Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики и технологии программирования

Дисциплина: Программирование

К ЗАЩИТЕ ДОПУСТИТЬ

\_\_\_\_\_\_\_\_\_\_И. А. Удовин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе  
на тему

Игра «Tower Attack»

Студент

Руководитель

А. Е. Головачук

И. А. Удовин

МИНСК 2020

Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики и технологии программирования

Дисциплина: Программирование

УТВЕРЖДАЮ

Заведующий кафедрой ИиТП

\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Волорова

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2020 г.

ЗАДАНИЕ

по курсовой работе студента  
Головачука Алексея Евгеньевича

1. Тема работы: Игра «Tower Attack»
2. Срок сдачи студентом законченной работы: 1 июня 2020 г.
3. Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение. Основная часть. 1 Общие положения. 2 Алгоритм реализации проекта. 3 Игровой движок (Game Engine). 4 Организация и ход игровой программы. 4.1 Аспекты для разработки игровой программы. 4.2 Тип и правило игровой программы. 4.3 Юниты игровой программы. 4.4 Виртуальный мир. 4.5 Подход, основанный на реакциях. 4.6 Анимации юнитов и реализация передвижения юнитов по виртуальному миру. 4.7 Очки здоровья и урон. 4.8 Пользовательский интерфейс (UI). 4.9 Экономика. 4.10 Тестирование. 5 Код игровой программы. Заключение. Список использованных источников.

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов курсовой работы | Объём этапа, % | Срок выполнения этапа | Примечания |
| Проектирование архитектуры (описание системы с точки зрения пользователя) | 19 | 13.03-25.03 |  |
| Подробный проект (спецификации) | 38 | 26.03-20.04 |  |
| Кодирование и отладка | 31 | 21.04-11.05 |  |
| Тестирование модулей | 8 | 12.05-17.05 |  |
| Интеграция (объединение модулей в систему) | 4 | 18.05-20.05 |  |

Дата выдачи задания: 06 марта 2020 г.

Руководитель И. В. Удовин

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

ОСНОВНАЯ ЧАСТЬ 6

1 Общие положения 6

2 Алгоритм реализации проекта 7

3 Игровой движок (Game Engine) 8

4 Организация и ход игровой программы 9

4.1 Аспекты для разработки игровой программы 9

4.2 Тип и правило игровой программы 10

4.3 Юниты игровой программы 10

4.4 Виртуальный мир 12

4.5 Подход, основанный на реакциях 13

4.6 Анимации юнитов и реализация передвижения юнитов

по виртуальному миру 15

4.7 Очки здоровья и урон 15

4.8 Пользовательский интерфейс (UI) 16

4.9 Экономика 17

4.10 Тестирование 18

5 Код игровой программы 18

ЗАКЛЮЧЕНИЕ 19

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20

**ВВЕДЕНИЕ**

Цель курсовой работы – разработать игровой программный продукт под названием «Code-18» в жанре «Tower Attack», дополнить знания синтаксиса языка C# и применить его на практике.

C# – 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.

Языки программирования тесно связаны друг с другом таким образом, что на каждый новый язык оказывают в той или иной форме влияние его предшественники. В ходе этого процесса свойства одного языка приспосабливаются к другому языку, полезные нововведения внедряются в существующий контекст, а устаревшие конструкции удаляются. Таким путем развиваются языки программирования и совершенствуется искусство программирования. И в этом отношении C# не является исключением.

Язык программирования C# является прямым наследником двух самых удачных языков программирования: С и C++. Он также имеет тесные родственные связи с еще одним языком: Java. Ясное представление об этих взаимосвязях имеет решающее значение для понимания С#. [1]

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 1-го курса факультета ИиТП БГУИР на II семестр 2020 года.

**ОСНОВНАЯ ЧАСТЬ**

1. **Общие положения**
   1. Для разработки игры используется программа Unity 3D и движок Unity.
   2. За концепцию взята механика игр жанра «Tower defense» и переработана под новый жанр «Tower attack».
   3. Для осуществления цели курсовой работы применяется язык программирования C# в игровой сфере.
   4. В ходе написания игрового кода изучены многие возможности языка C#, для чего и как его применять.
   5. Реализована начальная кодовая база для развития данного проекта:

* загрузочное меню;
* меню «паузы»;
* взаимодействие игровых персонажей (юнитов);
* режим «карта»;
* система навигации персонажей;
* движение персонажей по определенной траектории;
* создание и смерть игровых юнитов.
  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 |
| ↓ |
| Создание виртуального мира |
| ↓ |
| Создание игровых интерфейсов |
| ↓ |
| Программирование анимации юнитов |
| ↓ |
| Настройка взаимодействия юнитов |
| ↓ |
| Реализация передвижения юнитов  по виртуальному миру |
| ↓ |
| Очки здоровья и урон |
| ↓ |
| Настройка меню (меню паузы,  меню выигрыша и проигрыша) |
| ↓ |
| 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. [2]

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. [3]

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. [3]

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, сила притяжение и т.д.;
* 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;
* примитивный искусственный интеллект (ПИИ) – создание компьютерных агентов с использованием методов ПИИ; сюда входит написание скриптов и срабатывании их при определенных условиях.
* звуковое сопровождение – интеграция музыки, звуковых эффектов в нужное место и время.

В данной курсовой работе не реализованы такие аспекты как: сценарии, сетевые коммуникации, 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. [4]

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

Концепция представляет собой жанр игр «Tower defense». В отличии от последнего реализация данной идеи представлена в следующем виде: задача игрока состоит в том, чтобы довести юнитов до конечной точки, пройдя враждебное сопротивление, которое реализуется через ПИИ. Игрок имеет право выбрать юнитов и наилучший маршрут для достижения цели. Во время движения игрок имеет право менять маршрут, покупать юнитов, уничтожать вражеских юнитов, получая за это вознаграждение. Миссия считается проваленной в случае смерти всех юнитов и не возможности приобрести новых.

1. **Юниты игровой программы**

В данной игровой программе используются игровые: управляемые (синие) и враждебные (красные) юниты.

Синие юниты представлены на рисунках 4.3.1-4.3.3.

Характеристики tank:

* здоровье – 400;
* урон – 50;
* стоимость – 200;
* уязвимость – башня «Tesla».

Рисунок 4.3.1 – Tank

Характеристики caterpillar:

* здоровье – 150;
* урон – 50;
* стоимость – 100;
* уязвимость – башня «Minigan».

Рисунок 4.3.2 – Caterpillar

Характеристики helicopter:

* здоровье – 300;
* урон – 300 (50 урона с одного орудия);
* стоимость – 300;
* уязвимость – башня «Anti-air».

Рисунок 4.3.3 – Helicopter

Красные юниты представлены на рисунках 4.3.4-4.3.6.

Характеристики башни «Minigan»:

* здоровье – 200;
* урон – 4 (2 урона с одного орудия);
* вид атаки – пули;
* эффективность против – caterpillar.

Рисунок 4.3.4 – Башня «Minigan»

Характеристики башни «Anti-air»:

* здоровье – 200;
* урон – 100 (50 урона с одного орудия);
* вид атаки – ракета;
* эффективность против – helicopter.

Рисунок 4.3.5 – Башня «Anti-air»

Характеристики башни «Tesla»:

* здоровье – 500;
* урон – 2 урона/c;
* вид атаки – электричество;
* эффективность против – tank.

Рисунок 4.3.6 – Башня «Tesla»

1. **Виртуальный мир**

Виртуальный мир – искусственно созданный мир, построенный посредством программирования, на основе компьютерных технологий. Он представлен на рисунке 4.4.1.

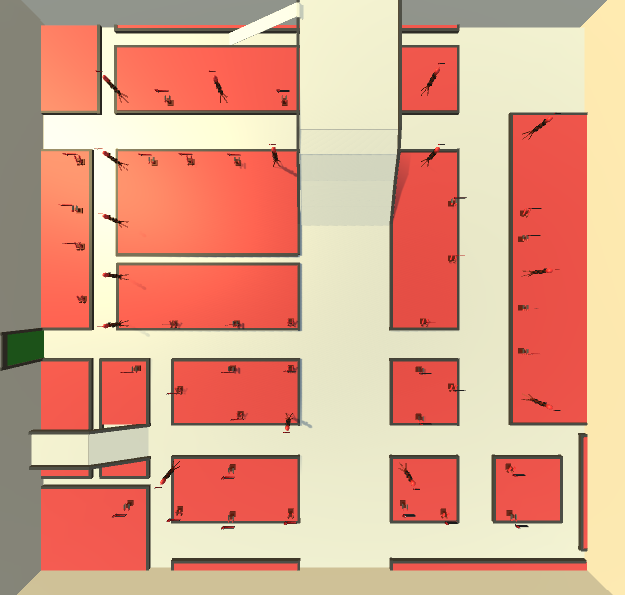


Рисунок 4.4.1 – Виртуальный мир

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. Следует заменить, что, вопреки распространенного мнения, система искусственного интеллекта (ИИ), основанные на реакциях (которые называют также активизируемыми системами ИИ), вполне могут проявлять недюжинную компетентность, во многих случаях преобладая над системами, основанными на планировании. [4]

Взаимодействие двух противоположных юнитов основаны на обнаружение столкновений. При попадании объекта с меткой (тегом) одного из юнитов либо самого юнита в зону действия поражения юнита, имеющий другой цвет, последний активируется с помощью функций:

public void OnTriggerEnter(Collider other)

{

if (other.CompareTag(bulletInfo))

{

float damage = 0;

damage = other.gameObject.GetComponent<BulletMove>().damage;

Destroy(other.gameObject);

// дополнительный код, не относящийся к данному пункту   
}

}

void OnTriggerEnter(Collider other)

{

if (other.CompareTag(tagNameFight))

{

obj.target.Add(other.gameObject.transform);

}

}

Активированный юнит добавляет информацию о месторасположении юнита, попавшего в зону его действия, в список своих целей, затем активирует поворот орудия/орудий на поражение (рис.4.5.1):

void Update()

{

if (target.Count != 0)

{

// дополнительный код, не относящийся к данному пункту

LookAtObj.transform.LookAt(target[0]);

if (isShoot == false)

{

StartCoroutine(shoot());

}

}

}

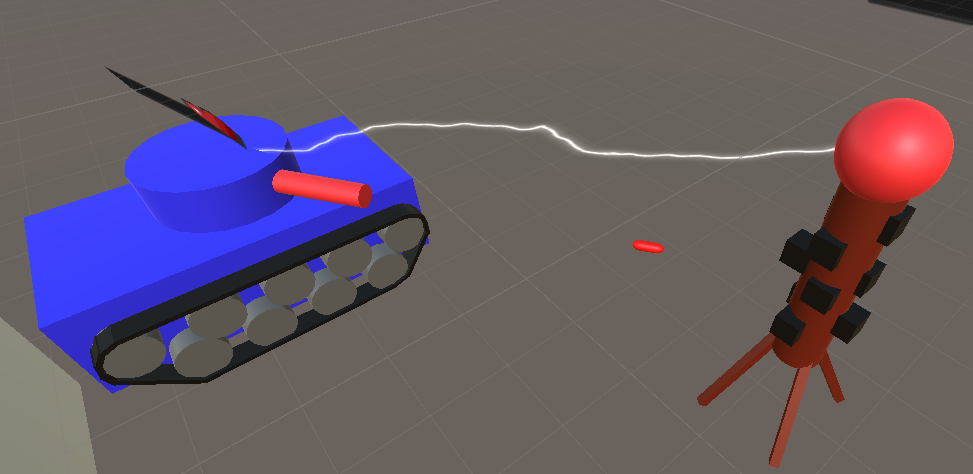


Рисунок 4.5.1 – Взаимодействие юнита tank и башни «Tesla»

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 функцию Vector3.MoveTowards, в которой применяются формулы скалярной разности, программа вычисляет координаты точек, на которые требуется переместить юнита. [5]

1. **Очки здоровья и урон**

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. [6]

В данном проекте индикатор очков здоровья представлен в виде шкалы здоровья на рисунке 4.7.1.



Рисунок 4.7.1 – Шкала здоровья

1. **Пользовательский интерфейс (UI)**

Раскрывая термин «интерфейс» узкоспециализированно, можно сказать, что он объединяет устройства ввода и вывода и программное обеспечение, которое обслуживает их. В более широком смысле интерфейс включает в себя все, что помогает пользователю взаимодействовать с продуктом. [7]

В проекте использованы следующие пользовательские интерфейсы, которые представлены на рисунках 4.8.1-4.8.4:

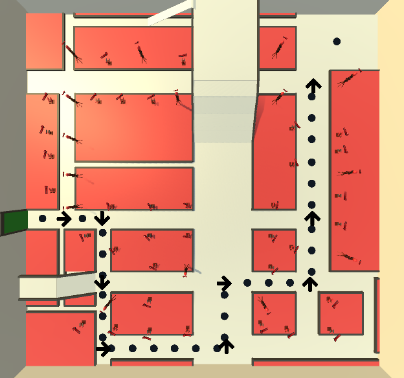
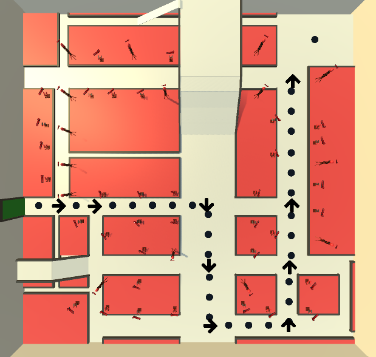


Рисунок 4.8.1 – Игровая карта с изображением разных маршрутов

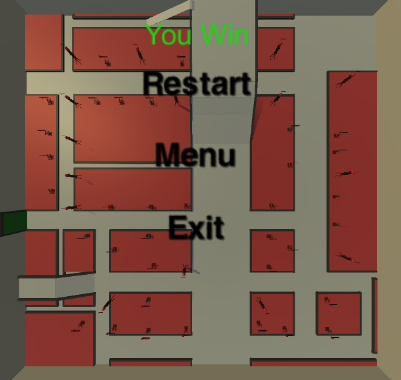
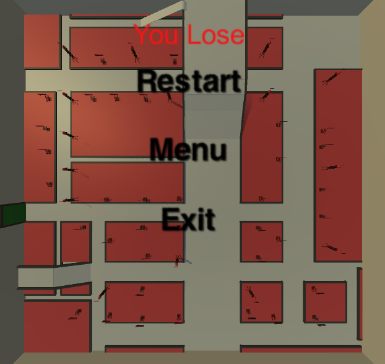


Рисунок 4.8.2 – Меню выигрыша и проигрыша

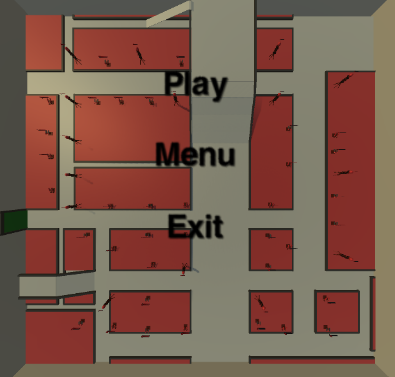


Рисунок 4.8.3 – Меню паузы

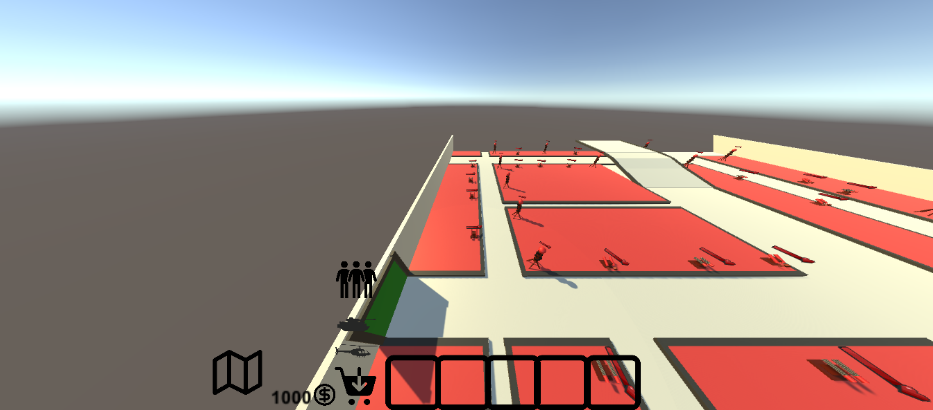


Рисунок 4.8.4 – Игровой интерфейс

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, прозрачная, линейная.

В игровой программе реализуется игровая валюта. Она используется для приобретения юнитов и добавляется при уничтожении вражеских юнитов (башен).

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. [8]

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/Code18>.

**ЗАКЛЮЧЕНИЕ**

Итогом курсовой работы стал новый игровой продукт «Code-18», полученный с использованием платформы Unity.

В процессе проведения курсовой работы изучены возможности применения языка C# в игровой сфере. Актуальность работы состоит в том, что в дополнении к общему курсу изучения языка C# рассмотрены новые возможности его применения 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 (смена маршрута);
3. выбор и покупка юнитов в соответствии с целями и имеющимися финансами;
4. возможность защиты или нападения при попадании объектов в зону реагирования юнита с применением поворота орудия;
5. анимация юнитов;
6. разнообразие видов юнитов;
7. звуковое сопровождение;
8. интуитивность интерфейсов.

Данный игровой продукт может использоваться в определенном круге обращения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

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)]. − 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.

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.

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

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