МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики и технологий программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Пошаговая 2D стратегия на Unity»

Выполнил:

студент гр.653502 Овчинников Л.Н.

Проверил: Козуб В.Н.

г. Минск

2017 год

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ 2**

**ЦЕЛИ И ЗАДАЧИ 3**

**1 ТЕОРЕТИЧСКАЯ ЧАСТЬ 4**

* 1. 2D, 3D ИЗМЕРЕНИЯ, ТИПЫ ГРАФИКИ 4
  2. ПРАВИЛА СТРАТЕГИЙ, ПРИНЦИП ПОШАГОВОЙ СТРАТЕГИИ 6

1. **РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 10**
   1. ПРАВИЛА ИГРЫ 10
   2. РАЗРАБОТА ПРОГРАММЫ 11
   3. ДАЛЬНЕЙШИЕ ПЛАНЫ НА ДОРАБОТКУ ИГРЫ 15

**3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 16**

**ЗАКЛЮЧЕНИЕ 18**

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 19**

**ВВЕДЕНИЕ**

С развитием компьютерных технологий развитие получили и компьютерные игры. Появилось огромное количество жанров, соответствующих различным интересам пользователей. Среди них: аркады, платформеры, скороллеры, симуляторы, стратегии, головоломки и множество других.

Игры в жанре «Стратегия» получили широкую популярность среди пользователей, чем и было обусловлено их развитие. Появлялись новые типы стратегий: 4X, TBS, RTS, гибридные стратегии сочетающие RTS и TBS механики, онлайн стратегии и экономические симуляторы.

Пошаговая стратегия - тип стратегии, где основной игровой процесс происходит в пределах ходов, в отличии от других типов стратегий, в которых игровой процесс привязан к настоящему времени. Наиболее известные пошаговые стратегии X-Com, Civilization, Xenonauts, Heroes of Might and Magic III, Duna.

Unity – широко известная бесплатная интегрированная среда для разработки игр и приложений в 2D и 3D измерениях. Unity поддерживает широкое множество платформ, таких как Windows, iOS, Linux, Android, Xbox One, PS Vita, PS 4.

Игра «RaibowSX» выполнена на Unity. Для курсового проекта был выбран язык программирования C#, так как он предназначен для разработки самых разнообразных приложений, но при этом сохраняет выразительность и элегантность, присущую языкам C.

**ЦЕЛИ И ЗАДАЧИ**

Основной целью данной курсовой работы является изучение и создание 2D пошаговой стратегии на основе интегрированной среды разработки Unity, а также программирование компонентов на языке C#.

Данная работа состоит из 3 разделов: первый раздел теоретический, содержащий общие сведения по теме, второй раздел практический, где описывается код и идея игры, а так же дальнейшие планы на разработку. Третий раздел – руководство пользователя.

Основные задачи, выполнение для написания работы:

1. Изучение необходимой литературы
2. Изучение методов для создания ИИ, поиска пути (PathFinder), интерфейса
3. Изучение жанра пошаговых стратегий
4. Изучение правил пошаговой стратегии
5. Разрисовка ассетов
6. Написание кода программы
7. Создание игры жанра пошаговой стратегии
8. Анализ полученной игры

**1.Теоретический раздел**

**1.1 2D, 3D ИЗМЕРЕНИЯ, ТИПЫ ГРАФИКИ**

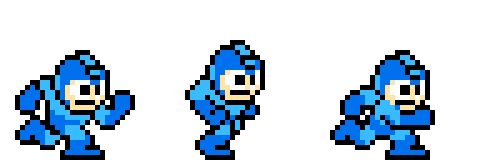
В нынешнее время невозможно представить жизнь без ЭВМ. Компьютеризация затронула все сферы нашей жизни, и с каждым годом масштаб влияния компьютеров в различных областях науки, техники, медицины, экономики увеличивается. Автоматизация облегчает человеческий труд, а иногда и вовсе заменяет его.

Компьютерные программы стали не только помощниками в работе, но также используются людьми при проведении досуга, они стали обширной базой для развлечения и препровождения свободного времени.

Игры, как тип развлечения, стал очень популярен, и в нынешнее время это одна из самых больших сфер, связанных с использование компьютера. С каждым годом качество игр растет, растет уровень графики, физики, аудио ряда. Из-за повышения качества игры вырастает и требовательность пользователей к техническим характеристикам компьютера.

Игры разделяются по измерениям, графике, геймплею, жанру.

По измерению игры различаются на 2D (Рис. 1) и 3D (Рис. 2), где 2D это плоскость с двумя координатными осями x, y, без понятия глубины, а 3D измерение, использующее все три координатные оси, x, y, z. К этому измерению относятся все объемные тела. В то время как в 2D используются спрайты.

(Рис.1)

(Рис.2)

По графике различие более широкое. Существует пиксельная, Low-Poly, рисованная, покадровая, а так же другие виды графики.

В пиксельной графике (Рис. 3) изображение редактируется на уровне пикселей, а разрешение изображения настолько мало, что отдельные пиксели чётко видны. Пиксельная графика очень стара, она зародилась в времена старых компьютеров и приставок. Картинки такого типа занимают меньше места, а так же не нагружают компьютер, что было очень важно когда память у компьютера была ограничена. Сейчас все еще существуют игры с пиксельной графикой. Некоторые новые игры специально стилизованы под старые игры, в таких играх также присутствует специальная 8-бит музыка и эффекты.

(Рис. 3)

Положительные качества такой графики – относительная легкость, дешевизна, а так же низкие требования к памяти процессора. Из отрицательных качеств – графика не переносит масштабирование, выглядит скудно по сравнению с другими типами.

Low-Poly графика (Рис. 4) - графика с использованием минимального количества полигонов без потери понимания объекта. Такой тип графики относится к 3D измерению. Используется в некоторых играх. Положительные качества такой графики – это относительная легкость и простота. Отрицательные качества – малое количества полигонов снижает качество общего вида игры, потому на замену Low-Poly моделям пришли High-Poly модели. А сами Low-Poly игры приобрели популярность в играх для мобильных устройств.

(Рис. 4)

В игре “RainbowSX” использовалась пиксельная графика в 2D измерении, из-за легкости ее рисования, а также легкости использования. Вся игра оформлена под 32 бит графику. Так же в игре использовалась 8 битная музыка.

* 1. **ПРАВИЛА СТРАТЕГИЙ, ПРИНЦИП ПОШАГОВОЙ СТРАТЕГИИ**

Смысл стратегических игр заключается в управлении определёнными ресурсами, которые необходимо использовать для получения преимущества над противником при помощи оперативного плана, разрабатываемого с учётом меняющейся обстановки. Обычными ресурсами в военных стратегиях являются войска и позиция, которые следует развивать и использовать для достижения преимущества и победы. В экономических стратегиях акцент ставится на развитие экономической инфраструктуры подконтрольной игроку стороны. В большинстве стратегий существуют экономическая и военная составляющие части.

Пошаговые стратегии или Turn-Based Strategy – подтип стратегий где игроки делают ходы, где каждому игроку дается ограниченное или не ограниченное количество времени на ход. Так же среди пошаговых стратегий различают гибридные стратегии, где весь игровой процесс привязан не только к ходам, но и к реальному времени, примеры X-COM Enemy Unknown (Рис. 5), Xenonauts (Рис. 6), где гемплей привязан к настоящему времени. В то время как в Civilization (Рис. 7) и Heroes of Might and Magic 3 (Рис. 8) гемплей был привязан только к ходам.

(Рис. 5)

(Рис.6)

(Рис. 7)

(Рис. 8)

Одно из правил пошаговых стратегий заключается в том, что подконтрольные юниты ограничены по количеству клеток для передвижения, а так же действия юнитов привязаны к определенной шкале, TP (time points) в X-COM Enemy Unknown (Рис. 9). В Wargame: Red Dragon (Рис.10) передвижение юнитов привязано к очкам передвижения.

(Рис. 9)

(Рис.10)

В некоторых TBS (Turn-Based Strategy) реализована процентная механика попаданий, среди которых все части X-COM (Рис. 11), а так же Xenonauts. Процентная механика заключается в том что шанс попадания рассчитывается от дистанции, типа вооружения, типа укрытия или каких-либо статистик противника. Процентный тип попадания может быть реализован и в RTS, наиболее известный пример линейка игр Wargame, где каждый юнит привязан к шансу попадания (Рис.12)

(Рис. 11)

(Рис. 12)

В моей игре геймплей будет привязан к очкам времени, они же TP, а так же будет реализована процентная механика попаданий.

**2. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**2.1 ПРАВИЛА ИГРЫ**

Игрок стартует с четырьмя подконтрольными юнитами, 1 солдат, класс солдата Assault (Рис. 1), вооружен дробовиком и общим вторичным оружием, из особенностей пули как GameObject вылетают веером, давая преимущество в борьбе на ближних дистанциях, шанс их попадания рассчитывается как общий, при промахе летят дальше задевая коллизию стен и исчезая, при попадании сносят урон равный урону одно пули. 2-3 солдаты, тип Rifleman (Рис. 2), вооружен винтовкой и общей вторичным оружием, из особенностей автомат может выпускать очередь из пуль, точность очереди мала, но убийственна на ближних и средних дистанциях, при промахе пули ведут себя так же как на дробовике. 4 солдат, тип Sniper (Рис. 3), вооружен снайперской винтовкой и пистолетом, из особенностей снайперская винтовка имеет идеальную точность на средней дистанции, однако точность снижается при ближней дистанции, при стрельбе в упор имеет около нулевой шанс попадания, при промахе пули ведут себя так же, как и на остальных типах вооружения.

(Рис. 1) (Рис. 2) (Рис. 3)



Игрок имеет право переключаться с одного героя на другого, но не давая выбрать всех героев одновременно, у каждого героя есть свое количество HP и TP. Чтобы победить, игроку нужно выполнить задачу или устранить всех противников. Чтобы пройти игру, игрок должен пройти цепочку заданий, часть которых будут за пределами миссии.

За пределами миссий игроку будет предоставлена возможность управления базой, кастомизация юнитов, их снаряжения, а так же выбор следующей миссии.

Об истории вселенной:

Действие игры происходит в 1970-х годах, где внезапно поднимается уровень преступности во всех городах по всему миру. Где антагонистами выступает группа террористов зовущая себя Дегама. Потому ближе к 1972 году Мировой совет созывает все страны для создания одной антитеррористической организации RaibowSX, которой игроку и придется управлять. После чего под финансированием всего мира строится штаб и нанимает специалистов для развития организации. Как одному из вышестоящих генералов игрок получает в управление базу, а так же ежемесячную поддержку многих стран. От игрока же требуется помогать странам бороться с группой Дегама. В случае если 7 стран покинут проект, игроку будет засчитано поражение.

**2.2 РАЗРАБОТКА ПРОГРАММЫ**

Далее будет подробно разобраны наиболее важные .cpp файлы использованные в данной программе, а так же этот раздел будет руководством по созданию программы

1. Камера управляется скриптом CameraAdjust где public List<GameObject> waypoints – массив игровых объектов по которым камера облетает игровую локацию, private bool Moveable – переменная блокирующая камеру при облете. Функция Move перемещает камеру по каждому waypoint если же функция запросит несуществующий элемент массива, Moveable переменная станет равна true тем самым даст игроку управлять камерой. Управление камерой происходит через WASD, а так же увеличение и уменьшение установлено на колесо мышки.

if (Camera.current != null)

{

Camera.current.transform.Translate(new Vector3(xAxisValue \* speed \* Time.deltaTime, yAxisValue \* speed \* Time.deltaTime, 0.0f));

}

Где xAxisValue и yAxisValue вводятся через клавиатуру:

float xAxisValue = Input.GetAxis("Horizontal");

float yAxisValue = Input.GetAxis("Vertical");

Приближение камеры:

if (Input.GetAxis("Mouse ScrollWheel") != 0)

{

scrollSpeed = 3f;

if (Input.GetAxis("Mouse ScrollWheel") < 0)

{

Camera.main.orthographicSize = Camera.main.orthographicSize + 1 \* Time.deltaTime \* scrollSpeed;

if (Camera.main.orthographicSize > 12)

{

Camera.main.orthographicSize = 12; // Max size

}

}

if (Input.GetAxis("Mouse ScrollWheel") > 0)

{

Camera.main.orthographicSize = Camera.main.orthographicSize - 1 \* Time.deltaTime \* scrollSpeed;

if (Camera.main.orthographicSize < 4)

{

Camera.main.orthographicSize = 4; // Min size

}

}

}

1. Интерфейс управляется скриптами Gametypes, EndButton, HunkerMod, OverwatchButton, AttackButton, MoveButton, SecondaryMod, MainButton

MainButton и SecondaryMod – две кнопки, основного и вторичного оружия, они связаны с cpp файлом Gametypes, который связан со всеми подконтрольными юнитами.

Все кнопки работают через стандартную функцию OnMouseOver и

MoveButton активирует сетку и дочерние клетки.

Активация происходит через скрипт Gametypes и код:

MoveNet = GameObject.Find("Move net");

MoveNet.SetActive(true);

Где Move net – родительский объект всех клеток.

Кнопка FireMod дезактивирует сетку, а так же открывает возможность стрельбы по противнику, стрельба происходит через создание игрового объекта с спрайтом пули, регистрация попаданий же при задевании коллизии пулей.

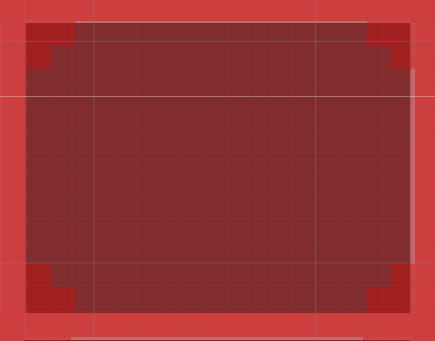
Кнопки OverwatchButton и HunkerMod переключают юнитов в owerwatch – мод при котором союзник или противник атакует движущуюся цель, HunkerDown – мод при котором шанс попасть по противнику уменьшается, однако юнит не может атаковать или двигаться.

Кнопка EndTurn разрешает вражескому ИИ атаковать или двигаться

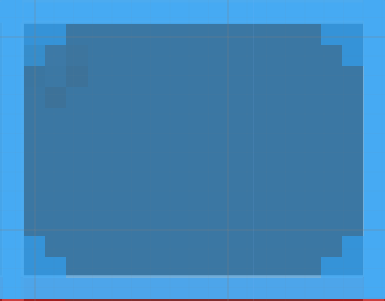
1. Сетка, все клетки сетки содержат скрипт MoveNet\_Cell, где содержится тип клетки, где - 0 временно занятая красная клетка (Рис. 4), 1 - свободная голубая клетка (Рис. 5), 2 - разрушаемая оранжевая клетка (Рис. 6) на месте дверей и окон, и 4 - логически закрытая клетка (Рис. 7), попасть на данные клетки не возможно, они не подчиняются скрипту, а так же не меняют спрайт при клике.

Все клетки с типом 1, при наведении заменяют спрайт на желтый (Рис. 8), что означает что клетка выделена, код:

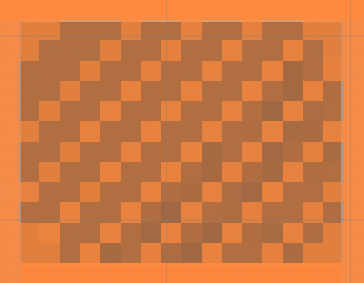
(Рис. 4)



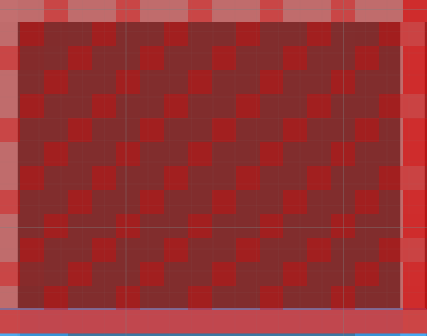
(Рис. 5)



(Рис. 6)



(Рис. 7)



(Рис. 8)



private void OnMouseOver()

{

if (Type == 1 && (id >= 46 || id < 19 || id == 28))

{

gameObject.GetComponent<SpriteRenderer>().sprite = MouseOver;

gameObject.GetComponent<SpriteRenderer>().sortingOrder = 3;

if (Input.GetMouseButton(0))

{

gameObject.GetComponent<SpriteRenderer>().sprite = StandSprite;

if (GameObject.Find("C-Alloy\_Shotgun\_Right").GetComponent<AlloyCaractaristic>().selected == true)

{

GameObject.Find("C-Alloy\_Shotgun\_Right").GetComponent<AlloyCaractaristic>().Moveing = true;

GameObject.Find("C-Alloy\_Shotgun\_Right").GetComponent<AlloyCaractaristic>().CellId = id;

}

if (GameObject.Find("C-Alloy\_Rifle\_Right0").GetComponent<AlloyCaractaristic>().selected == true)

{

GameObject.Find("C-Alloy\_Rifle\_Right0").GetComponent<AlloyCaractaristic>().Moveing = true;

GameObject.Find("C-Alloy\_Rifle\_Right0").GetComponent<AlloyCaractaristic>().CellId = id;

}

}

if (Input.GetMouseButtonUp(0))

{

GameObject.Find("C-Alloy\_Shotgun\_Right").GetComponent<AlloyCaractaristic>().Moveing = false;

GameObject.Find("C-Alloy\_Rifle\_Right0").GetComponent<AlloyCaractaristic>().Moveing = false;

}

}

В этой же части кода описано изменение временно заблокированных клеток на свободные для передвижения. Так же тут описана передача нового id на котором находится юнит, это сделано для того что бы вычитать из TP юнита определенное количество очков за каждую клетку.

1. Скрипт персонажей. Персонажи управляются скритами AlloyCaractaristic и EnemyCaractaristicks

В скрипте AlloyCaractaristic описаны характеристики подконтрольных юнитов, а именно HP – очки жизни, TP – очки действия, Id – личный номер, CellId – номер клетки на которой стоит персонаж, OldCellId – номер клетки до передвижения, он разворачивает спрайт персонажа в другую сторону при передвижении на лево. HitChance – шанс попадания юнита, будет отображаться при наведении мыши на цель с вкюченым FireMod. Так же есть bool переменные, Secondary – при true меняет спрайт на аналогичный, только с основным вторичным вооружением, при false меняет на аналогичный спрайт с первичным вооружением. Selected – ключевая переменная, отражает статус юнита, выбран или не выбран, используется при перемещении, выборе оружия, а так же при входе в overwatch, hunkerdown или при атаке, место появление пули будет зависеть от координат выбранного юнита.

Moveing - bool переменная, используется при динамическом изменении занятых клеток в сетке. EnemyTurn – переменная блокирующая управление при вражеском ходе. Переменные: public Sprite Shotgun0L, Shotgun0R, Shotgun1L, Shotgun1R, Rifle0L, Rifle0R, Rifle1L, Rifle1R,

SRifle0L,SRifle0R, SRifle1L,SRifle1R, Secondary0L,Secondary0R, Secondary1L, Secondary1R; отвечают за спрайт при перемещении, при смене оружия. Переменные:

public int AlloyType;

private int FaceType;

Отвечают за тип юнита, за тип лица, с его случайной генерацией. (Рис 9,10)

(Рис. 9)



(Рис. 10)



**2.3 ПЛАНЫ НА ДОРАБОТКУ ПРОГРАММЫ**

В игре на данный момент не реализован ИИ противника, а так же FireMod, OverwatchMod, HunkerDownMod, кроме всего прочего нет переключения между ходами. Требуется доделать ИИ противников, пошаговую механику, механику Overwatch и HunkerDown, также сделать разрушаемость деревянного пола и стен, сделать еще от 4 до 50 миссий и базу, по механике напоминающую HQ в X-Com.

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Разрабатываемый проект является игровой программой "RaibowSX”.

Рекомендуемая конфигурация аппаратного и программного обеспечения:

\* свободного места на жёстком диске – около 300 мегабайт;

\* мышь, клавиатура;

\* операционная система семейства Windows.

Для начала работы необходимо запустить исполняемый файл RainbowSX.exe.

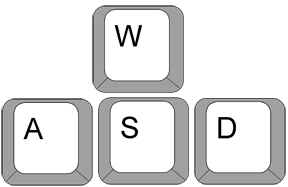
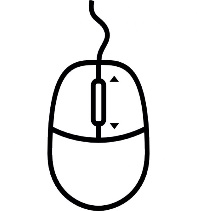
После запуска программы на экране появится игровая сцена и начнётся облет камеры с BGM (фоновой музыкой).

Управление:

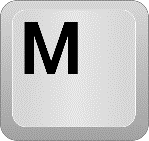
* Левая кнопка мыши: выбор (Рис. 1)

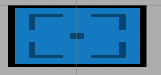
(Рис. 1)

* Кнопка WASD: перемещение главной камеры (Рис. 2а). Скролл мышки увеличение и уменьшение камеры. (Рис. 2б)

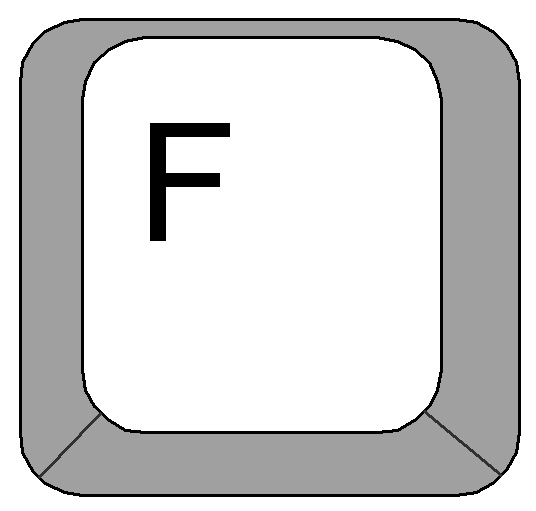
(Рис. 2а) (Рис. 2б)

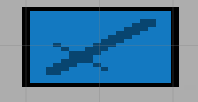
* Включение сетки для передвижение юнита, кнопка M (Рис. 3а) или кнопка на интерфейсе (Рис.3б)

(Рис. 3а) (Рис. 3б)

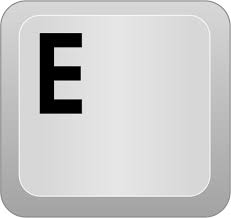


* Включение атаки, кнопка F (Рис. 4) или нажатие кнопки на игровом интерфейсе (Рис. 5)

(Рис. 4) (Рис.5)



* Переключение оружия кнопки Q и E(Рис.6,7) или иконки оружий на игровом интерфейсе (Рис. 8а, 8б, 8в, 9)

(Рис. 6) (Рис. 7)

(Рис. 8а) (Рис. 8б)

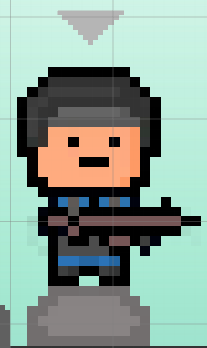
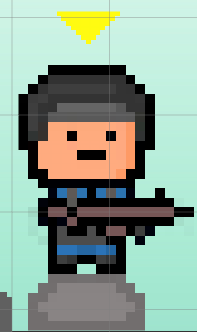


(Рис. 8в) (Рис. 9)



* Выбор юнита происходит через нажатие 1-4 на клавиатуре или при прямом нажатии на персонажа, при выборе над героем загорается желтая иконка (Рис. 10), если персонаж не выбран иконка над персонажем серого цвета (Рис. 11)

**(**Рис. 10) (Рис.11)



**ЗАКЛЮЧЕНИЕ**

В результате работы над данным курсовым проектом была разработана игровая программа "RaibowSX" и документация к ней.

В процессе работы над программой были изучены литературные и интернет источники, развиты навыки работы с компонентами среды Visual Studio 2015, языка программирования С#, а так же Unity и ее возможностей при создании компьютерных игр.

Компьютерная игра имеет простой интерфейс, удобна в использовании, в нее можно поиграть в свободное время, приукрасить свой досуг.

**СПИСОК ИСТОЧНИКОВ**

* Unity documentation
* Unity Game Development Essentials
* Джозеф Албахари - C# 6.0. Справочник. Полное описание языка
* Unity 3D Game Development by Example (Beginner's Guide)

**Интернет источники:**

* https://docs.unity3d.com/ru/current/Manual/index.html
* https://habrahabr.ru/
* https://unity3d.com/ru/learn/tutorials/topics/interface-essentials
* https://unity3d.com/ru/learn/tutorials/s/scripting
* https://unity3d.com/ru/learn/tutorials/s/graphics
* https://unity3d.com/ru/learn/tutorials