стартовые параметры. Это делается посредством метода Start, который по умолчанию предлагает Unity при написании скрипта к объекту (Рисунок 5). **Рисунок 5** — метод Start для модели игрока.

**3.3.3. Игровой процесс.**

 После того, как появилась модель игрока и начали генерироваться противники, начинает реализовываться метод по перемещению объектов по игровому полю (Рисунок 6).

**Рисунок 6** — метод для перемещения объекта.

Данный метод используется во всех перемещающихся объектах, разница только в правилах, по которым объект может двигаться.   
Алгоритм достаточно прост:

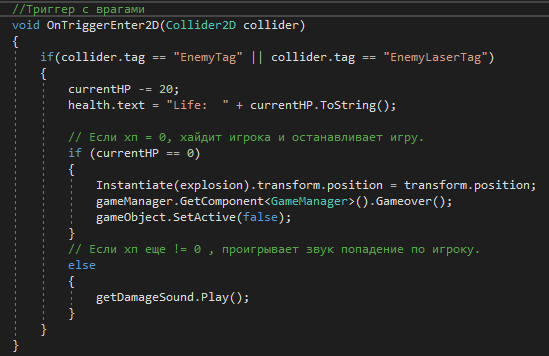
1. Высчитывается текущая позиция объекта.
2. Высчитывается точка, в которую объект должен переместиться
3. Проверяется, выходит ли эта точка за границы поля.
4. Если точка не выходит за границы, то перемещает объект.

Далее начинает реализовываться метод стрельбы моделей. Он проходит через невидимые объекты Weapons, принадлежащие к моделям игрока и противников.

Его алгоритм:

1. Высчитывается текущая позиция модели.
2. Если оружие не на перезарядке, генерирует объект Laser.
3. Перемещает объект Laser на несколько пикселей через некоторый промежуток времени.
4. В случае, если Laser вылетает за поле – удаляет его.

Если же лазер попадает по цели, начинает работать метод OnTriggerEnter2D (Рисунок 7). Этот метод предоставляется Unity, который позволяет написать событие в случае, если какой-нибудь объект соприкасается с коллайдерным квадратом объекта, который можно настроить в свойствах объекта.



**Рисунок 7** — Пример метода OnTriggerEnter2D для модели игрока.

На рисунке видно, что если противник попадает по игроку 1 раз, то у игрока отнимается 20 очков здоровья. Если оставшееся здоровье больше 0, то проигрывается звук попадания по игроку. А если здоровье становится равно 0, то происходит анимация взрыва модели игрока, а сама модель переходит в режим неактивности и пропадает со сцены. Это нужно для того, чтобы при рестарте игры мы могли добавить модель игрока обратно на поле.  
  
 Строка gameManager.GetComponent<GameManager>().Gameover(); переводит игру в режим завершения игры, о чем пойдет речь в следующем пункте.

**3.3.4. Завершение игры.**

После того, как у игрока закончились очки здоровья, скрипт GameManager переводит игру в режим GameOver. Это означает, что с поля убирается модель игрока, а также уничтожаются все оставшиеся модели противника.

На экран выводятся сообщение об окончании игры и сообщения о текущем счете и максимально набранном счете за сессию игры. Также выводятся кнопки-переключатели.

**4.Планы по улучшению игры**

* Улучшение ИИ.
* Добавление разных типов противников.
* Добавление боссов.
* Добавление различных моделей игрока.
* Добавление подбираемых улучшений и очков здоровья.
* Добавление новых локаций.
* Переход игры на смартфоны.

**5. Руководство пользователя**

Для запуска приложения необходимо открыть файл SpaceInvaders.exe.  
После запуска перед пользователем появится главное меню игры.  
Управление:

Выбор: ЛКМ (Рисунок1). Передвижение: WASD (Рисунок 2).

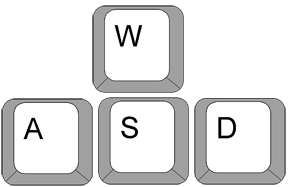
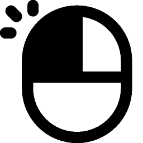
****

Рисунок 1 Рисунок 2

Стрельба: Space (Рисунок 3). Выход из игры: Escape (Рисунок 4)

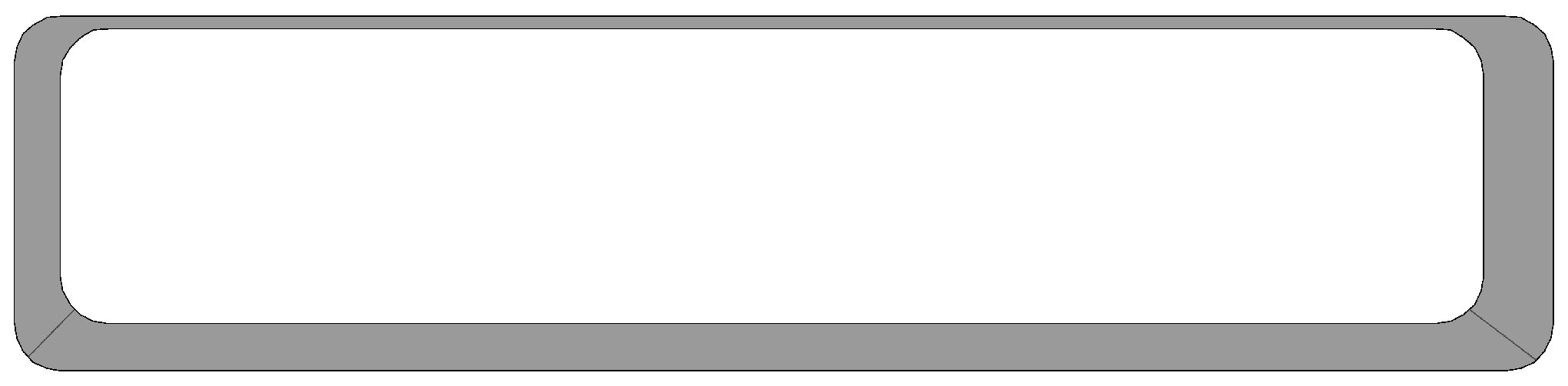


Рисунок 3. Рисунок 4

**Заключение**

По результатам курсовой работы было создано игровое приложение по мотивам оригинальной игры Space Invaders.Во время разработки программы были учтены все требования, предъявляемые к игровым приложениям такого рода. В результате работы были изучены основы игрового движка Unity, закреплены знания по языку программирования C#, были изучены основные принципы создания игровых приложений.   
 Во время разработки данного проекта возникли некоторые идеи о будущем расширении игровых возможностей данной игры. Развитие проекта не остановится на месте.

**Список использованных источников:**

* Руководство по С# - Часть 1 [электронный ресурс] – Режим доступа: <https://professorweb.ru/my/csharp/charp_theory/level1>
* Руководство по С# - Часть 2 [электронный ресурс] – Режим доступа: <https://professorweb.ru/my/csharp/charp_theory/level2>
* Обучающие материалы по Unity [электронный ресурс] – Режим доступа: <https://unity3d.com/ru/learn/tutorials>
* Документация по Unity [электронный ресурс] – Режим доступа: <https://docs.unity3d.com/ru/current/Manual>