Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

Югорский государственный университет

Институт цифровой экономики

КУРСОВАЯ РАБОТА

По дисциплине «Языки программирования высокого уровня»

«Разработка 2D платформера-головоломки на движке Unity3D»

Студент гр. 1192б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Б.Г.Проскуряков/

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Е.И.Сафонов/

(подпись)

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Ханты-Мансийск

2020 г.

# Реферат

Курсовой проект, 27 страниц; 19 рисунков; 2 источника; 2 приложения

Перечень ключевых слов:

C#, Unity, Unity3D, VisualStudio, ООП, Платформер, Разработкаигры

Объектом курсового проекта является получение практических навыков по разработке игр

Цель работы: Разработка 2D платформера-головоломки на движке Unity3D

В результате данного проекта была разработана игра для ПК на языке программирования C# с использованием ООП

# Содержание

[Термины и определения 5](#_Toc60168032)

[Перечень сокращений и обозначений 6](#_Toc60168033)

[Введение 7](#_Toc60168034)

[1. Игровой движок Unity3D 8](#_Toc60168035)

[1.1. Описание 8](#_Toc60168036)

[1.2. Преимущества 8](#_Toc60168037)

[1.2.1. Простота использования 8](#_Toc60168038)

[1.2.2. Uniny Asset Store 9](#_Toc60168039)

[1.2.3. Кроссплатформенность 9](#_Toc60168040)

[1.2.4. Сеттинг 9](#_Toc60168041)

[1.2.5. Доступность 10](#_Toc60168042)

[1.3. Недостатки 10](#_Toc60168043)

[1.3.1. Узконаправленность языка программирования 10](#_Toc60168044)

[1.3.2. Оптимизация 10](#_Toc60168045)

[1.3.3. Потребление памяти 10](#_Toc60168046)

[2. Разработка программы 11](#_Toc60168047)

[2.1. Технические и программные средства 11](#_Toc60168048)

[2.2. Цель игры 11](#_Toc60168049)

[2.3. Реализация механик игры 13](#_Toc60168050)

[2.3.1. Передвижение игрока 13](#_Toc60168051)

[2.3.2. Зона смерти 14](#_Toc60168052)

[2.3.3. Куб убийца 15](#_Toc60168053)

[2.3.4. Падающие платформы 16](#_Toc60168054)

[2.3.5. Главное меню 17](#_Toc60168055)

[2.3.6. Пауза в игре 18](#_Toc60168056)

[2.3.7. Финиш и победа в игре 19](#_Toc60168057)

[2.3.8. Таймер и поражение 20](#_Toc60168058)

[Заключение 22](#_Toc60168059)

[Список используемых источников 23](#_Toc60168060)

[Приложение А 24](#_Toc60168061)

[ПриложениеБ 28](#_Toc60168062)

# Термины и определения

В настоящем отчёте о разработка 2D платформера-головоломки на движке Unity3D применяют следующие термины с соответствующими определениями.

Box Collider2D – это коллайдер, используемый при работе с 2D физикой в Unity

Canvas для Unity UI – это область, внутри которой размещаются все элементы пользовательского интерфейса.

OnTriggerEnter2D – срабатывает, когда другой объект входит вколлайдер, прикрепленный к этому объекту (только 2D-физика)

Rigidbody – это компонент, обеспечивающий игровым объектам взаимодействие с помощью физики

Ассеты – это компоненты, которые представляют собой графику, звуковое сопровождение или скрипты. Они прикрепляются к объектам и составляют важную часть игры

Кликер – это компьютерная игра, игровой процесс в которой состоит из выполнения простых действий, таких как многократное нажатие на экран

Коллайдеры – это компоненты, встроенные в Unity, которые обеспечивают обнаружение столкновений с использованием их различных «ограничивающих рамок»

Пиксельная игра– это игра, графика в которой специально создана с помощью пикселей, и стилизована под 8- или 16-битную картинку

Платформер – жанр компьютерных игр, в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня

Префаб – это особый тип ассетов, позволяющий хранить весь GameObject со всеми компонентами и значениями свойств.

Респаун — в компьютерных играх место постоянного появления какого-либо объекта или персонажа игрового мира, происходящее в определённой точкеигрового пространства

Сеттинг –это среда, в которой происходит действие

Скрипт – это последовательность действий, описанных с помощью скриптового языка программирования для автоматического выполнения определенных задач

Тег – именованная метка

# Перечень сокращений и обозначений

В настоящем отчете о разработка 2D платформера-головоломки на движке Unity3D применяют следующие сокращения и обозначения.

AR – Дополненная реальность

UI – Пользовательский интерфейс

VR – Виртуальная реальность

ИИ – Искусственный интеллект

ООП – Объектно-ориентированное программирование

ОС – Операционная система

ПК – Персональный компьютер

# Введение

Целью данной курсовой работы является разработка 2D игры на движке Unity3D.

В разделе «Игровой движок Unity3D»представлено описание игрового движка, его преимущества и недостатки.

В разделе «Разработка программы» производится разработка программы, описание оборудования и программного обеспечения для ее реализации.

# Игровой движок Unity3D

## Описание

Unity3D– это кроссплатформенный игровой движок, изначально выпущенный Unity3D Technologies в 2005 году. Основное внимание Unity3D уделяется разработке как 2D, так и 3D игр и интерактивного контента. Unity3D теперь поддерживает более 20 различных целевых платформ для развертывания, в то время как его наиболее популярными платформами являются системы ПК, Android и iOS.

Unity3D предоставляет полный набор инструментов для проектирования и создания игр, включая интерфейсы для графики, аудио и инструменты для создания уровней, требующие минимального использования внешних программ для работы над проектами. [1]

Вот так выглядит стартовый экран приложения, при первом запуске проекта (Рисунок 1).

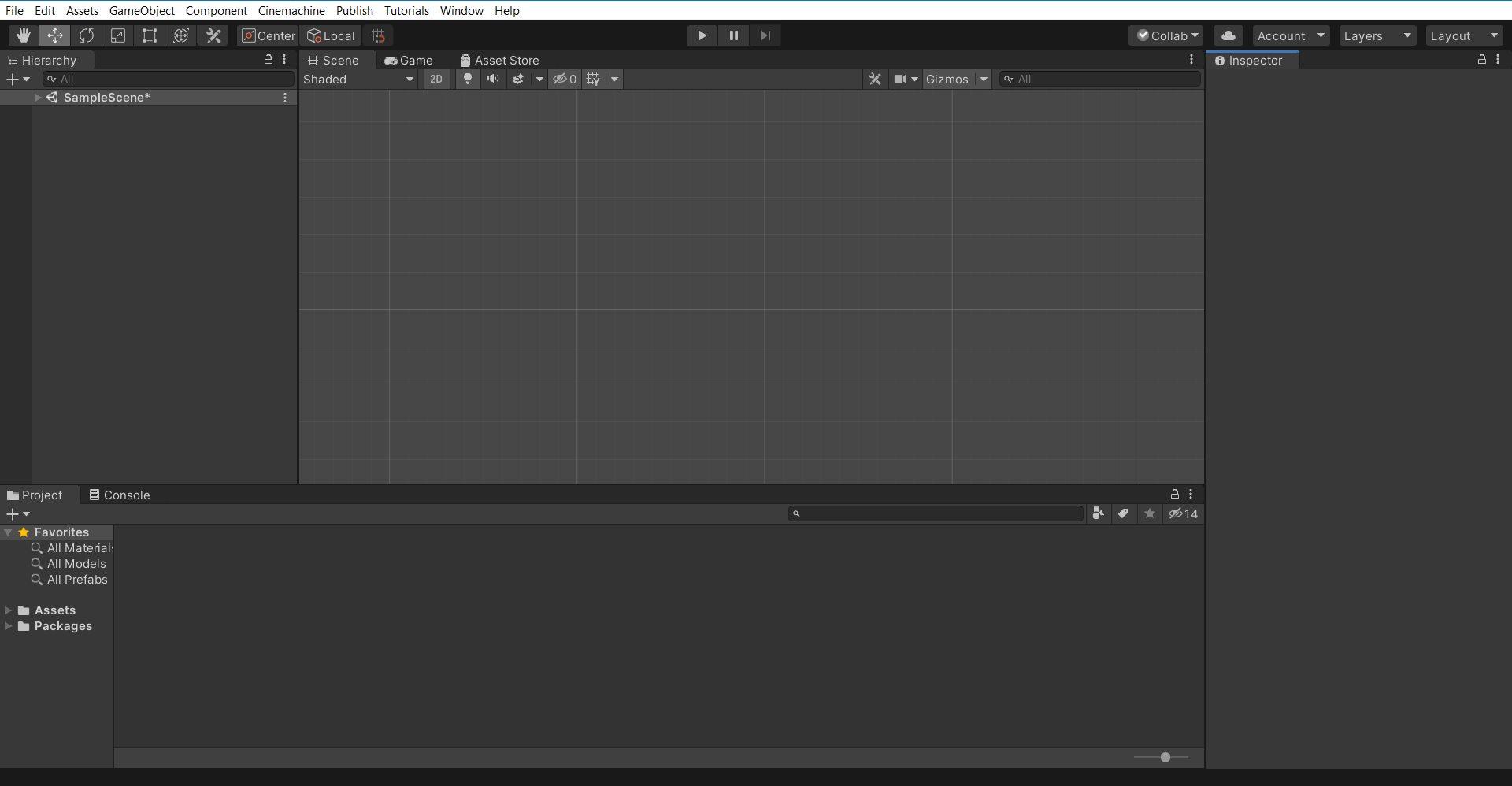


Рисунок 1 – Стартовый экран Unity3D

## Преимущества

### Простота использования

Движок Unity3D дает возможность разрабатывать игры, не требуя для этого каких-то особых знаний. Здесь используется компонентно-ориентированный подход, в рамках которого разработчик создает объекты (например, главного героя) и к ним добавляет различные компоненты (например, визуальное отображение персонажа и способы управления им). Благодаря удобному Drag&Drop интерфейсу и функциональному графическому редактору движок позволяет рисовать карты и расставлять объекты в реальном времени и сразу же тестировать получившийся результат. [2]

### Uniny Asset Store

У компании Unityприсутствует огромная библиотека ассетов и плагинов, с помощью которых можно значительно ускорить процесс разработки игры. Их можно импортировать и экспортировать, добавлять в игру целые заготовки – уровни, врагов, паттерны поведения ИИ и так далее. Никакой возни с программированием. Многие ассеты доступны бесплатно, другие предлагаются за небольшую сумму, и при желании можно создавать собственный контент, публиковать его в Unity AssetStore и получать от этого прибыль.[2]

### Кроссплатформенность

Unity 3D поддерживает огромного количества платформ, технологий, API. Созданные на движке игры можно легко портировать между ОС Windows, Linux, OSX, Android, iOS, на консоли семейств PlayStation, Xbox, Nintendo, на VR- и AR-устройства. Unity поддерживает DirectX и OpenGL, работает со всеми современными эффектами рендеринга, включая новейшую технологию трассировки лучей в реальном времени.[2]

### Сеттинг

Физика твердых тел, ragdoll и тканей, система Level of Detail, коллизии между объектами, сложные анимации – все это можно реализовать силами движка. Стереотипное мнение о том, что движок пригоден только для небольших инди-игр и неспособен выдавать красивую картинку, давно уже не актуально: достаточно посмотреть технодемо ADAM, The Blacksmith и Book of the Dead от создателей среды Unity, чтобы убедиться в ее выдающихся способностях.[2]

### Доступность

Unity доступен бесплатно, что открывает перед независимыми разработчиками дверь в игровую индустрию. Конечно, существуют ограничения: бесплатная версия движка демонстрирует лого Unity перед запуском игры, а проект, созданный с ее помощью, не должен приносить разработчику больше $100 тысяч в год. Впрочем, тарифы на подписку не опустошат кошельки даже начинающей команды: Про-версия стоит $125 в месяц, что не так уж много в сравнении с другими движками, причем базовая версия содержит ровно тот же функционал, что и профессиональная.[2]

## Недостатки

### Узконаправленность языка программирования

Если команда захочет разработать что-нибудь сложнее простого кликера или платформера, то ей придется искать хорошего программиста на C#, который напишет скрипты и компоненты, внедрит их в игру и заставит работать. [2]

### Оптимизация

Создание масштабных и сложных сцен с множеством компонентов может негативно повлиять на производительность игры, в результате чего разработчикам придется потратить дополнительное время и ресурсы на оптимизацию, а возможно – и удаление некоторых элементов из проекта. [2]

### Потребление памяти

Приложения, созданные на Unity, довольно «тяжеловесны»: даже самая простая пиксельная игра может занимать несколько сотен мегабайт на ПК. Конечно, для жесткого диска компьютеров это небольшой объем, но, если проект разрабатывается и для мобильных платформ, то следует задуматься об оптимизации его размера.[2]

# Разработка программы

## Технические и программные средства

Разработка выполнялась на персональном ноутбуке с данными характеристиками:

1. Операционная система: Windows 10 Prox64
2. Процессор: AMD Ryzen 5 4600H with Radeon Graphics 3.00GHz
3. Видеоадаптер: NVIDIA GeForce GTX 1650 Ti 4Gb (128-bit)
4. Оперативная память: 16Gb

Для разработки игры было выбрано следующее программное обеспечение:

1. Программа написана в Visual StudioCommunity 2019
2. Версия C#– 7.3
3. Версия Unity3D–2019.4.11f1 Personal

## Цель игры

Игровым персонажем нужно за отведённое время добраться до финиша, избегая ловушек и преодолевая сложные препятствия на своём пути.

После каждой смерти уровень не начинается заново и время на таймере не сбрасывается, также как и не перезагружается сцена, но игрок, мгновенно после смерти, перемещается на место старта.

Для того, чтобы перезапустить уровень полностью (сбросить счётчик таймера и возобновить физику некоторых ловушек), нужно нажать на кнопку перезапуска уровня (Рисунок 2), которая находится в правомверхнем углу экрана.



Рисунок 2 – Кнопка перезагрузки уровня

В игре также есть главное меню, в которое игрок попадает сразу после входа в игру (Рисунок 3)



Рисунок 3 – Главное меню

После нажатия кнопки «Начать игру», игрок появляется на сцене первого уровня (Рисунок 4).

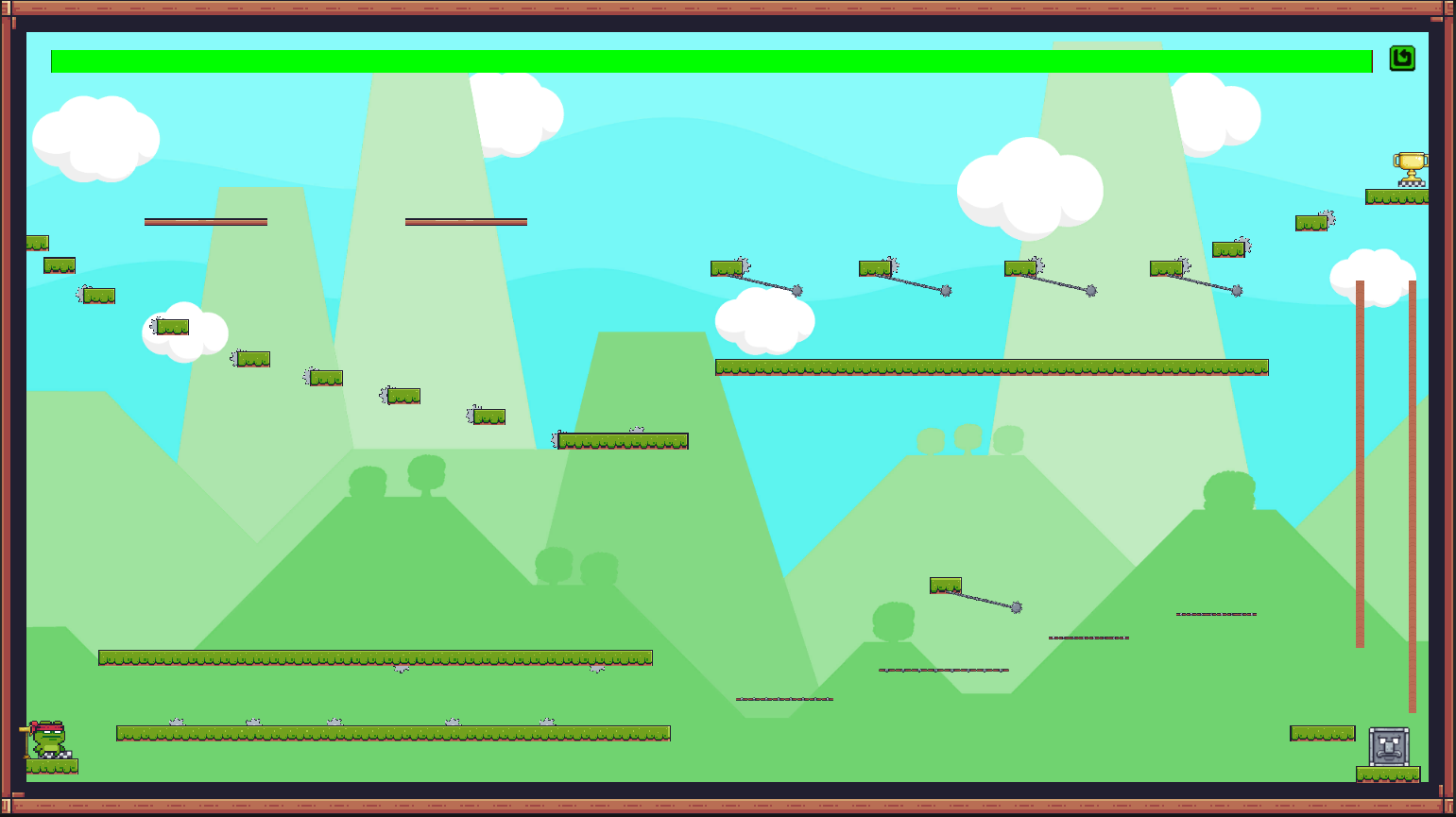


Рисунок 4 – Старт игры

## Реализация механик игры

### Передвижение игрока

За бег персонажа отвечает метод «Run» в котором сначала задаётся направление передвижения, в моём случаи, игрок может бегать только по горизонтали, то есть вправо или влево. После чего, идет анимация поворота, то есть если сначала персонаж бежит вправо, тогда и персонаж смотрит вправо и выполняет анимацию в указанную сторону, а как только игрок нажмет клавишу влево, то спрайт персонажа сразу зеркально отразится в указанную сторону. В конце идёт присваивание скорости передвижения абсолютно твёрдому телу (rigidbody 2D) Сам игровой персонаж представлен на (Рисунке 5).



Рисунок 5 – Игровой персонаж

За прыжок отвечает метод «Jump»в котором задаётся клавиша для прыжка «W». Для механики прыжка используется функция «AddForce», благодаря ей Unity3D сам вычисляет конечную скорость прыжка, на основе массы объекта, гравитации и т.д.

Метод «CheckingGround» нужен, чтобы проверять, на земле ли находится наш персонаж? И предотвращать безграничное количество прыжков, когда персонаж отталкивается от воздуха, а не от платформы.

Бег, поворот, прыжок и проверка возможности прыжка, представлены в листинге кода А.1 в «»

### Зона смерти

Зона смерти присутствует на каждой ловушке в игре, также, если игрок выйдет за пределы сцены (к примеру, упадёт с платформы), то тоже сработает смертельная зона.

При каждой смерти, персонажа будет перемещать на точку респауна, но сцена не будет перезагружаться. Так выглядит место респауна в игре (Рисунок 6)



Рисунок 6 – Место респауна в игре

Реализация зоны смерти происходит следующим образом: как только тег игрока (в нашем случаи он называется «Player») соприкасается с BoxCollider2D, сразу срабатывает OnTriggerEnter2D и персонаж перемещается в точку респауна.

Зона смерти и перемещение на место возрождения представлены в листинге кода 1.

Листингкода 1 – ФайлDieZone.cs

public GameObject respawn;

void OnTriggerEnter2D(Collider2D other){

if (other.tag == "Player")

other.transform.position = respawn.transform.position;

}

### Куб убийца

В данной игре Куб убийца выполняет роль «Лифта», то есть платформы, которая двигается по вертикали (вверх-вниз). На эту платформу нужно будет запрыгнуть и, с её помощью, подняться наверх (Рисунок 7).



Рисунок 7 – Куб убийца

Но если упасть в место под кубом, то Куб убийца убьёт персонажа шипами, которые находятся под ним, на которых, в свою очередь, поставлена «Зона смерти», в следствии чего, игрок возродится на месте респауна (Рисунок 8).

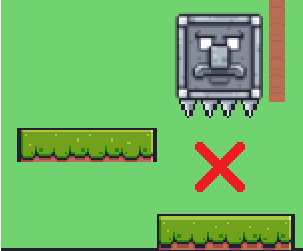


Рисунок 8 – Место смерти от Куба убийцы

Куб убийца движется от одного объекта к другому (в нашем случаи от «MoveingPoint» до «MoveingPoint (1)» и обратно), процесс будет повторятся до тех пор, пока не выйдет время, отведённое на уровень, или пока игрок не дойдёт до финиша. В самом же Unity можно задать определенную скорость передвижение Куба убийцы, за это отвечает переменная «speed».

Траектория передвижение Куба убийцы представлена в листинге кода А.2 в «Приложение А»

### Падающие платформы

В игре присутствуют платформы, которые в случаи контакта с персонажем, спустя несколько секунд, падают, но спустя определённое время поднимаются обратно, чтобы игрок мог несколько раз пытаться проходить по этим платформам без необходимости рестарта уровня (Рисунок 9).



Рисунок 9 – Падающие платформы

Здесь реализована такая же механика, как и с «Зона смерти», то есть, как только тег игрока заденет коллайдер падающей платформы, то Rigidbody, который закреплён на объекте платформы, перейдёт из состояния Кинематики в состояния Динамики, это значит, что на него будет действовать физика, поэтому платформа под действием силы тяжести начнёт падать. Спустя несколько секунд сработает метод BackPlatform(), который вернёт в Rigidbody состояние Кинематики, после чего метод Update() начнёт перемещать платформу вертикально вверх до точки, где она была изначально с заданной скоростью равной 20f.

Проверка на соприкосновение с тегом, падение и восстановление платформ более подробно представлены в листинге кода А.3 в «Приложение А»

### Главное меню

Главное меню, это самая первая сцена, на которую попадет игрок при запуске, уже готовой, игры.

На экране главного меню находятся две кнопки «Начать игру» и «Выйти из игры». Первая кнопка перемещает игрока на сцену с уровнем игры (Рисунок 10).

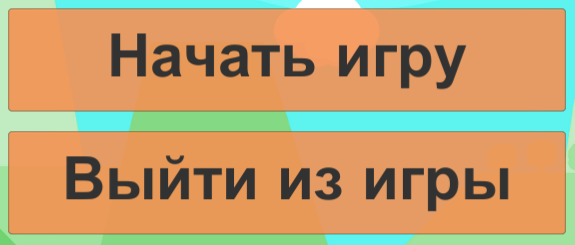


Рисунок 10 – Кнопки главного меню

В свою очередь вторая, полностью закрывает приложение (Работает только в уже готовой игре, если нажать на эту кнопку в самом Unity, то ничего не произойдёт)

Реализация кнопок и переход между сценами представлен в листинге кода 2

Листингкода 2 – ФайлMainMenu.cs

public void StartGame(){

SceneManager.LoadScene("Level1");

Time.timeScale = 1f;

}

public void QuitGame(){

Application.Quit();

}

### Пауза в игре

В игре присутствует меню паузы, которое можно вызвать, нажав на клавиатуре клавишу «ESC». В этот момент скорость игры снижает до значения, равному (0f), это значит, что абсолютно всё в игре останавливается и можно работать только с меню, которое вышло поверх игры (Рисунок 11).

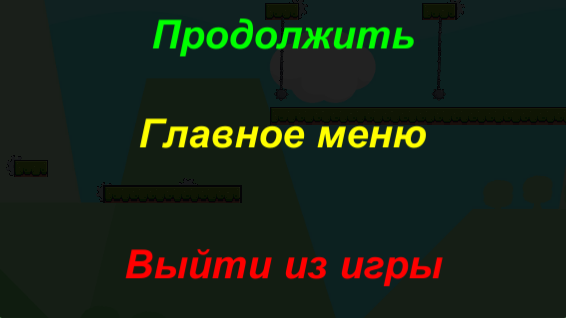


Рисунок 11 – Меню паузы

Само меню представляет собой Canvasс панелью, в которой размещены три кнопки, в моём случаи это кнопки:

«Продолжить» – данная кнопка возвращает нас в игру и восстанавливает её скорость, делая её равной (1f), также можно вернутся в игру повторным нажатием клавиши «ESC».

«Главное меню» – эта кнопка перемещает нас на самую начальную сцену, то есть в главное меню игры, при этом сбрасывая весь прогресс игры.

«Выйти из игры» – кнопка, которая отвечает за выход из уже готовой игры (полностью закрывает приложение).

Изначально меню паузы скрыто и появляется только тогда, когда его вызывают, иерархию меню паузы из редактора в Unity можно увидеть на (Рисунке12).

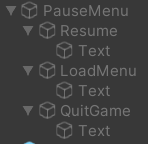


Рисунок 12 – Иерархия меню паузы в редакторе Unity

Где PauseMenu является панелью, а Resume, LoadMenu и QuitGame являются кнопками, к каждой из которых привязан свой текст, функционал которых представлен в листинге кода А.4 в «Приложение А»

### Финиш и победа в игре

Финишем считает BoxCollider 2Dобъекта, как только тег игрока касается коллайдера этого объекта, игра уходит на паузу, а подцентру экрана появляется панель с поздравительным текстом и двумя кнопками (Рисунок 13):



Рисунок 13 – Финиш

«Переиграть уровень» – которая просто перезапустит текущую сцену – листинг кода 3.

Листингкода 3 – ФайлRestartLevel.cs

public void RestartLvl(){

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 0);

Time.timeScale = 1f;

}

«Выйти в главное меню» – которая перенесет нас на начальную сцену, то есть в главное меню.

Изначально победное меню скрыто и появляется только тогда, когда игрок дошел до финиша, синий цвет в редакторе означает, что данные объекты являются префабом, а иерархию меню из редактора в Unity можно увидеть на (Рисунке 14).

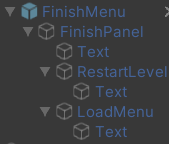


Рисунок 14 – Иерархия меню финиша в редакторе Unity

Реализация финиша представлена в листинге кода А.5 «Приложение А»

### Таймер и поражение

Игру можно как выиграть, так и проиграть, для этого в игру и был добавлен таймер, его можно настроить на любую продолжительность и по истечению времени, он остановит игру и сообщит игроку о том, что он проиграл и предложив ему попробовать ещё раз или выйти в главное меню(Рисунок 15)

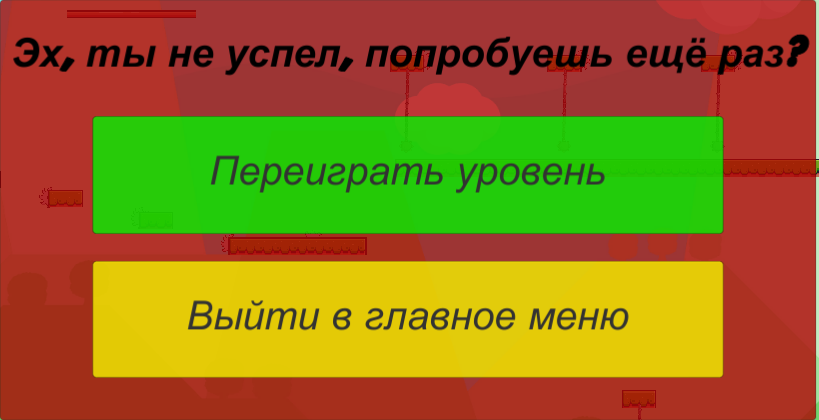


Рисунок 15 – Меню проигрыша

Класс таймера наследуется от финиша, это сделано для того, что лишний раз не писать одинаковый код к кнопкам, так как они точно такие же, как и на меню финиша.

Изначально, меню проигрыша скрыто и появляется только тогда, когда у игрока закончилось время, говоря простыми словами – когда зелёная полосаполностью станет красной (Рисунок 16 и 17).



Рисунок 16 – Зелёная полоса таймера



Рисунок 17 – Красная полоса таймера

Иерархию меню из редактора в Unity можно увидеть на (Рисунке 18).

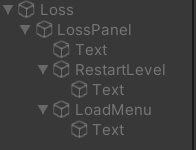


Рисунок 18 – Иерархия меню проигрыша в редакторе Unity

Реализация таймера и меню проигрыша представлены в листинге кода А.6 «Приложение А»

# Заключение

В ходе данной курсовой работы была произведена разработка 2D платформера-головоломки на движке Unity3D, полная иерархия объектов в редакторе Unity представлена в «ПриложениеБ» (Рисунок Б.1).

В процессе выполнения курсовой работы, я впервые работал над подобным проектом, и могу сказать, что инструмент для разработки игры «Unity» показался мне очень приветливым для новичков вроде меня. Мне удалось довольно быстро разобраться с интерфейсом программы и стандартным набором инструментов для работы со спрайтами. Но всё же, хоть я и реализовал все свои задумки в данном проекте, впечатлений оказались не очень радостными, ибо я считаю, что для создания каких бы то ни было игр, должно быть минимум 2 участника, один из которых должен быть геймдизайнером, иначе приходится либо пользоваться чужими спрайтами, которые не всегда придутся разработчику по вкусу, либо же придётся заказывать спрайты и тратить кучу времени на объяснения художнику своих хотелок.

Разработанная видеоигра может быть применена в качестве самостоятельного коммерческого проекта.

# Список используемых источников

1. CoderLessons [электронный ресурс]/ Единство - введение/ Бавария – Мюнхен: 2019. - . - режим доступа: https://coderlessons.com/tutorials/noveishie-tekhnologii/uchis-edinstvu/unity-kratkoe-rukovodstvo, свободный
2. Cubic [электронный ресурс]/ Движок Unity – особенности, преимущества и недостатки/ Амстердам: 2017.- . - режим доступа https://cubiq.ru/dvizhok-unity/, свободный

## Приложение А

Листинг кода A.1 – Файл PlayerMove.cs

public class PlayerMove :MonoBehaviour

public Rigidbody2D rb;

public Vector2 moveVector;

public float speed = 2f;

public Animator anim;

public SpriteRenderersr;

public float JumpForse = 7f;

public bool onGround;

public Transform GroundCheck;

public float checkRadius = 0.4f;

public LayerMask Ground;

void Start(){

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

sr = GetComponent<SpriteRenderer>();

}

void Update(){

Run();

Flip();

Jump();

CheckingGround();

}

void Run(){

moveVector.x = Input.GetAxis("Horizontal");

anim.SetFloat("MoveX", Mathf.Abs(moveVector.x));

rb.velocity = new Vector2(moveVector.x \* speed, rb.velocity.y);

}

void Flip(){

if (moveVector.x> 0)

sr.flipX = false;

if (moveVector.x< 0)

sr.flipX = true;

}

void Jump(){

if (Input.GetKeyDown(KeyCode.W) &&onGround)

rb.AddForce(Vector2.up \* JumpForse);

}

void CheckingGround(){

onGround = Physics2D.OverlapCircle(GroundCheck.position, checkRadius, Ground);

anim.SetBool("onGround", onGround);

}

Листинг кода А.2 – Файл KillerCube.cs

private bool i;

public Transform target1;

public Transform target2;

public float speed;

void Start(){

i = true;

}

void Update(){

if (i == true){

transform.position = Vector3.MoveTowards(transform.position, target1.position, speed \* Time.deltaTime);

if (transform.position == target1.position){

i = false;

}

}

if (i == false){

transform.position = Vector3.MoveTowards(transform.position, target2.position, speed \* Time.deltaTime);

if (transform.position == target2.position){

i = true;

}

}

}

Листинг кода А.3 – Файл FallingPlatform.cs

Rigidbody2Drb;

Vector2 currentPosition;

bool moveingBack;

void Start(){

rb = GetComponent<Rigidbody2D>();

currentPosition = transform.position;

}

void Update(){

if(moveingBack == true){

transform.position = Vector2.MoveTowards(transform.position, currentPosition, 20f \* Time.deltaTime);

}

if(transform.position.y == currentPosition.y){

moveingBack = false;

}

}

private void OnCollisionEnter2D(Collision2D collision){

if (collision.gameObject.tag.Equals("Player") &&moveingBack == false){

Invoke("FallPlatform", 1f);

}

}

void FallPlatform(){

rb.isKinematic = false;

Invoke("BackPlatform", 3f);

}

void BackPlatform(){

rb.velocity = Vector2.zero;

rb.isKinematic = true;

moveingBack = true;

}

Листинг кода А.4 – Файл PauseMenu.cs

public static bool GameIsPaused = false;

public GameObject PauseMenuUI;

void Update(){

if (Input.GetKeyDown(KeyCode.Escape)){

if (GameIsPaused){

Resume();

}

else{

Pause();

}

}

}

public void Resume(){

PauseMenuUI.SetActive(false);

Time.timeScale = 1f;

GameIsPaused = false;

}

void Pause(){

PauseMenuUI.SetActive(true);

Time.timeScale = 0f;

GameIsPaused = true;

}

public void LoadMenu(){

SceneManager.LoadScene("MainMenu");

Resume();

}

public void QuitGame(){

Application.Quit();

}

Листинг кода А.5– Файл Finish.cs

public class Finish :RestartLevel{

public static bool FinishGame = false;

public GameObject FinishScreen;

private void OnTriggerEnter2D(Collider2D Finish){

if(Finish.tag == "Player"){

FinishScreen.SetActive(true);

Time.timeScale = 0f;

FinishGame = true;

}

else{

FinishScreen.SetActive(false);

}

}

public void LoadMenu(){

SceneManager.LoadScene("MainMenu");

Time.timeScale = 1f;

}

}

Листинг кода А.6 – Файл Timer.cs

public class Timer : Finish

{

Image timerBar;

public float maTime = 20f;

float timeLeft;

public GameObject LossScreen;

void Start(){

timerBar = GetComponent<Image>();

timeLeft = maTime;

Time.timeScale = 1f;

}

void Update(){

if (timeLeft>0){

timeLeft -= Time.deltaTime;

timerBar.fillAmount = timeLeft / maTime;

LossScreen.SetActive(false);

}

else{

LossScreen.SetActive(true);

Time.timeScale = 0f;

}

}

}

## ПриложениеБ

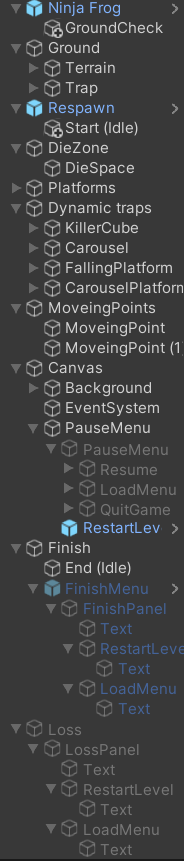


Рисунок Б.1 – Полная иерархия объектов в редакторе Unity.