Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики и технологии программирования

Дисциплина: Объектно-ориентированное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе  
на тему

Игра «XSquad»

Студент

Руководитель

А. Е. Головачук

М. Г. Рогов

МИНСК 2021

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

ОСНОВНАЯ ЧАСТЬ 7

1 Общие положения 7

2 Алгоритм реализации проекта 8

3 Игровой движок (Game Engine) 9

4 Организация и ход игровой программы 10

4.1 Аспекты для разработки игровой программы 10

4.2 Тип и правило игровой программы 10

4.3 Игровые юниты, виды оружия и укрытий 11

4.4 Виртуальный мир и генерация карты 13

4.5 Пользовательский интерфейс (UI) 15

4.6 Анимации юнитов и реализация передвижения юнитов

по виртуальному миру 16

4.7 Подход, основанный на реакциях 17

4.8 Система здоровья и урона 17

4.9 Экономика 18

4.10 Тестирование 18

5 Код игровой программы 19

ЗАКЛЮЧЕНИЕ 20

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21

## ВВЕДЕНИЕ

Цель курсовой работы – разработать игровой программный продукт под названием «XSquad» в жанре «Пошаговая Тактика», изучить объектно-ориентированное программирование, рассмотреть паттерны проектирования и применить их на практике.

Объектно-ориентированное программирование (ООП) – это подход, при котором вся программа рассматривается как набор взаимодействующих друг с другом объектов. Объект – это сущность, имеющая свои атрибуты и поведения. А для создания объекта существует шаблон, который является классом.

Основными принципами ООП являются:

* **инкапсуляция**– это свойство системы, позволяющее объединить данные и методы и скрыть детали реализации от пользователя;
* **наследование**– это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс– потомком, наследником или производным классом;
* **полиморфизм**– это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта;
* **абстрагирование**– это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция – это набор всех таких характеристик. [1]

Паттерны проектирования– это шаблоны которые описывают типичные способы решения часто встречающихся проблем при проектировании программ.

Пошаговая тактика (англ. Tactical turn-based, TTS) – это жанр стратегической компьютерной игры, в котором разделённый на отдельные ходы игровой процесс симулирует боевые действия малого масштаба, используя при этом элементы военного оперативного искусства и военной тактики.

Для создания игрового продукта использовался язык C# – это объекно-ориентированный язык программирования, действительно интересный инструмент, достойный внимания. Он уверенно занимает высокие позиции в рейтингах востребованных языков программирования на рынке труда. Разработан в 1998 – 2001 годах группой инженеров компании Microsoft и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft/NET Framework. Впоследствии был стандартизирован как ЕСМА-334 и ISO/IEC 23270. [2]

Инструментарий C# позволяет решать широкий круг задач, язык действительно очень мощный и универсальный. На нем разрабатывают:

* приложения для WEB;
* приложения платформ Андроид или iOS;
* программы для Windows;
* различные игровые программы.

Основанием для выполнения курсовой работы послужили требования, изложенные в учебном плане 2-го курса факультета ИиТП БГУИР на II семестр 2021 года.

## ОСНОВНАЯ ЧАСТЬ

1. **Общие положения**
   1. Для разработки игры используется программа Unity 3D и движок Unity.
   2. За концепцию взята механика игр жанра пошаговой тактики.
   3. Для осуществления цели курсовой работы применяется язык программирования C# в игровой сфере.
   4. В ходе написания игрового кода изучены основы ООП, рассмотрены некоторые паттерны проектирования и применены в игровом продукте.
   5. Реализована начальная кодовая база для развития данного проекта:

* пошаговая тактика;
* движение персонажей (юнитов) по определенной траектории;
* взаимодействие игровых юнитов;
* система навигации юнитов;
* система укрытий;
* генерация карт;
* создание и смерть игровых юнитов;
* смена команд;
* рельеф;
* вариации оружий и укрытий;
* «HUD»;
* загрузочные меню;
* меню паузы;
* режим редактирования.
  1. Разработка курсовой работы с целью создания игровой программы

проведена на базе факультета ИиТП БГУИР.

1. **Алгоритм реализации проекта**

|  |
| --- |
| Выбор игрового движка |
| ↓ |
| Определение игровых юнитов, видов оружия и укрытий |
| ↓ |
| Создание виртуального мира |
| ↓ |
| Создание игровых интерфейсов |
| ↓ |
| Программирование анимации юнитов |
| ↓ |
| Настройка взаимодействия юнитов |
| ↓ |
| Реализация передвижения юнитов  по виртуальному миру |
| ↓ |
| Система здоровья и урона |
| ↓ |
| Настройка меню (меню паузы, редактирования) |
| ↓ |
| Расчет экономики |
| ↓ |
| Тестирование |
| ↓ |
| Написание пояснительной записки |

1. **Игровой движок (Game Engine)**

Термин «игровой движок» возник в середине 1990-х.

Игровой движок ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) game engine) – базовое [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [компьютерной игры](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%B8%D0%B3%D1%80%D0%B0). Разделение игры и игрового движка часто расплывчато. Но в общем случае термин «игровой движок» применяется для того программного обеспечения, которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений. [3]

Любой игровой движок предоставляет множество функциональных возможностей, которые задействуются в различных играх.

Unity – профессиональный игровой движок, который используется при создании видеоигр для различных платформ. Он предлагает моделирование физических сред, карты нормалей, преграждение окружающего света в экранном пространстве (Screen Space Ambient Occlusion, SSAO), динамические тени… список можно продолжать долго. [4]

Выбор игрового движка Unity для реализации данного проекта сделан, основываясь на двух факторах:

* низкий порог вхождения;
* чрезвычайно производительный визуальный рабочий процесс.

Визуальный рабочий процесс представляет собой достаточно уникальную вещь, выделяющую данный инструмент из большинства других сред разработки игр. В то время как остальные инструменты разработки игр зачастую представляют собой мешанину разрозненных частей, которые требуется контролировать, или, возможно, библиотеку, для работы с которой нужно настраивать собственную интегрированную среду разработки (Integrated Development Environment, IDE), цепочку сборки и прочее в этом роде, рабочий процесс в Unity привязан к тщательно продуманному визуальному редактору. В этом редакторе компонуются сцены будущей игры, связывая игровые ресурсы и код в интерактивные объекты. Именно он позволяет быстро и рационально создавать профессиональные игры, обеспечивая невиданную продуктивность труда разработчиков и предоставляя в их распоряжение исчерпывающий перечень самых современных технологий в области видеоигр. [4]

1. **Организация и ход игровой программы**
2. **Аспекты для разработки игровой программы**

Для разработки игровой программы используют такие аспекты, как:

* физика – использование игрового движка, в том числе симуляция «физики» (физическая модель) – движение объектов и другие виды;
* графика – управление использованием графического содержимого и памяти компьютера; разработка графического движка, интеграция трехмерных моделей, текстур для работы по движку физики;
* геймплей – реализация различных игровых механик и особенностей;
* пользовательский интерфейс – программирование пунктов меню и визуального интерфейса;
* устройства ввода – настройка работы с устройствами ввода, таких как клавиатура и мышь.

В данной курсовой работе не реализованы такие аспекты как: сценарии, сетевые коммуникации, звуковое сопровождение, инструменты разработки игр (производство вспомогательного специализированного программного обеспечения, сопровождающих разработку игры).

1. **Тип и правило игровой программы**

Для контроля поведения в создании компьютерных игр вытекают два типа направлений. Первый тип реализует принцип: проектирование с явным контролем (explicit design), контролируя в игре все до последней мелочи. В играх второго типа интересные игровые возможности порождаются в результате взаимодействия системы искусственного интеллекта и игровой среды. Ключевой особенностью таких игр является отсутствие жестко прописанного сценария. Суть таких игр сводиться к взаимодействию в игровом мире всех персонажей (как представляющих живых игроков, так и управляемых компьютером). Такой принцип проектирования называется проектированием с неявным контролем. [5]

Игровая программа, представленная в данном курсовом проекте, построена на втором принципе.

Концепция представляет собой жанр игр пошаговой тактики, примером которой могут служить «шахматы». Идея игры «XSquad» заключается в том, что игрок с помощью юнитов должен «зачистить» игровую область от врагов. Для этого он может приминать любые средства: оружие, защитные сооружения и способности юнита. В процессе игры он имеет право выбирать и менять тактику поведения, то есть игроку дается полная свобода действий. После завершения уровня, игрок вправе подготовиться к следующему уровню, при этом он имеет право оставить или поменять команду. Миссия игрока будет провалена, когда у него не остается юнитов.

1. **Игровые юниты, виды оружия и укрытий**

В данной игровой программе используются:

* игровые управляемые юниты (рис. 4.3.1);

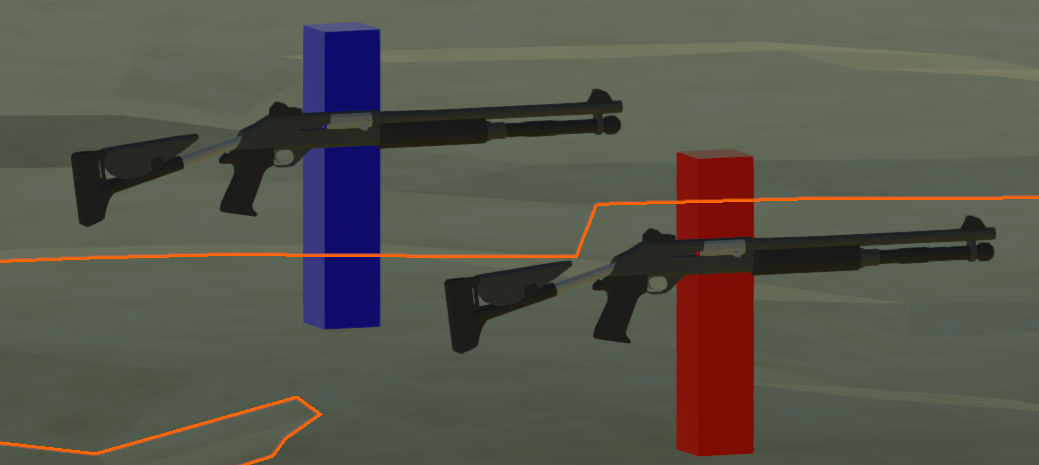


Рисунок 4.3.1 – Синие и красные юниты

* следующие виды оружия (рис. 4.3.2);



Рисунок 4.3.2 – Виды оружия

* следующие виды укрытий (рис. 4.3.3) ;



Рисунок 4.3.3 – Виды укрытий

1. **Виртуальный мир и генерация карты**

Виртуальный мир – искусственно созданный мир, построенный посредством программирования, на основе компьютерных технологий.

В данной игровом продукте представлен гибкий интерфейс разработки собственных карт (редактор карт). Он позволяет выбрать размер карты и расставить на ней юнитов. Карта представляет собой множество шестиугольных ячеек. Параметры каждой ячейки карты, а именно: тип почвы (песок, трава, камень, снег, грязь), высота (равнины, холмы, горы), объем воды, препятствие – могут быть изменены.

Для уровней был спроектирован отдельный блок кода – генератор карты с гибкой настройкой, возможностями которого являются изменение соотношения суши/воды и высоты рельефа и эрозия почвы. Данное нововведение обеспечивает реиграбельность. Так же в данном блоке кода реализуется генерация карт с уникальным ключом, при вводе которого происходит восстановление первоначального вида карты.

Разные виды генераций карт представлены на рисунках 4.4.1 – 4.4.4.

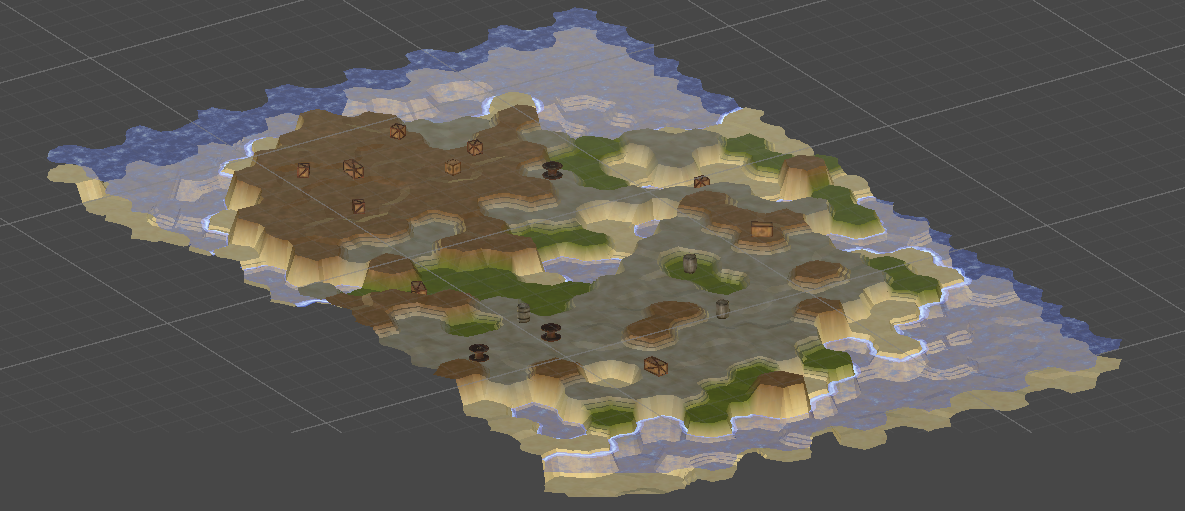


Рисунок 4.4.1 – Сгенерированная маленькая карта

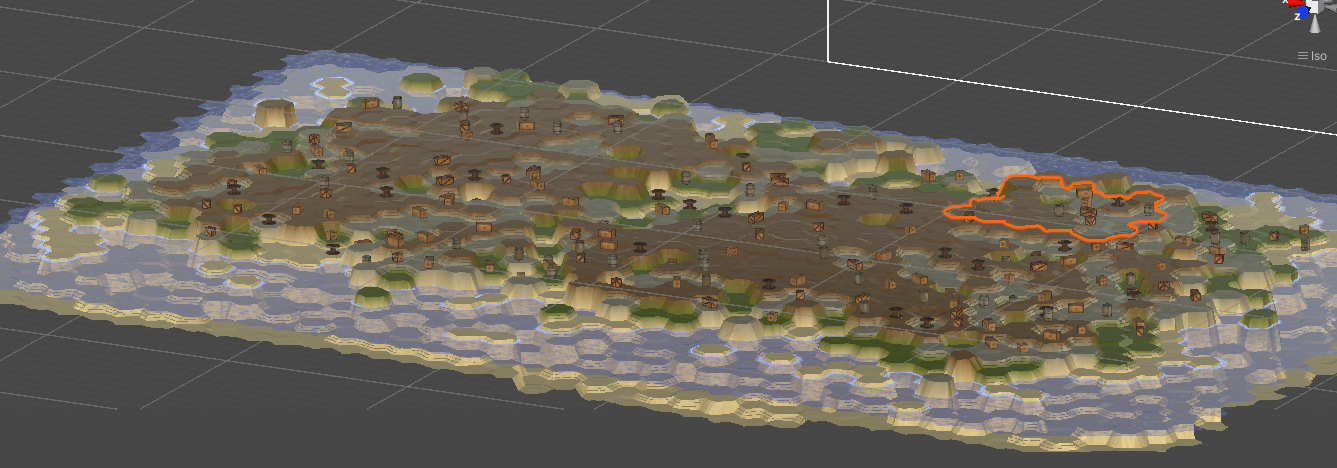


Рисунок 4.4.2 – Сгенерированная средняя карта с большим числом объектов

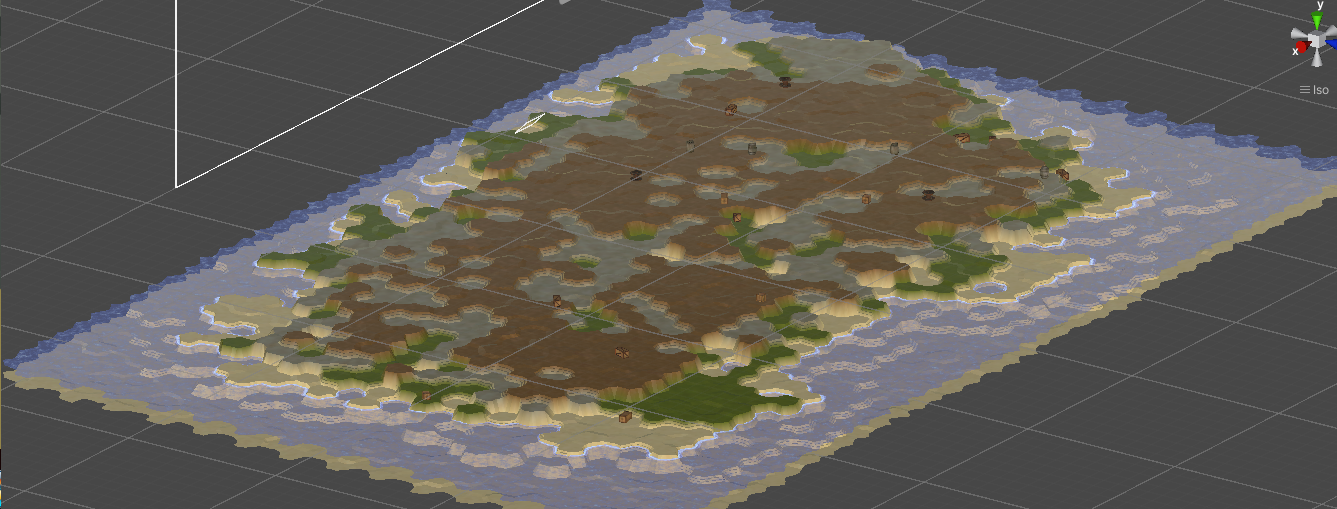


Рисунок 4.4.3 – Сгенерированная средняя карта с малым числом объектов

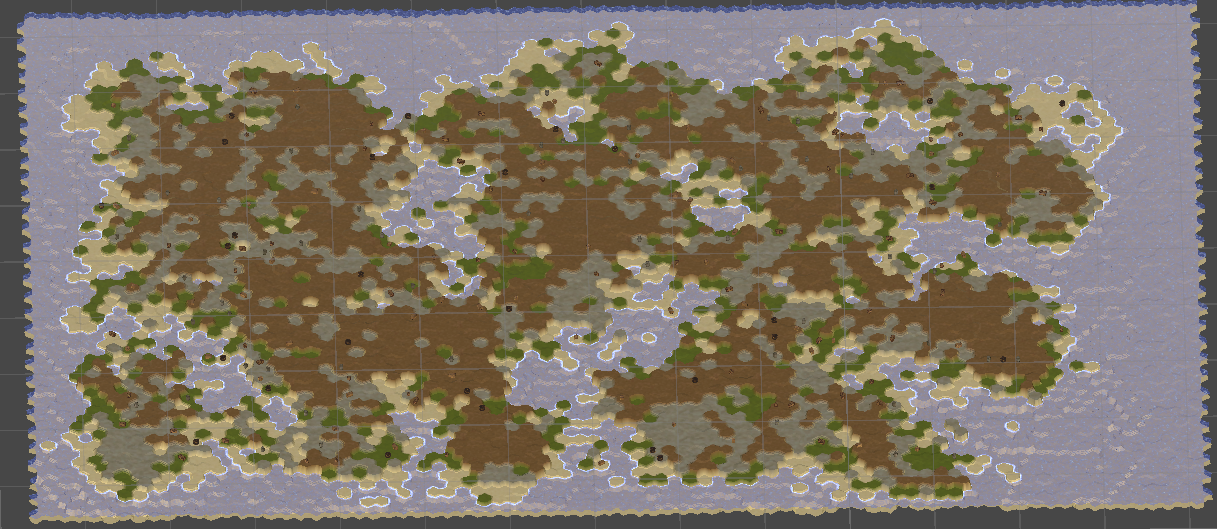


Рисунок 4.4.4 – Сгенерированная большая карта

1. **Пользовательский интерфейс (UI)**

Раскрывая термин «интерфейс» узкоспециализировано, можно сказать, что он объединяет устройства ввода и вывода и программное обеспечение, которое обслуживает их. В более широком смысле интерфейс (рис. 4.5.1) включает в себя все, что помогает пользователю взаимодействовать с продуктом. [6]



Рисунок 4.5.1 – «HUD»

1. **Анимации юнитов и реализация передвижения юнитов по виртуальному миру**

Анимация и передвижения юнитов основана на перемещении. Для перемещения по прямолинейным траекториям устанавливаются координаты между началом и концом прямой. Используя функции Vector3.MoveTowards и DotWeen, в которых применяются формулы скалярной разности, программа вычисляет координаты точек, на которые требуется переместить юнита. [7]

Для поиска пути (рис. 4.6.1) и перемещения юнита по ландшафту (рис. 4.6.2) в данном игровом продукте был реализован алгоритм «А\*».



Рисунок 4.6.1 – Поиск пути



Рисунок 4.6.2 – Перемещение юнита

1. **Подход, основанный на реакциях**

Подход, основанный на реакциях, идеален для компьютерных игр, поскольку он одновременно и прост, и эффективен. Кроме того, предсказуемость поведения персонажей, получаемого при подобном подходе, это также еще одно достоинство, значение которого для проектирования и тестирования трудно оценить. [5]

В данном игровом продукте реализован метод обнаружения вражеских юнитов с выводом их на «HUD» (рис. 4.7.1). Для взаимодействия с ними могут быть использованы любые доступные способы, представленные на рисунке 4.7.2.



Рисунок 4.7.1 – Обнаруженные вражеские юниты



Рисунок 4.7.2 – Способности юнита

1. **Система здоровья и урона**

Очки здоровья, либо очки жизни – величина в [ролевых](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) и [компьютерных играх](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%D1%8B%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B), определяющая максимальное количество урона, которое может получить объект. Если текущее количество очков здоровья достигает нуля, то объект погибает, либо теряет сознание. Таким объектом могут быть [управляемый игроком персонаж](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%B6), [противники](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B)), либо [боссы](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D1%81%D1%81_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B)). Также очками здоровья в играх могут обладать разрушаемые части игровой карты, предметы, транспортные средства, боевые машины и их отдельные части. [8]

1. **Экономика**

Глобально экономики делятся на:

* активные – игрок может сам тратить получаемые ресурсы;
* пассивные – игрок не может влиять на трату ресурсов;
* статические – игрок не влияет на скорость получения ресурса;
* динамические – игрок может изменять скорость получения ресурса;
* скрытые – игрок не видит самого ресурса;
* прозрачные – игроку видим ресурс;
* линейные – со временем игры объёмы ресурса не меняются;
* инфляционные – объёмы и масштабы изменяются со временем.

Экономика представленного проекта – активная, динамическая, скрытая, линейная.

В игровой программе реализуется игровой опыт. Он используется для улучшения юнитов и добавляется при уничтожении вражеских юнитов.

1. **Тестирование**

Для проверки работы программы используются тестовые сценарии в компонентном, интеграционном и системном тестировании. Тестовые сценарии, как правило, пишутся для проверки компонентов, в которых наиболее высока вероятность появления отказов или вовремя не найденная ошибка может быть дорогостоящей.

Тестирование программного продукта проводится по следующим стратегиям: [тестирование белого ящика](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D0%B8_%D0%B1%D0%B5%D0%BB%D0%BE%D0%B3%D0%BE_%D1%8F%D1%89%D0%B8%D0%BA%D0%B0) и [тестирование чёрного ящика](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D0%B8_%D1%87%D1%91%D1%80%D0%BD%D0%BE%D0%B3%D0%BE_%D1%8F%D1%89%D0%B8%D0%BA%D0%B0).

Стратегия белого ящика представляет компонентное тестирование, при котором проверяются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции работоспособны и устойчивы до определённой степени.

Тестирование чёрного ящика происходит через те же [интерфейсы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), что и пользователь.

При тестировании серого ящика имеется доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется. [9]

1. **Код игровой программы**

Для написания кода программы были использованы функции, с помощью которых юниты могут перемещаться, находить противников, поворачиваться и стрелять. Код игровой программы предоставлен на сайте [https://github.com/DcDrugs/XSquad](https://github.com/DcDrugs/Code18).

## ЗАКЛЮЧЕНИЕ

Итогом курсовой работы стал новый игровой продукт «XSquad», полученный с использованием платформы Unity.

В процессе проведения курсовой работы изучены возможности применения паттернов проектирования в игровой сфере. Актуальность работы состоит в том, что в дополнении к общему курсу изучения объектно-ориентированного проектирования рассмотрены новые возможности реализации архитектуры на практике.

В результате исследования были выявлены следующие недостатки полученного программного обеспечения:

* + 1. примитивность игрового продукта;
    2. неуравновешенный игровой баланс;
    3. несбалансированная экономика.

В результате созданный игровой продукт по первому пункту требует большего количества времени и средств, по вторым двум пунктам необходимо небольшое улучшение, на которое потребуется минимум изменений исходного кода программы.

Достоинствами данного продукта является следующее:

1. следование юнитов по назначенному маршруту;
2. нелинейность действий;
3. анимация юнитов;
4. разнообразие видов оружия и укрытий;
5. интуитивность интерфейсов;
6. генератор карт.

Данный игровой продукт может использоваться в определенном круге обращения.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Вайсфельд М. Объектно-ориентированное мышление. − СПб.: Питер, 2014. – 304 c.

Шилдт, Герберт. C# 4.0: полное руководство.: Пер. с англ. − М.: ООО «И.Д. Вильямс», 2011. − 1056 с.

Jason, Gregory. Game Engine Architecture.: [[англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA" \o "Английский язык)]. − CRC Press, 2009. − 864 с.

Хокинг Дж. Unity в действии. Мультиплатформенная разработка на C#.: Пер. с англ. И. Рузмайкиной. − СПб.: Питер, 2016. − 336 с.

Шампандар, Алекс Искусственный интеллект в компьютерных играх: как обучить виртуальные персонажи реагировать на внешние воздействия.: Пер. с англ. − М.: ООО «И.Д. Вильямс», 2007. − 768 с.

Мандел Т. Разработка пользовательского интерфейса.: Пер. с англ. – М.: ДМК Пресс. – 416 с.

Адамс Д. DirectX: продвинутая анимация.: Пер. с англ. − М.: КУДИЦ-ОБРАЗ, 2004. − 480 c.

Скотт Роджерс. Level Up!: The Guide to Great Video Game Design. – [John Wiley & Sons](https://ru.wikipedia.org/wiki/John_Wiley_%26_Sons), 2010. – 520 с.

Куликов, С. C. Тестирование программного обеспечения. Базовый курс.: Минск: Четыре четверти, 2017. − 312 с.