Федеральное государственное образовательное бюджетное учреждение высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ

РОССИЙСКОЙ ФЕДЕРАЦИИ»

(Финансовый университет)

**Колледж информатики и программирования**

ПМ.03 Участие в интеграции программных модулей

Группа: 4ПКС-219

УТВЕРЖДАЮ

Председатель предметно-цикловой комиссии программирования в компьютерных системах

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Т.Г. Аксёнова/

«5» июня 2023 г.

**ДИПЛОМ**

**На тему: Разработка 2D инди игры на Unity с элементами РПГ крафта и экономики**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Руководитель диплома

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Т.Г. Аксёнова/

Исполнитель диплома

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /М.Я. Алексеев/



Оценка за проект: \_\_\_\_\_\_\_\_\_\_\_\_

«5» июня 2023 г.

Москва

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc123077115)

[ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 5](#_Toc123077116)

[1.1 Описание предметной области 5](#_Toc123077117)

[1.2 Сравнительный анализ программ-аналогов 5](#_Toc123077118)

[1.3 Постановка задачи 5](#_Toc123077119)

[1.4 Характеристика инструментальных средств разработки 5](#_Toc123077120)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ 6](#_Toc123077121)

[2.1 Анализ требований и разработка спецификаций 6](#_Toc123077122)

[2.2 Проектирование программного обеспечения 6](#_Toc123077123)

[2.3 Разработка программного обеспечения 6](#_Toc123077124)

[2.4 Отладка и тестирование программного обеспечения 6](#_Toc123077125)

[2.5 Руководство по использованию программы 6](#_Toc123077126)

[ЗАКЛЮЧЕНИЕ 7](#_Toc123077127)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 8](#_Toc123077128)

# ВВЕДЕНИЕ

На сегодняшний день человек обречён постоянно сталкиваться с вычислительными устройствами. Если еще с десяток лет назад компьютеры можно было встретить только в лабораториях, институтах и на крупных предприятиях, то сейчас они проникли почти во все сферы жизни общества. Одной из них является индустрия развлечений, которая в свою очередь неразрывно связана с играми.

Практически с самого своего зарождения, разработка компьютерных игр стала одной из самых трудоемких задач для различных видов специалистов из сферы информационных технологий. Это программисты, художники, сценаристы, экономисты, системные аналитики, маркетологи и т.д.

С точки зрения пользователя, компьютерная игра может иметь разные задачи. Для одних людей компьютерная игра — это способ отдохнуть после трудного дня или скоротать время, для других — это инструмент для общения и социального взаимодействия, для кого-то игра — это возможность чему-то научиться.

Главной идеей данной работы является разработка градостроительной стратегической инди игры с элементами RPG и экономики. В основе лежит приключение, где все действия происходят в режиме реального времени, а мир подкидывает игроку не мало испытаний, для преодоления которых предстоит прости всевозможные трудности. Исполняя роль главного героя в непредсказуемом мире – боритесь не только за собственную жизнь, но и процветание целого поселения, отправляйтесь на исследование локаций, добывайте ценные ресурсы, решайте неожиданные задачи и справляйся с трудностями, которые дадут доступ к новым возможностям.

Целью данного проекта ставится разработка игры, предоставляющей спектр возможностей для постройки развитой колонии, при помощи управления на макро/микро уровне отдельными юнитами.

В задачи проекта входит создание стратегической игры в реальном времени, а также проектирование, реализация, тестирование, анализ приложений-аналогов и документирование по проекту.

Объектом исследования выступают RTS игры.

Предметом исследования является технологии разработки RTS игры.

Основное функциональное назначение разрабатываемого программного средства состоит в предоставлении пользователю возможности не только с интересом проводить время за игрой, но и при этом развивать нестандартное мышление в критических обстоятельства, лидерские качества управления социумом, а также пониманию важности принятия решений и ответственности за них.

После анализа ситуации на рынке, можно предположить, что данная идея достаточно актуальна, дабы обеспечить ей продвижение в массовом секторе.

Для решения поставленного вопроса будет использоваться межплатформенная среда разработки Unity и язык программирования C#.

# ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

## Описание предметной области

Видеоигры в жанре стратегии в реальном времени (RTS) — это игры, в которых вы управляете отрядами юнитов, строите здания и собираете ресурсы для постройки небольшого городка или города. У них часто есть воздушная орфографическая камера, и вы действуете как всеведущее высшее существо, которое может макро/микроуправлять своими юнитами по всей карте. В отличие от «пошаговых стратегических игр», в которых каждый игрок играет по очереди, а остальные должны ждать, пока вы (или ИИ) закончите ход, стратегия в реальном времени заставляет игроков играть цельно и непрерывно.

## Сравнительный анализ программ-аналогов

Прекрасными примерами этого жанра являются серия Warcraft / Starcraft от Blizzard, серия Caesar от Impressions Games или серия Empire Earth .от Stainless Steel Studios и Mad Doc Software.

## Постановка задачи

## Характеристика инструментальных средств разработки

Разрабатываемое в рамках курсового проекта программное обеспечение будет спроектировано с применением среды разработки Unity, редактора кода Visual Studio и языка программирования C#.

Unity – это игровой движок 2D и 3D, который существует с 2005 года. Разработанный Unity Technologies, он был создан для того, чтобы предоставить большему количеству разработчиков доступ к инструментам разработки игр, что в те дни было новым предприятием. За свою долгую жизнь двигатель значительно изменился и расширился, сумев идти в ногу с новейшими практиками и технологиями.

Даже сегодня основной задачей игрового движка является как предоставление наиболее надежного набора инструментов, возможного для индустрии разработки игр, так и максимально упрощение использования движка разработчиками игр любого уровня квалификации (да, включая начинающих разработчиков). Они также расширили свой охват и в других отраслях, уделяя огромное внимание 3D-разработке в реальном времени, что делает ее одним из самых мощных доступных движков.

Microsoft Visual Studio – это интегрированная среда разработки. Microsoft Visual Studio содержит набор инструментов, облачных сервисов и расширений, которые необходимы разработчикам для создания и сопровождения программ. Данное программное обеспечение включает в себя компиляторы, средства автодополнения кода, визуальные редакторы макетов и многие другие функции, облегчающие процесс разработки программного обеспечения.

C# — это современный объектно-ориентированный и компонентно-ориентированный язык программирования общего назначения, разработанный и поддерживаемый компанией Microsoft. С# имеет свои корни в семействе языков C и потому имеет много общего с языками C, C++, Java и JavaScript. Программы на C# выполняются в среде .NET, виртуальной системе выполнения, называемой CLR, и наборе библиотек классов. CLR — это реализация Microsoft общеязыковой инфраструктуры (CLI), международного стандарта.

В качестве инсталлятора будет использоваться встроенный функционал Unity для создания билдинга проекта.

# ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ

## Анализ требований и разработка спецификаций

## Проектирование программного обеспечения

Одна из основных механик игры RTS: размещения зданий. Это логика, которая позволяет вам:

 выбрать тип здания для постройки;

 подготовьте место, где вы хотите его разместить (при проверке / показе недопустимого размещения);

 на самом деле поместите его на землю, чтобы он был «закреплен» и больше не мог быть перемещен;

 (при желании оставайтесь в режиме «размещения», чтобы вы могли создать другое здание того же типа).

На этом раннем этапе нашего проекта у нас еще нет игровых активов (таких как модели зданий или персонажей, звуки и музыка, красивые изображения и значки…). Мы начнем с использования примитивов Unity, таких как кубы, сферы, пустые изображения и т. д. Однако мы можем использовать систему префабов, чтобы легко подготовить игровые объекты, которые мы хотим создать на нашей сцене во время игры.

Префабы — это функция Unity, которая позволяет нам подготовить игровой объект со всеми его сетками, компонентами, сценариями и другими уникальными свойствами, чтобы, когда вам нужно добавить этот объект в вашу сцену, он уже был правильно настроен. Например, представьте себе сложную модель персонажа со сценарием для управления движением, аниматором для управления анимацией бега и прыжка, источником звука для создания звуковых эффектов при переключении оружия и некоторым физическим коллайдером для взаимодействия с окружающей средой при движении. Если бы вам приходилось перенастраивать все это каждый раз, когда вы добавляете персонажа в сцену (скажем, когда вы меняете уровень игры и перезагружаете новую сцену), это было бы довольно болезненно. Вместо этого вы можете настроить все один раз, а затем «создать экземпляр» префаба.

## Разработка программного обеспечения

Первое, что мы хотим сделать, это получить список всех доступных типов зданий. Итак, давайте определим класс для представления некоторых абстрактных данных о здании (тип здания). Ему не нужно наследовать от класса MonoBehaviour, потому что он просто содержит некоторые данные:

using System.Collections.Generic;

public class BuildingData

{

private string \_code;

private int \_healthpoints;

public BuildingData(string code, int healthpoints)

{

\_code = code;

\_healthpoints = healthpoints;

}

public string Code { get => \_code; }

public int HP { get => \_healthpoints; }

}

Листинг 1 – класс BuildingData

Теперь мы можем создать глобально доступную переменную со списком данных построения ссылок — сегодня мы просто определим ее вручную, но в конечном итоге мы должны загрузить эти ссылки из более легко модифицируемого файла данных. Чтобы централизовать эти глобальные переменные и упростить их получение, мы создадим новый файл C#: Globals.cs. И этот файл будет содержать статические переменные или функции, поэтому их можно будет вызывать без создания экземпляра скрипта.

public class Globals

{

public static BuildingData[] BUILDING\_DATA = new BuildingData[]

{

new BuildingData("Building", 100)

};

}

Листинг 2 – класс Globals

Мы определили наш «абстрактный» класс данных. Тут необходимо быть осторожными: это не совсем абстрактно с точки зрения C#, потому что мы фактически создаем его экземпляр, чтобы иметь наш список доступных зданий; я имею в виду, что мы не будем напрямую представлять эти данные на экране, а вместо этого передадим их нашим реальным экземплярам зданий — тем, которые мы размещаем на земле, когда перетаскиваем мышь и нажимаем.

Эти экземпляры обрабатываются другим классом Building:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Building

{

private BuildingData \_data;

private Transform \_transform;

private int \_currentHealth;

public Building(BuildingData data)

{

\_data = data;

\_currentHealth = data.HP;

GameObject g = GameObject.Instantiate(

Resources.Load($"Prefabs/Buildings/{\_data.Code}")

) as GameObject;

\_transform = g.transform;

}

public void SetPosition(Vector3 position)

{

\_transform.position = position;

}

public string Code { get => \_data.Code; }

public Transform Transform { get => \_transform; }

public int HP { get => \_currentHealth; set => \_currentHealth = value; }

public int MaxHP { get => \_data.HP; }

public int DataIndex

{

get {

for (int i = 0; i < Globals.BUILDING\_DATA.Length; i++)

{

if (Globals.BUILDING\_DATA[i].Code == \_data.Code)

{

return i;

}

}

return -1;

}

}

}

Листинг 3 – класс Building

Мы создаем новый Buildingэкземпляр, используя BuildingDataссылку, чтобы у него были все необходимые метаданные. Экземпляр также имеет игровой объект и связанное с ним преобразование, потому что это реальный объект в сцене. Этот игровой объект загружается из нашей папки «Ресурсы», это префаб, который мы создали ранее. Также обратите внимание, что у нас есть более сложные методы доступа, чем в нашем предыдущем BuildingDataклассе:

* для \_currentHealthполя у нас есть и геттер, и сеттер ( HP), потому что нам может понадобиться быстрый способ обновить значение вне класса;
* DataIndexаксессор не соответствует какому-либо частному полю в классе, это «вычисляемое» свойство, которое дает нам индекс экземпляра данных абстрактного типа здания в глобальном списке.

В дополнение к этим аксессорам и конструктору у нас есть пользовательская функция для установки положения игрового объекта на сцене, SetPosition().

В RTS-играх довольно часто возникает «фантомная» визуализация здания, которое вы хотите разместить, прежде чем фактически разместить его на местности. Таким образом, вы сможете лучше подготовить это размещение и проверить правильность размещения здания. Этот «призрак» следует за вашей мышью, прикрепляя к земле здание, которое скоро будет построено.

Для этого мы будем использовать систему физического луча Unity: это позволяет нам направлять луч от камеры на землю в зависимости от положения мыши; и, следовательно, получить точную точку в трехмерном мире на земле, на которую указывает наша мышь.

Прежде чем добавлять все проверки размещения, мы внедрим некоторую логику для обновления материалов экземпляра здания и четко покажем, размещено ли оно или является «фантомом» с допустимым или недопустимым размещением. Действительно, экземпляр сборки может находиться в любом из этих трех состояний:

* «valid»: когда здание еще не размещено и находится в «фантомном» режиме, если все проверки размещения в порядке, то «фантомное» здание должно отображаться в виде зеленой прозрачной формы;
* «invalid»: все еще в «фантомном» режиме, но некоторые проверки размещения не проходят — тогда здание должно использовать красный прозрачный материал;
* «fixed»: после действительного размещения здания нам нужно восстановить его исходный материал (материалы), которые установлены на префабе здания (с нашим базовым кубом это материал Unity по умолчанию).

В играх «безделушки» — это все маленькие объекты, которые оживляют мир, вызывают столкновения и с которыми обычно нельзя взаимодействовать: деревья, камни, стены… Здесь, скажем, у нас есть наши маленькие камни на земле, и мы не может разместить здание, если оно сталкивается со скалой. Чтобы проверить это столкновение, мы будем использовать коллайдеры, которые мы добавили в наши префабы, а также встроенные функции Unity OnTriggerEnter()и OnTriggerExit(). Это делается на BuildingManager:

using UnityEngine;

[RequireComponent(typeof(BoxCollider))]

public class BuildingManager : MonoBehaviour

{

private BoxCollider \_collider;

private Building \_building = null;

private int \_nCollisions = 0;

public void Initialize(Building building)

{

\_collider = GetComponent<BoxCollider>();

\_building = building;

}

private void OnTriggerEnter(Collider other)

{

if (other.tag == "Terrain") return;

\_nCollisions++;

CheckPlacement();

}

private void OnTriggerExit(Collider other)

{

if (other.tag == "Terrain") return;

\_nCollisions--;

CheckPlacement();

}

public bool CheckPlacement()

{

if (\_building == null) return false;

if (\_building.IsFixed) return false;

bool validPlacement = HasValidPlacement();

if (!validPlacement)

{

\_building.SetMaterials(BuildingPlacement.INVALID);

}

else

{

\_building.SetMaterials(BuildingPlacement.VALID);

}

return validPlacement;

}

public bool HasValidPlacement()

{

return \_nCollisions == 0;

}

}

Листинг 4 – класс BuildingManager

В результате некоторые детали и код были намеренно опущены из работы, для упрощения и компактности изложения, но в итоге нам удалось запрограммировать одну из наших основных механик RTS, и теперь мы можем размещать здания на земле, проверять правильность размещения и обновлять как состояние здания, так и его отображение по мере необходимости.

В дальнейшем необходимо добавить базовую систему управления пользовательским интерфейсом и множество механик расширяющих геймплей.

## Отладка и тестирование программного обеспечения

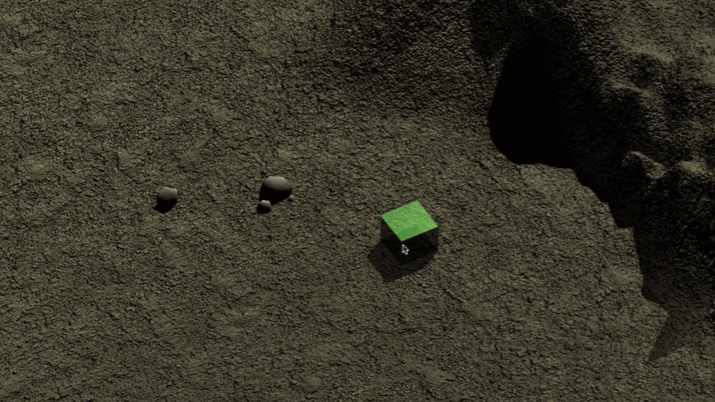


Рисунок 1 – Статус 1

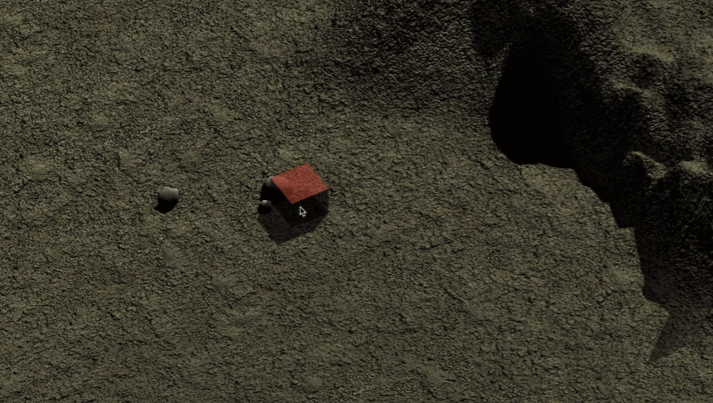


Рисунок 2 – Статус 2

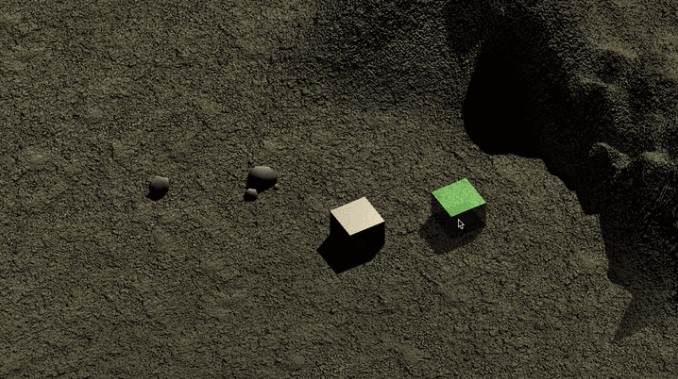


Рисунок 3 – Статус 3



Рисунок 4 – Темы меню

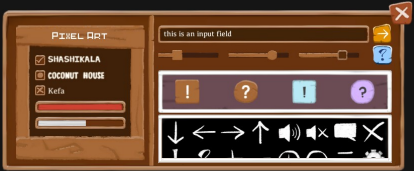


Рисунок 5 – Настройки меню

## Руководство по использованию программы

Руководство системного программиста

Руководство программиста

Руководство оператора

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

*Законодательные и нормативные акты:*

1. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. – М.: Стандартинформ, 2012. – 61 с.
2. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2010. – 92 с.
3. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 47 с.
4. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 39 с.
5. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 122 с.
6. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 32 с.
7. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

*Учебная и научная литература:*

1. Иванова, Г.С. Технология программирования: учебник для студентов вузов обуч. по напр. «Информатика и вычислительная техника» / Г.С. Иванова. – 3-е изд., стер. – Москва: Кнорус, 2018. – 333 с.
2. Павловская, Т.А. C#. Программирование на языке высокого уровня: учебник для студентов вузов. – СПб: Питер, 2020. – 432 с.
3. Перлова, О.Н., Ляпина, О.П., Гусева, А.В. Проектирование и разработка информационных систем: учебник. – 2-е изд, стер. – М.: Издательский центр «Академия», 2018. – 256 с.

*Интернет-документы:*

1. Введение в язык C# и .NET Framework. – [Электронный ресурс]. – URL: <https://learn.microsoft.com/en-us/dotnet/csharp/?ref=hackernoon.com> (дата обращения: 06.10.2022)
2. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. – Москва: ФОРУМ: ИНФРА-М, 2020. – 400 с. – (Среднее профессиональное образование). – ISBN 978-5-8199-0812-9. – Текст: электронный. – URL: <https://znanium.com/catalog/product/1067012> (дата обращения: 06.10.2022)
3. Гуриков, С. Р. Введение в программирование на языке Visual C#: учебное пособие / С.Р. Гуриков. — МОСКВА: ФОРУМ: ИНФРА-М, 2020. (Высшее образование: Бакалавриат). – ISBN 978-5-00091-458-8. ­– Текст: электронный. – URL: <https://znanium.com/catalog/product/1092167> (дата обращения: 06.10.2022)
4. Интернет-сервис для построения схем и диаграмм Draw.io. – [Электронный ресурс]. – URL: <https://www.draw.io/> (дата обращения: 06.10.2022)
5. Интернет-сервис для построения UML-диаграмм. – [Электронный ресурс]. – URL: <https://plantuml.com/> (дата обращения: 06.10.2022)
6. Полное руководство по языку программирования C# 6.0 и платформе .NET 4.6. – [Электронный ресурс]. – URL: <http://metanit.com/sharp/tutorial/> (дата обращения: 06.10.2022)