Государственное образовательное учреждение высшего

профессионального образования

«Московский Государственный Технический Университет имени

Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

КУРСОВОЙ ПРОЕКТ

2Д игра в жанре платформер

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Мацак И. В.

(подпись, дата)

Разработчик проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Велинецкий А. В.

(подпись, дата)

Москва 2020

Оглавление

[Цель 3](#_Toc59723947)

[Основные определения 3](#_Toc59723948)

[Введение 3](#_Toc59723949)

[Основная идея 4](#_Toc59723950)

[Разработка игры 4](#_Toc59723951)

[Выбор технологий 10](#_Toc59723952)

[Заключение 12](#_Toc59723953)

[Список используемых источников 13](#_Toc59723954)

# Цель

Создание 2д игры в среде разработки Unity, используя алгоритмический язык программирования C#.

# Основные определения

**Алгоритмический язык программирования** - формальный язык, используемый для записи, реализации и изучения алгоритмов. В отличие от большинства языков программирования, алгоритмический язык не привязан к архитектуре компьютера, не содержит деталей, связанных с устройством машины.

**Объектно-ориентированное программирование** (ООП) - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

**Unity** — межплатформенная среда разработки компьютерных игр

**Интерфейс** - общая граница между двумя функциональными объектами, требования к которой определяются стандартом. Это совокупность средств, методов и правил взаимодействиямежду элементами [системы.](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)

# Введение

История компьютерных игр начинается в 1940-х и 1950-х годах, когда в академической среде разрабатывались простые игры и симуляции. Компьютерные игры длительное время не были популярны, и только в 1970-х и 1980-х годах, когда появились для широкой публики аркадные автоматы, игровые консоли и домашние компьютеры, компьютерные игры становятся частью поп-культуры. С созданием компьютерной графики и появлением настоящих домашних компьютеров игровая индустрия стремительно выросла. Игры выпускались тысячными тиражами, не считая пиратских копий. Сейчас игровая индустрия является одной из точек опоры, на которых стоит индустрия персональных ЭВМ. Мне бы также хотелось попробовать себя в создании игр, но для того, чтобы игра получила хоть какую-то популярность, она должна выделяться. Это можно сделать несколькими путями: создать уникальную механику, использовать новый визуальный стиль или объединить уже придуманные механики так, чтобы их сочетание порождало нечто новое и неизведанное.

Основная идея

В начале игры персонаж игрока появляется в неизведанном для себя мире, ничего не знающий и не умеющий. Он встречает на своем пути торговца, который говорит ему, что отсюда еще никто не выбирался и существует только 1 способ отсюда выйти, который лежит через запутанные катакомбы. Наш персонаж идет туда и по неопытности сразу попадает в переполох, и когда он уже почти не может дышать, время вдруг идет вспять, и персонаж опять оказывается на изначальном месте и единственное что у него «осталось от прошлой жизни» это его опыт. За ценные знания торговец может предложить персонажу какие-нибудь улучшения.

# Разработка игры

Рассмотрим некоторые функции движка unity:

* Void Start() – вызывается до обновления первого кадра только если скрипт, в котором присутствует данная функция, включен. Для объектов, добавленных на сцену, функция Start() будет вызываться во всех скриптах до функции Update( )
* Void Update() – вызывается 1 раз за кадр
* Void FixedUpdate() - вызывается с частотой фиксированных кадров
* void OnTriggerEnter2D(Collider2D collision) – вызывается каждый раз, когда другой объект типа Collider2D входит в триггер данного объекта
* void OnTriggerExit2D(Collider2D collision) – вызывается каждый раз, когда другой объект типа Collider2D выходит из триггера данного объекта

Программа состоит из 21 класса:

* Класс «Управление персонажем на ПК» (CharacterControllerScript)

- Один из самых важных классов в данной программе. Отвечает за управление персонажа. Остановимся на этом классе поподробнее, рассмотрим некоторые функции и методы класса:

* В функции Start() происходит инициализация компонентов игрового объекта, поведение которого контролирует данный скрипт. Также происходит загрузка данных последнего сохранения персонажа (см. класс Сохранение/загрузка)
* В функции Update() происходит считывание введенных пользователем данных и соответствующая их обработка. Так, если пользователь нажмет на «Пробел» персонаж прыгнет. Здесь же реализованы такие механики, как рывок, движение персонажа, подбор предметов и полет
* Void Flip() – Отвечает за поворот персонажа
* void AutoLoadCharacter() – отвечает за автоматическую загрузку персонажа во время перехода между сценами
* void LoadCharacter() – отвечает за ручную загрузку персонажа при нажатии на соответствующую кнопку в меню, либо *быстрой клавишей* «F9»
* void BuyHpBuff()/BuyDMGBuff() – служат для совершения покупок улучшений в магазине
* В функции void OnTriggerEnter2D(Collider2D collision) идет обработка столкновения персонажа с различными предметами (монетами и шипами)
* Класс «Управление персонажем под Андроид» (CCSAndroid)

- идентичен классу управление персонажем на ПК, за исключением того, что на экране появляются кнопки управления и обрабатывается нажатие на них

Далее под понятием игрок/персонаж понимается объект класса CharacterControllerScript.

* Класс «Затемнение» (Black)
* Отвечает за анимацию затемнения экрана при переходе между локациями
* Класс «Снаряд» (Bullet)

- Отвечает за поведение пули в пространстве, а именно задает ей скорость и обрабатывает столкновения с врагами и окружением

* Класс «Следование Камеры» (CameraFollow2D)

- Отвечает за поведение камеры во время игры. Следит за расположением и поворотами персонажа, чтобы показать больше того, что находится впереди персонажа.

* Класс «Смена Локаций» (ChangeLocation)

- класс, отвечающий за генерацию локации, переход между комнатами, появление или возрождение врагов и особых предметов:

* В функции Start( ) происходит генерация карты уровня
* В функции void TeleportAndSpawn(GameObject Player, string what\_door) происходит перемещение игрока при входе в дверь, а также появление врагов в нужной комнате
* В функции Update() отслеживается возможность выхода игрока за пределы данной комнаты
* Функция void OnTriggerEnter2D(Collider2D collision) обрабатывает взаимодействие игрока с дверьми, то есть выход из комнаты.
* Класс «Место смерти» (DieSpace)

- «страховочный» класс. Если происходит баг или намеренный взлом механик игры и персонаж выходит за пределы карты, то данный класс перемещает его на нужное место

* Класс «Враг» (Enemy)

- полностью описывает поведение врага

* В функции Update() происходит описание его движения в тех или иных случаях. Также отвечает за то, чтобы при смерти врага из него в случайном направлении и со случайной силой выпадали монеты
* В функции void OnTriggerEnter2D(Collider2D collision) происходит обработка столкновения с окружением
* void OnTriggerStay2D(Collider2D collision) – функция обрабатывающая столкновение с игроком и вызывающая функцию void OnEnemyAttack()
* void OnEnemyAttack() – отвечает за нанесение урона по персонажу, а также проигрывание анимации получения урона
* void TakeDamage(int damage) – отвечает за получение урона от игрока
* Класс «Вход» (Entering)

- класс, отвечающий за обработку переходов между сценами. При попадании в триггер и при нажатии клавиши входа, загружает советующую сцену

* Класс «Загрузка» (Loading)

- класс, отвечающий за загрузку сцены

* Класс «Оружие» (Gun)

- класс, отвечающий за обработку положения мыши/джойстика относительно персонажа, а также за произведение выстрела, то есть создания объекта класса Bullet

* Класс «Полоска здоровья» (HealthBar)

- класс, отвечающий за показания анимированной полоски здоровья персонажа

* Класс «Меню» (Menu)

- класс, отвечающий за все меню в игре, кроме настроек. В моей программе меню бывает 2-х типов: начальное (при входе в игру) и внутриигровое (вызывается на нажатие кнопки «Esc» в самой игре).

* На начальном меню существует 3 кнопки: играть, настройки и выход. При нажатии на кнопку играть вызывается функция void GoToMain(), которая загружает начальную сцену
* При нажатии на кнопку настроек происходит появление меню настроек (см. класс Настройки)
* При нажатии на выход вызывается функция void QuitGame() которая закрывает игру

Во внутриигровом меню, помимо кнопок вызова настроек, возврата назад в игру и выхода из игры, есть кнопки сохранения и загрузки игры, которые вызывают функции SaveGame() и LoadGame() соответственно (см. класс Сохранение/загрузка)

* Класс «Атака персонажа» (PlayerAttack)

- класс, отвечающий за атаку персонажа.

* В функции Update() осуществляется поиск объекта типа Enemy в определенной области перед персонажем. Если объект найден и нажата кнопка атаки, производится нанесение урона врагу, а точнее вызов функции void TakeDamage(int damage) о которой я писал ранее
* Класс «Сохраненные данные» (SaveData)

- класс, содержащий в себе все нужные данные о персонаже, а именно, здоровье, урон, монеты и позицию

* Класс «Сохранение/загрузка» (SaveLoad)

- класс, отвечающий за сохранение и загрузку данных

* static void SaveGame(CharacterControllerScript character) – создает или открывает бинарный файл и записывает в него данные из объекта типа SaveData
* static SaveData LoadGame() – читает данные из файла сохранения и возвращает тип SaveData
* static void AutoSaveGame(CharacterControllerScript character)/ static SaveData AutoLoadGame() – аналогичные функции, только они вызываются без команды пользователя. Например, при переходе между сценами нужно сохранить текущие данные персонажа, а затем при входе в следующую сцену снова их загрузить
* Класс «Настройки» (Settings)

- класс, отвечающий за меню настроек. Среди них – настройка разрешения экрана, громкости звука, настройка качества изображения, а также включение/выключение полноэкранного режима. Также в меню присутствует кнопка подтверждения настроек и возврата в меню.

* Класс «Вектор значений» (VectorValue)

- класс, хранящий положение предыдущего входа в пространстве. Это используется для того, чтобы при переходе между сценами запомнить положение откуда вошел персонаж, чтобы при выходе появиться из этого же положения

# Выбор технологий

В настоящее время существует большое количество инструментов для разработчика игр, для исследования был выбран один из самых новых и мощных движков – Unity 3D. У данного инструмента существуют как преимущества, так и недостатки. Поговорим о преимуществах данного игрового движка. Среди них можно отметить возможномть:

* моделировать физическую среду
* моделировать карты нормалей
* моделировать преграждение окружающего света в экранном пространстве (SSAO)
* моделировать динамические тени и т. д.

Данный функционал существует во многих игровых движках, преимущества Unity в том, что в нем есть чрезвычайно производительный визуальный процесс и мощная межплатформенная поддержка. Визуальный редактор Unity значительно упрощает взаимодействие объектов на сцене и процесс сборки проекта. Особенно полезен этот редактор при разработке проектов с последовательным улучшением, например, в циклах создания прототипов или тестирования. Дополнительным преимуществом платформы Unity является присутствие сильной межплатформенной поддержки, которая подразумевает собой возможность создания и развертывания игры на различных платформах.

Имея большое количество преимуществ, в Unity присутствуют и недостатки основным из которых является отсутствие возможности обращения к внешним библиотекам кода.

Немного поговорим об интерфейсе и возможностях Unity

Компонентами интерфейса Unity являются:

* вкладка Scene (редактор текущей сцены)
* вкладка Game (показывает, что будет видно или видно при запуске игры)
* вкладка Hierarchy (показывает список текущих объектов на сцене)
* панель Inspector (отображает информацию о выделенном в данный момент объекте)
* вкладка Project (показывает текущий репозиторий проекта)
* вкладка Console (используется для вывода данных об ошибках, либо информации от Debug, очень полезно при тестировании игры)

Также немаловажным выбором является выбор языка программирования. В Unity поддерживаются всего 2 языка программирования JavaScript и C#. Я выбрал язык программирования C#, т. к. уже знаком с Си-подобным языком С++ и мне было проще его осваивать, знаю синтаксис. Еще стоит упомянуть, что все это работает на базе платформы [.NET Framework](https://shwanoff.ru/dotnet-framework/). Что это означает? Написанный тобой код на языке C# транслируется в промежуточный язык (IL), который в свою очередь уже преобразуется в машинный код на твоем компьютере прямо во время выполнения приложения (JIT).

# Заключение

В процессе выполнения курсового проекта был частично изучен язык программирования C#, были освоены принципы объектно-ориентированного программирования, а также был изучен игровой движок Unity3D. Результатом работы стала 2Д игра в жанре платформер. В игре присутствуют такие механики, как: передвижение персонажа, стрельба, рывок, полет, процедурная генерация уровней, магазин, несколько типов врагов и save and load system.

# Список используемых источников

1. Документация: документация по C#. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>
2. Документация: Unity user manual. URL: https://docs.unity3d.com/Manual/index.html