МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Энергетический факультет
Кафедра информатики вычислительной техники и прикладной математики

КУРСОВОЙ ПРОЕКТ

По дисциплине: Интерактивные графические системы

На тему: Разработка интерактивной графической системы

Выполнил студент группы ИВТ – 17 Демидов Владимир Юрьевич

Руководитель работы: старший преподаватель, Долгих Р.С.

Чита
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Энергетический факультет
Кафедра информатики вычислительной техники и прикладной математики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

По дисциплине: Интерактивные графические системы

На тему: Разработка интерактивной графической системы

Выполнил студент группы ИВТ – 17 Демидов Владимир Юрьевич

Руководитель работы: старший преподаватель, Долгих Р.С.

Чита
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Энергетический факультет
Кафедра информатики вычислительной техники и прикладной математики

ЗАДАНИЕ
на курсовой проект

По дисциплине: Интерактивные графические системы

Студенту: Демидову Владимиру Юрьевичу

Специальности 09.03.01 Информатика и вычислительная техника

1 Тема курсового проекта: Разработка интерактивной графической системы

2 Срок подачи студентом законченной работы: 13.01.2021 г.

3 Исходные данные к работе: литературные источники, источники в сети интернет

4 Перечень подлежащих разработке в курсовой работе вопросов:

– Постановка и анализ задачи;

– Выбор средств реализации;

– Реализация игры.

Дата выдачи задания

«9» сентября 2020 г.

Руководитель курсовой работы _____ / Долгих Р.С./
(подпись, расшифровка подписи)

Задание принял к исполнению

«___» _____ 20__ г.

Подпись студента _____ / В.Ю. Демидов/

РЕФЕРАТ

Пояснительная записка – 29 страниц, 24 – рисунков, 8 источников, 0 таблиц.

ИГРА, ПЛАТФОРМЕР, UNITY, КЛАСС, АЛГОРИТМ, ИГРОК, УРОВЕНЬ, МЕТОД, ПРОЦЕДУРА, СКРИПТ.

В данной курсовой работе пошагово рассматриваются этапы создания интерактивной графической системы (игры) при помощи среды разработки Unity, с использованием высококровного языка программирования C#. Этапы включают в себя: создание простых объектов, создание взаимодействия между объектами при помощи скриптов, реализация анимации, переходов и меню, и завершительная часть – сбор всего в полноценную систему (игру).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С ростом и развитием компьютерных технологий востребованность красивых, лаконичных и интуитивно понятных интерактивных графических систем неуклонно растет. Связано это с тем, что графические системы просты в использовании и понимании для обычных пользователей благодаря разделению сложных системных частей от интерфейсных.

Существует большое множество интерактивных графических систем, одна из наиболее интересных – компьютерные игры (сфера развлечений). Именно эту тему затрагивает данная курсовая работа.

Создание компьютерной игры – сложный процесс, требующий четкого понимания принципов создания таких графических систем. Однако, существуют инструменты, позволяющие значительно упростить и упорядочить создание игр. Один из лучших подобных инструментов – среда разработки (игровой движок) Unity.

1. Постановка и анализ задачи

В данной курсовой работе в качестве интерактивной графической системы была представлена идея игры, в которой игрок управляет шаром, маневрируя между препятствиями по узкой дороге, стараясь дойти до финиша.

Подобный жанр игр называется «аркада». Игры, в которых игроку приходится действовать быстро, полагаясь в первую очередь на свои рефлексы и реакцию. Аркады характеризуются развитой системой бонусов: начисление очков, постепенно открываемые элементы и этапы игры, и т.д.

Термин «аркада» по отношению к компьютерным играм возник во времена игровых автоматов, которые устанавливались в торговых галереях (arcades). Игры на них были простыми в освоении для привлечения большего количества игроков. Впоследствии эти игры перешли на игровые приставки (консоли) и до сих пор являются основным жанром на них.

Из термина становится ясно, что игра будет простой, но от этого задача ее создания не становится легче. Хоть и реализация проста, потенциал у подобной игры огромен. Можно делать большое количество уровней, и не обязательно однонаправленных, или даже одноэтажных. Можно добавить двигающиеся препятствия, даже живых врагов, задача которых – съесть игрока. Также есть потенциал развития системы рекордов и достижений, за которые доступны секретные уровни, или даже новые игровые фигуры.

Для простоты реализации интерактивной графической системы, будет реализована лишь основная часть идеи – несколько уровней, физика и система баллов (счет).

2. Выбор средств реализации

В данной курсовой работе компьютерная игра – аркада реализована на игровом движке Unity.

Unity –межплатформенная среда разработки компьютерных игр. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

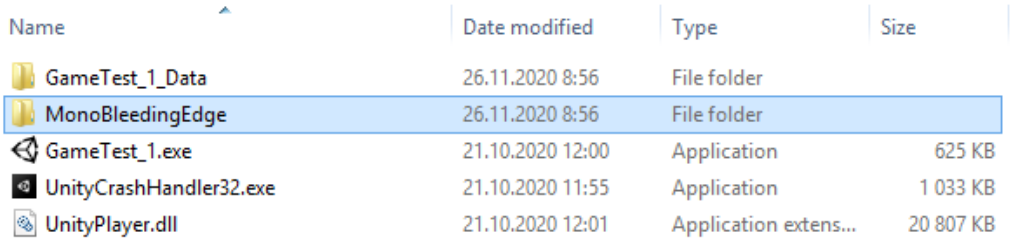
На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями и инди разработчиками.

В Unity можно писать код на языках Java и C#. В рамках курсовой работы было решено использовать язык C#.

Unity – мощный инструмент для создания приложений, в который включено очень много инструментов, таких как: иерархическое дерево всех компонентов сцены, редактор сцены, превью игры, редакторы анимации, и многие другие. В совокупности они позволяют создавать впечатляющие проекты за относительно короткие сроки.

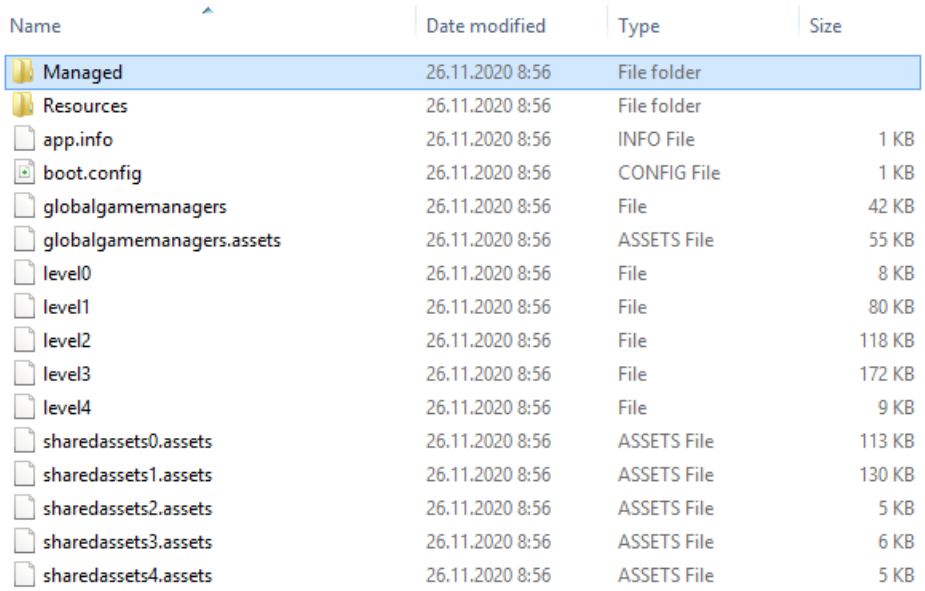
3. Организация данных на физическом уровне

Данные, используемые программой, представлены в соответствии с рисунком 1.



Name	Date modified	Type	Size
GameTest_1_Data	26.11.2020 8:56	File folder	
MonoBleedingEdge	26.11.2020 8:56	File folder	
GameTest_1.exe	21.10.2020 12:00	Application	625 KB
UnityCrashHandler32.exe	21.10.2020 11:55	Application	1 033 KB
UnityPlayer.dll	21.10.2020 12:01	Application extens...	20 807 KB

Рисунок 1 – данные, используемые программой



Name	Date modified	Type	Size
Managed	26.11.2020 8:56	File folder	
Resources	26.11.2020 8:56	File folder	
app.info	26.11.2020 8:56	INFO File	1 KB
boot.config	26.11.2020 8:56	CONFIG File	1 KB
globalgamemanagers	26.11.2020 8:56	File	42 KB
globalgamemanagers.assets	26.11.2020 8:56	ASSETS File	55 KB
level0	26.11.2020 8:56	File	8 KB
level1	26.11.2020 8:56	File	80 KB
level2	26.11.2020 8:56	File	118 KB
level3	26.11.2020 8:56	File	172 KB
level4	26.11.2020 8:56	File	9 KB
sharedassets0.assets	26.11.2020 8:56	ASSETS File	113 KB
sharedassets1.assets	26.11.2020 8:56	ASSETS File	130 KB
sharedassets2.assets	26.11.2020 8:56	ASSETS File	5 KB
sharedassets3.assets	26.11.2020 8:56	ASSETS File	6 KB
sharedassets4.assets	26.11.2020 8:56	ASSETS File	5 KB

Рисунок 2 – содержимое папки GameTest_1_Data

В папке GameTest_1_Data хранятся: информация об уровнях (в каждом уровне свой набор переменных для объектов игры), а также некоторые системные файлы, такие как точка входа в процедуру, глобальные файлы игрового управления и т.п.

В папке GameTest_1_Data/Managed хранятся второстепенные библиотеки и ресурсы среды Unity.

В папке GameTest_1_Data/Resources содержатся ресурсы проекта.

В папке MonoBleedingEdge хранятся необходимые для полноценной работы приложения библиотеки C# и MonoDevelop.

4. Реализация

На первом шаге в Unity необходимо создать несколько фигур.

Фигуры:

1. Шар;
2. Огромный прямоугольник (плоскость для шара и препятствий);
3. Прямоугольник – препятствие;
4. Квадрат – препятствие;
5. Цилиндр – препятствие.

В Юнити уже заложены инструменты для создания базовых геометрических фигур.

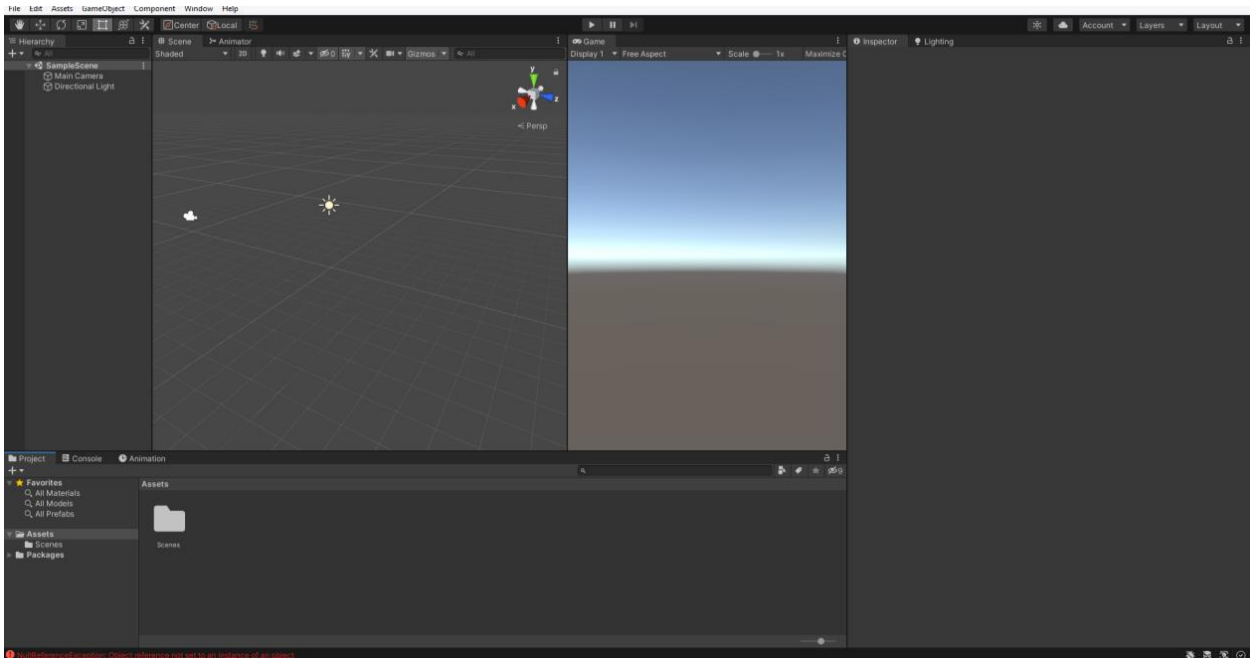


Рисунок 3 – Интерфейс Юнити

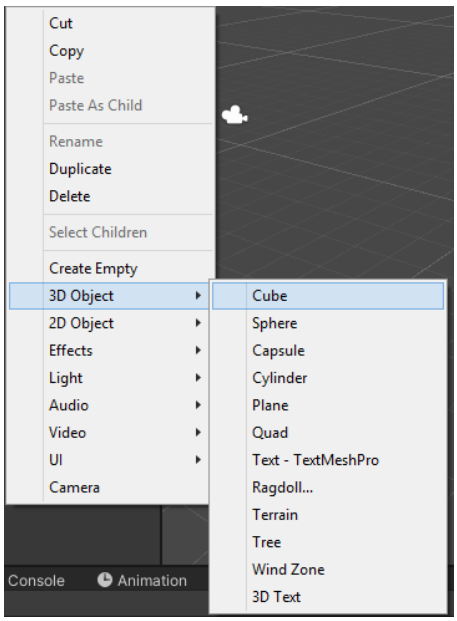


Рисунок 4 – Меню простых 3D объектов

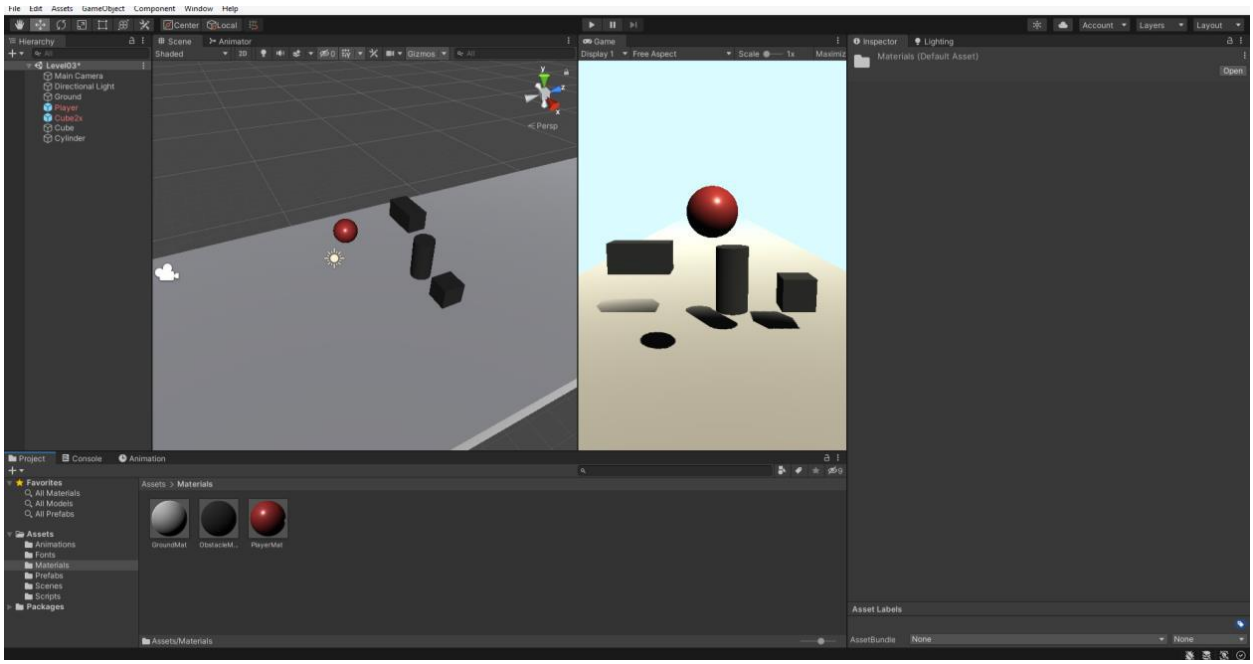


Рисунок 5 – Все 5 объектов

В Unity существует множество инструментов для создания игр. Для создания пяти фигур использовались: иерархическое дерево всех компонентов сцены, редактор сцены, превью игры, редакторы анимации, набор шейдеров и материалов по умолчанию.

Следующий этап – создание взаимодействия между объектами. Любое взаимодействие между объектами отсутствует. Они просто висят в воздухе

даже после запуска игры. Для реализации взаимодействия между объектами необходимо подключить базовые компоненты Юнити: гравитацию, коллайдеры.

Компоненты для фигур:

1. Шар – гравитация, коллайдер шара;

2. Огромный прямоугольник (плоскость для шара и препятствий) – только коллайдер, иначе плоскость упадет вниз;

3. Прямоугольник – препятствие – гравитация, коллайдер прямоугольника;

4. Квадрат – препятствие – гравитация, коллайдер квадрата;

5. Цилиндр – препятствие – гравитация, коллайдер Цилиндра.

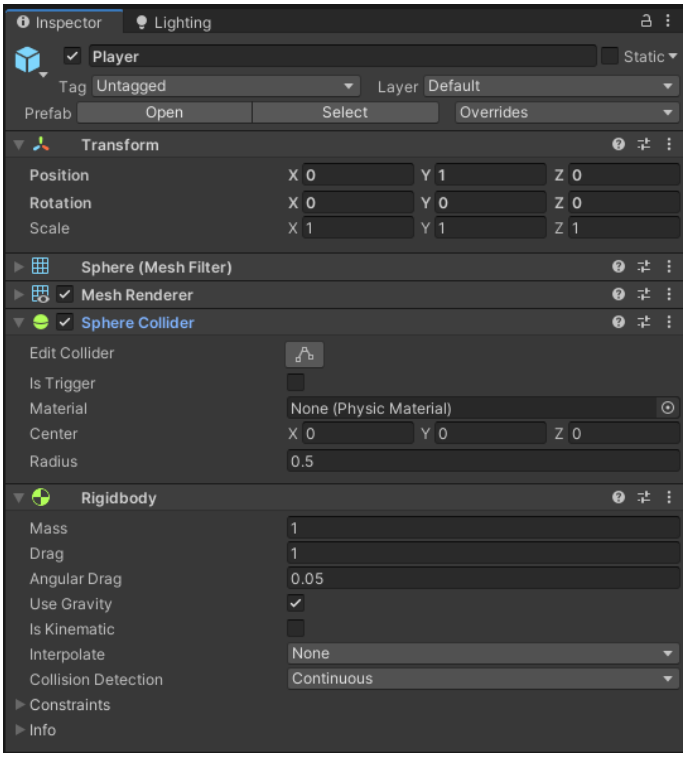


Рисунок 6 – Коллайдер сферы (за счет радиуса), и компонент RigidBody, отвечающий за простую физику – гравитацию

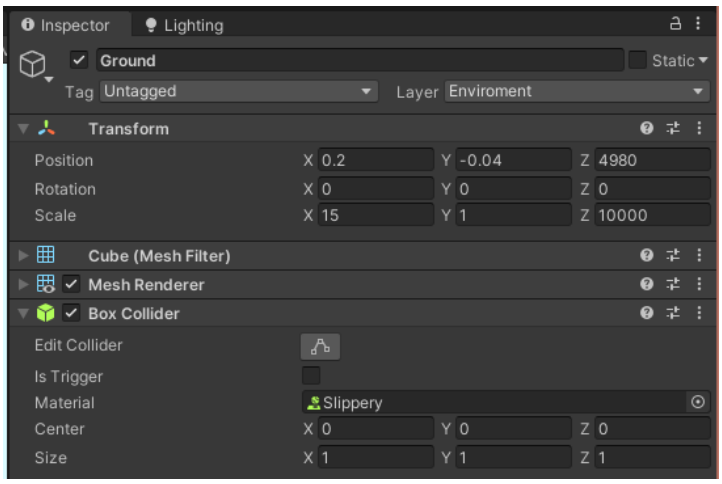


Рисунок 7 – Коллайдер квадрата для плоскости, на которой будут расположены все фигуры. В качестве материала взята скользящая плоскость – делает движения шара «как по льду», и более плавными

Рисунок 8 – Динамическое и статическое трение 2

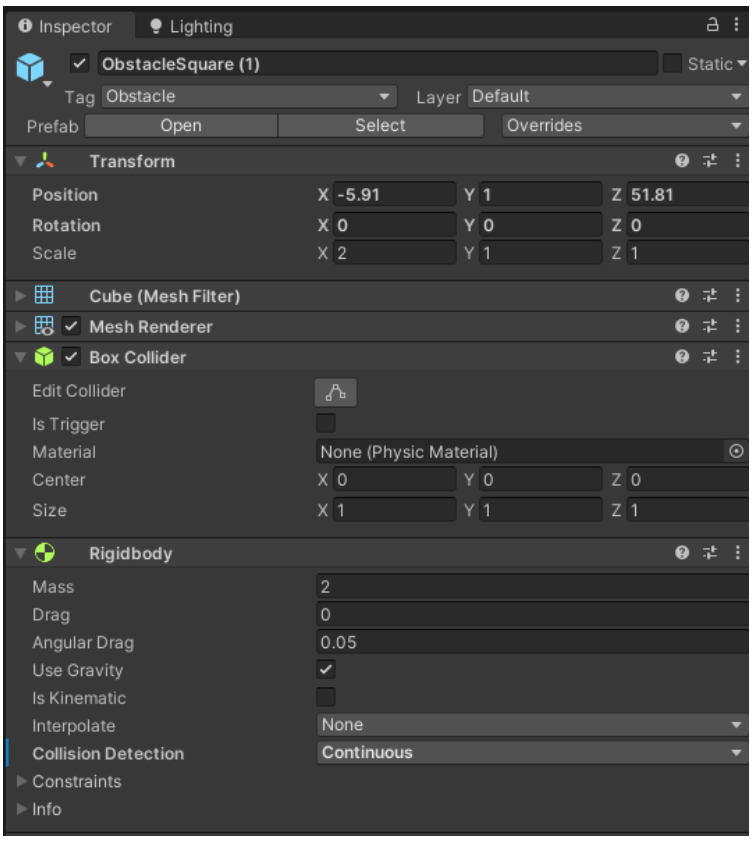


Рисунок 9 – Коллайдер квадрата и Rigidbody для прямоугольника – препятствия, а показатель массы в два раза больше чем у Шара

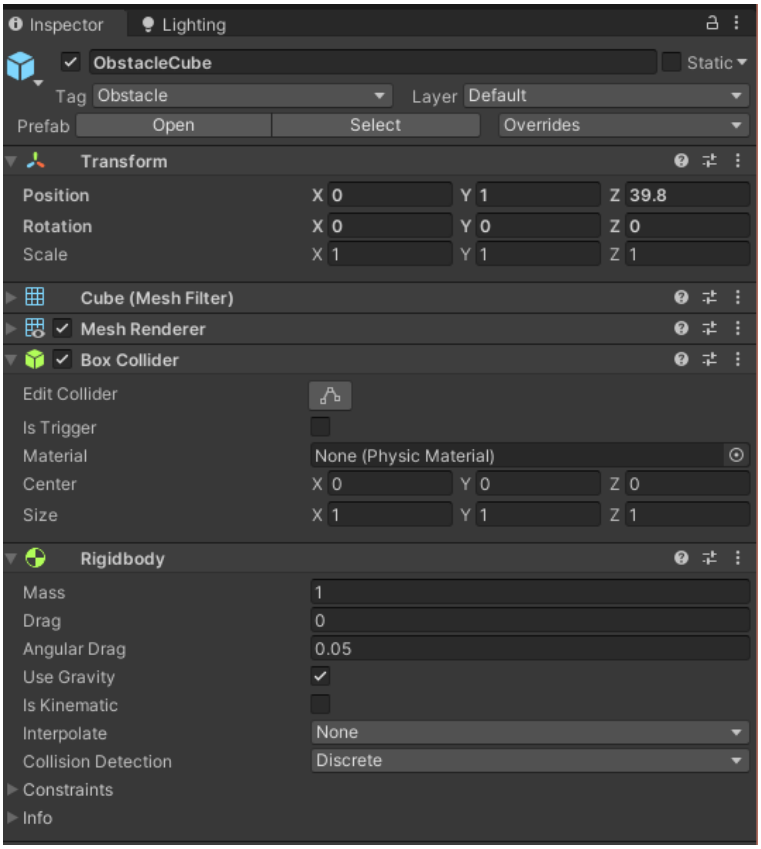


Рисунок 10 – Коллайдер куба и Rigidbody для куба – препятствия, а показатель массы равен показателю массы шара – они примерно одинаковы по размерам

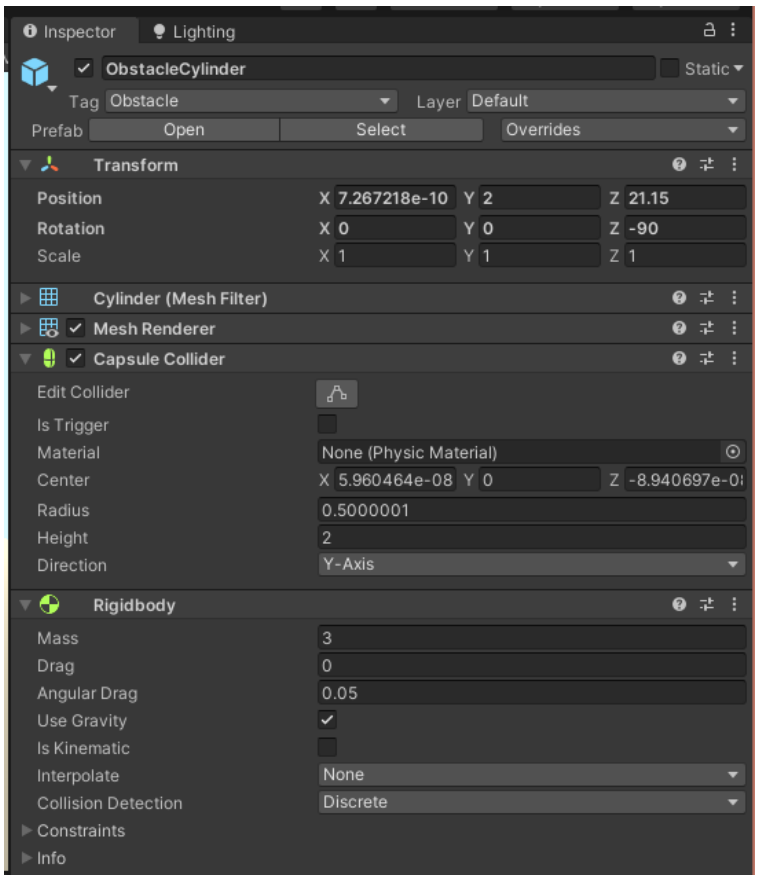


Рисунок 11 – Коллайдер капсулы и Rigidbody для цилиндра – препятствия, а показатель массы в три раза превышает массу Шара

Все установки для каждого объекта на сцене поставлены, и теперь объекты могут взаимодействовать друг на друга – шар катится, если он врезается в препятствие, то оно отлетает, а если препятствие тяжелее шара, то сам шар отскакивает (происходит частично-пластичное столкновение с предметом, часть энергии передается объекту, другая направляет шар в противоположном направлении). Создана плоскость бесконечной длины, на которую были размещены все препятствия. Все фигуры четко взаимодействуют друг с другом.

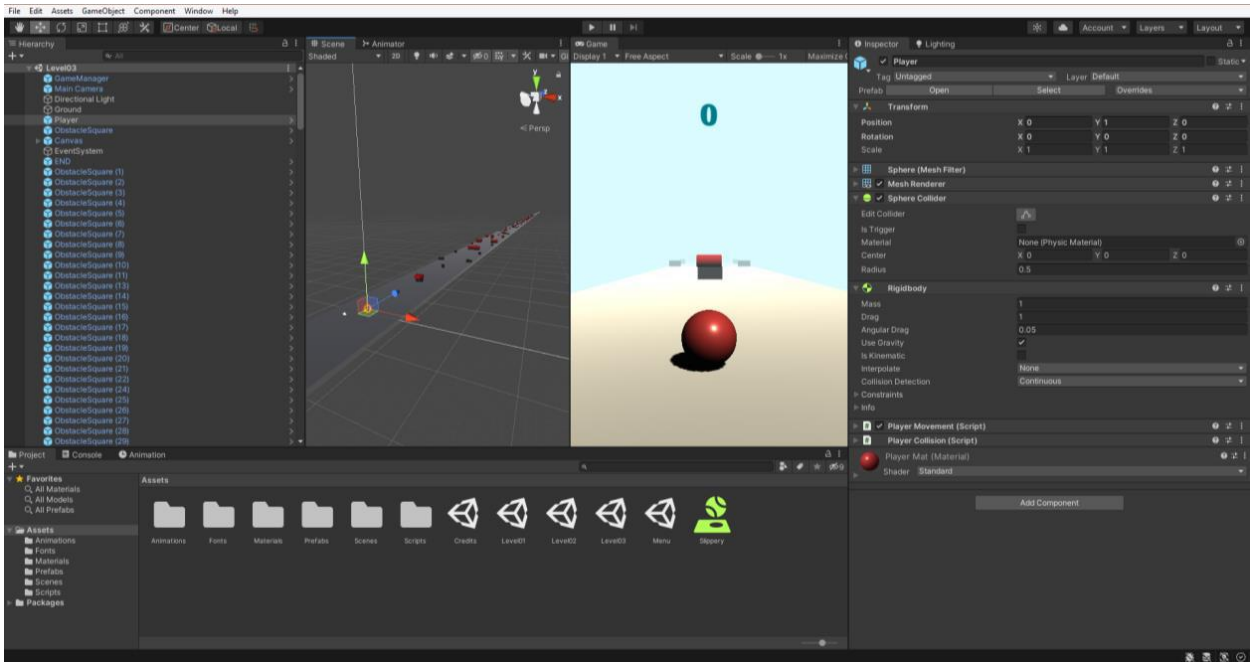


Рисунок 12 – Все фигуры размещены на плоскости. Слева – редактор сценария, справа – вид камеры.

В Юнити уже существует базовый набор «физики» - набор коллайдеров, возможность добавлять силы, действующие на предметы (ускорение, скорость), и еще очень много других возможностей. После тестирования, была выявлена 90% точность (1 из 10 проверок содержала баг) коллизий между Шаром и другими объектами. Шар проскакивал сквозь объект, однако коллизия все же случалась.

Следующий этап разработки – реализация анимации. Анимация – сложная задача, особенно для людей, впервые начинающих разработку компьютерной игры. Однако Юнити предоставляет гибкий функционал для создания анимаций, посредством скриптов (движение тел), либо посредством «аниматора», для заставок, игровых меню и переходов.

Первым делом необходимо заставить шар двигаться вперед с постоянной скоростью. Для этого был создан скрипт, который придает шару постоянную силу вперед (значение по умолчанию – 2000). В результате шар стал постоянно двигаться вперед.

Код – реализация (c#).

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour{

    // A reference to the Rigid Body component
    public Rigidbody rbRef;

    // Variables
    public float forwardForce = 2000f;

    // Start is called before the first frame update
    //void Start(){
        //Debug.Log("Start func, Hi!");
        //rbRef.useGravity = false;
        //rbRef.AddForce(0,20,200);
    //}

    // Update is called once per frame
    // FixedUpdate is neede for physics!
    void FixedUpdate(){
        // Time.deltaTime is a value that helps fixate the amm of F depended on the
framerate
        // Add a forward force
        rbRef.AddForce(0, 0, forwardForce * Time.deltaTime);

    }

}
```

Используется ссылка на RigidBody Шара (переименован на Игрока), чтобы передать ему фиксированную величину силы в положительном направлении оси Зед.

Следующая задача – сделать так, чтобы шаром можно было управлять – двигать его влево или вправо и преодолевать препятствия.

Код – реализация (c#).

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour{

    // A reference to the Rigid Body component
    public Rigidbody rbRef;

    // Variables
    public float forwardForce = 2000f;
    public float sidewaysForce = 500f;

    // Start is called before the first frame update
    //void Start(){
        //Debug.Log("Start func, Hi!");
        //rbRef.useGravity = false;
        //rbRef.AddForce(0,20,200);
    //}

    // Update is called once per frame
    // FixedUpdate is neede for physics!
    void FixedUpdate(){
```

```
        // Time.deltaTime is a value that helps fixate the amm of F depended on the
framerate
        // Add a forward force
        rbRef.AddForce(0, 0, forwardForce * Time.deltaTime);

        // User input conditionals
        if (Input.GetKey("d")) { // D key pressed and held
            rbRef.AddForce(sidewaysForce * Time.deltaTime, 0, 0,
ForceMode.VelocityChange);
            // ForceMode.VelocityChange switches the vel immediately, with no regard
towards mass
            // better controls
        }
        if (Input.GetKey("a")){ // A key pressed and held
            rbRef.AddForce(-sidewaysForce * Time.deltaTime, 0, 0,
ForceMode.VelocityChange);
        }
    }
}
```

Каждый кадр происходит проверка на нажатие клавиш A или D, при нажатии шару придается небольшой толчок в определенном направлении (в положительном или отрицательном направлении оси Икс). ForceMode.VelocityChange – для моментальной смены направления (управление удобнее).

Рисунок 13 – Шаром можно управлять, и он летит вперед

Затем, необходимо добавить главное меню, заставки – переходы на след. уровень, и заставку – конец (credits).

Для анимации заставок можно использовать инструмент «Аниматор».

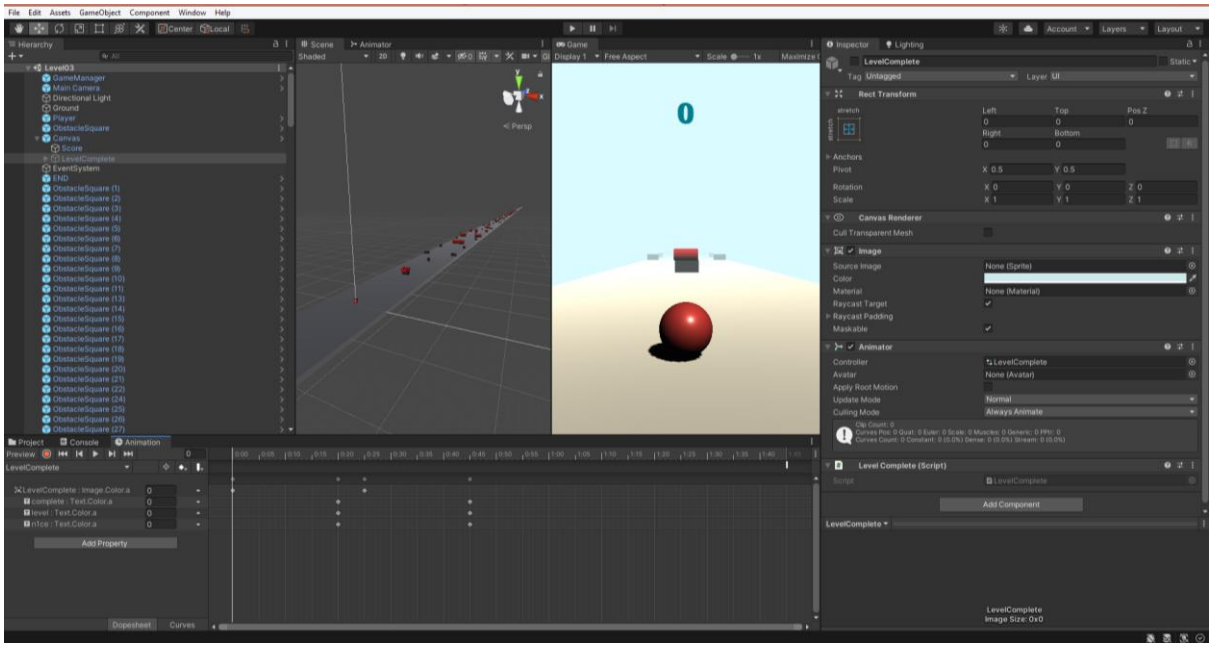


Рисунок 14 – Нижняя панель – аниматор. Лента, разбитая на промежутки по 1/12 секунды (5 долей секунды), на которой можно ставить флажки – этапы начала и конца определенных событий, скриптов итп. Редактор анимаций в Юнити напоминает таковой в Sony Vegas Pro, с исключением того что в Юнити можно использовать скрипты, и другие сложные компоненты

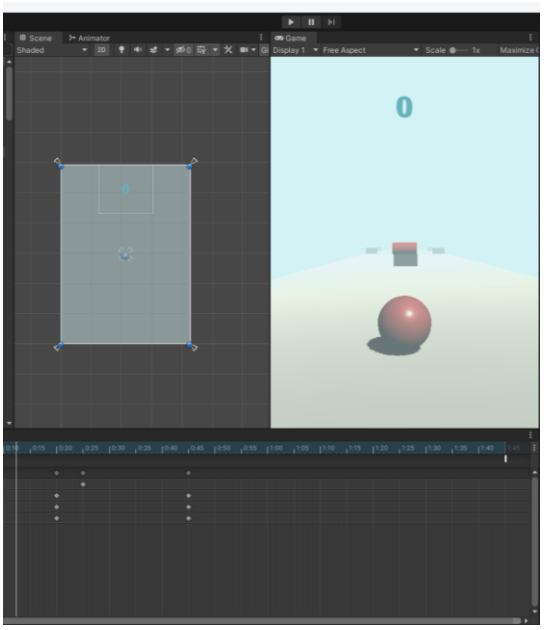


Рисунок 15 – экран плавно заполняет заставка

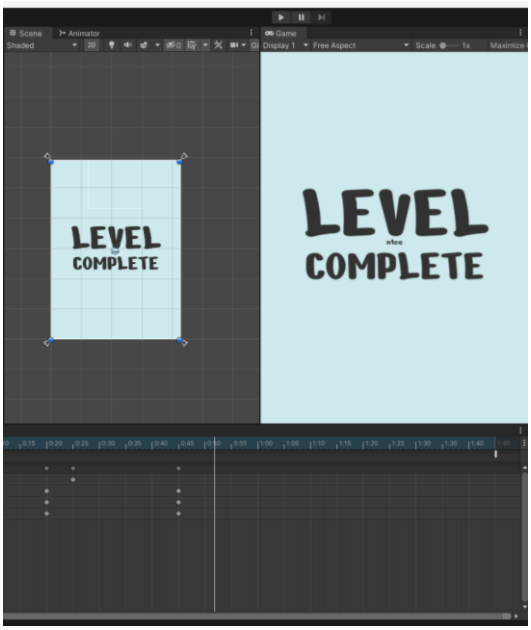


Рисунок 16 – заставка «Уровень Пройден», затем происходит переход на след. уровень.

Добавление главного меню и титров.

Рисунок 17 – окно приветствия

Рисунок 18 – окно завершения

Следующий, последний этап – завершение (исправление ошибок, установление связи между всеми разработанными элементами). Уже были реализованы фигуры, физика и взаимодействия между объектами, анимации и переходы между уровнями, управление, меню и заставки. Однако, игра еще не готова. Меню и Титры не связаны, уровни не переходят сами на следующие, игрок может пролетать сквозь препятствия, или вовсе может слететь с края платформы и бесконечно лететь вниз… это все необходимо исправить.

Первым делом необходимо решить проблему прохождения сквозь предметы. В Юнити существует несколько режимов проверки коллизий – дискретный, постоянный, постоянный динамический, и постоянный теоретический. Для тел, сталкивающихся на большой скорости лучше всего использовать постоянный метод проверки коллизий (Continuous collision detection) (CCD). Когда тело двигается с большой скоростью, оно может пройти сквозь него. Это можно объяснить тем, что каждый кадр координата тела меняется, он переносится вперед, и вовсе не захватывает промежуток координат другого тела.

Sweep-based CCD использует алгоритм TOI (Time Of Impact) для вычисления потенциальных коллизий объектов, проверяя его траекторию

спереди используя показатель скорости. Данный метод решает проблему прохождения сквозь предметы на большой скорости, за счет большей нагрузки на процессор. У этого метода есть и минус – так как он основан на проверке линейной траектории, он игнорирует угловое движение (когда одна часть зафиксирована, а другая – двигается) Для решения этой проблемы можно использовать Speculative CCD (постоянный теоретический), однако в результате тестов выявлено 100% коллизий для установок скорости по умолчанию.

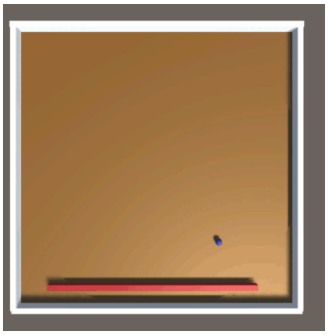


Рисунок 19 – sweep-based CCD. один угол прямой зафиксирован

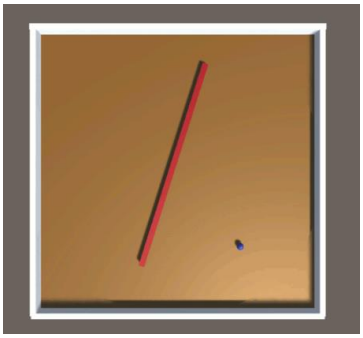


Рисунок 20 – другой конец прямой движется с большой скоростью

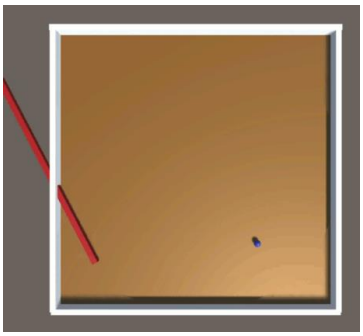


Рисунок 21 – коллизия не произошла, прямая даже влетела в границу песочницы

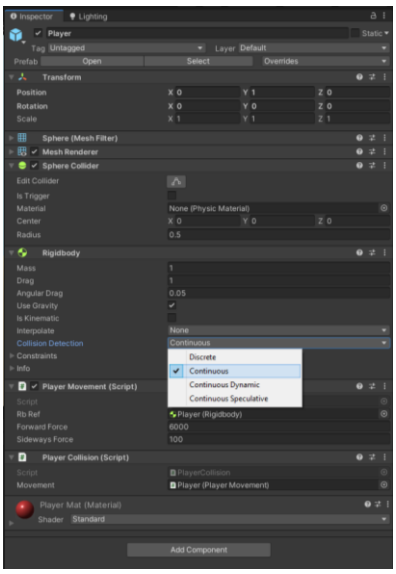


Рисунок 22 – установка CCD для Игрока

Для переходов между уровнями можно написать скрипт, создать невидимую фигуру-триггер, которая при коллизии с Игроком переносит его на след. уровень. Для начала необходимо указать, какие уровни вообще существуют.

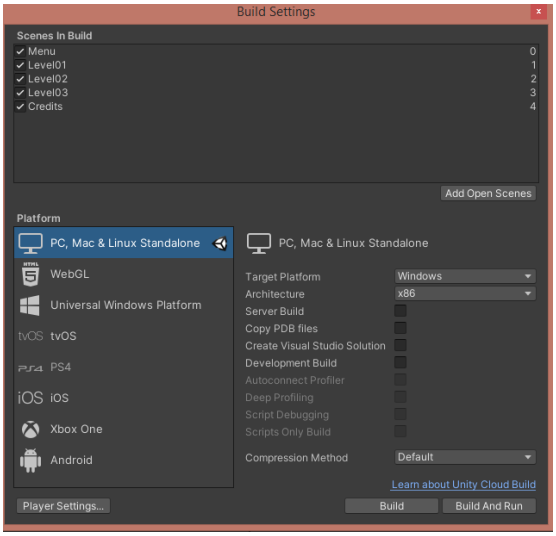


Рисунок 23 – настройки сборки. В иерархическом порядке выставлены главное меню, три уровня, и титры

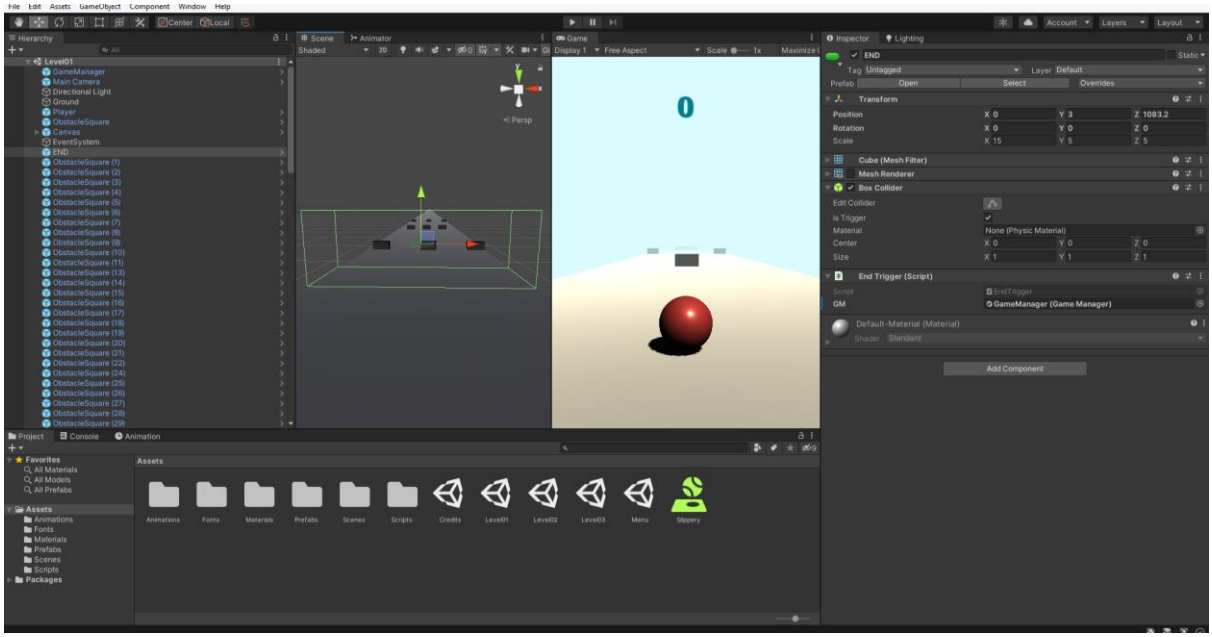


Рисунок 24 – Создана невидимая фигура-триггер, которая при коллизии обращается к методу из скрипта игрового менеджера.

Код – реализация End-Trigger (c#).

```
using UnityEngine;

public class EndTrigger : MonoBehaviour{
    // Vars and Refs
    public GameManager GM;

    // just like oncollenter, we can see if any trigger areas
    // have been collided
    void OnTriggerEnter(){
        GM.CompleteLevel();
    }
}
```

В момент коллизии будет вызван метод «Завершить уровень»

Код – реализация Game Manager (c#).

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour{
    // Variables
    public GameObject completeLevelUI;

    // Functions
    public void CompleteLevel(){
        completeLevelUI.SetActive(true);
    }
}
```

В методе CompleteLevel активируется элемент интерфейса «заставка – переход уровень завершен», и загружается след. сцена.

Код – реализация скрипта Level Complete (c#).

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class LevelComplete : MonoBehaviour{
    public void LoadNextLevel(){
        // scene name in ()
        // instead of using names much better using build indexes
        // 2get 2the next scene - this scene +=1
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
}
```

Просто загружается следующая сцена

Условия победы есть, но игра слишком простая – игрок просто сталкивается с препятствиями и продолжает катится вперед. А если он слетает с платформы, то летит бесконечно вниз. Необходимо добавить условия смерти игрока.

Первым делом, если высота игрока становится ниже -1, то с небольшой задержкой отправить его в начало уровня. За движение и позицию игрока отвечает скрипт передвижения игрока

Код – дополнение в PlayerMovement (c#).

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour{

    // A reference to the Rigid Body component
    public Rigidbody rbRef;

    // Variables
    public float forwardForce = 2000f;
    public float sidewaysForce = 500f;

    // Start is called before the first frame update
    //void Start(){
        //Debug.Log("Start func, Hi!");
        //rbRef.useGravity = false;
        //rbRef.AddForce(0,20,200);
    //}

    // Update is called once per frame
    // FixedUpdate is neede for physics!
    void FixedUpdate(){
        // Time.deltaTime is a value that helps fixate the amm of F depended on the
framerate
        // Add a forward force
        rbRef.AddForce(0, 0, forwardForce * Time.deltaTime);
```

```
        // User input conditionals
        if (Input.GetKey("d")) { // D key pressed and held
            rbRef.AddForce(sidewaysForce * Time.deltaTime, 0, 0,
ForceMode.VelocityChange);
            // ForceMode.VelocityChange switches the vel immediately, with no regard
towards mass
            // better controls
        }
        if (Input.GetKey("a")){ // A key pressed and held
            rbRef.AddForce(-sidewaysForce * Time.deltaTime, 0, 0,
ForceMode.VelocityChange);
        }
        // check if Y pos of player gets below a certain number
        if (rbRef.position.y < -1f){
            FindObjectOfType<GameManager>().gameOver();
        }

    }
}
```

Код – дополнение в GameManager (c#).

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour{
    // Variables
    bool gameHasEnded = false;
    public float restartDelay = 1f;
    public GameObject completeLevelUI;

    // Functions
    public void gameOver(){
        if (gameHasEnded == false) {
            gameHasEnded = true;
            //Debug.Log("GAME OVER!");
            // invoke calls a func with a delay
            Invoke("restartGame",restartDelay);
        }
    }

    public void restartGame() {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

    public void CompleteLevel(){
        completeLevelUI.SetActive(true);
    }
}
```

После того как игрок слетает вниз, через секунду он появляется в начале уровня. Теперь необходимо добавить условия смерти для столкновений с препятствиями.

Код – реализация скрипта PlayerCollision (c#).

```
using UnityEngine;
```

```
public class PlayerCollision : MonoBehaviour{
    // Variables
    // A way to disable movement on collision - disabling the movement script
    public PlayerMovement movement;

    // whenever a collision occurs, unity has a function:
    void OnCollisionEnter(Collision collisionInfo){
        //Debug.Log(collisionInfo.collider.name);
        // checking for names is tedious and heavy..
        if (collisionInfo.collider.tag == "Obstacle") {
            //Debug.Log("We hit somethin' with a tag Obstacle!");
            movement.enabled = false;
            FindObjectOfType<GameManager>().gameOver();
        }
    }
}
```

Теперь, когда игрок сталкивается с объектом отключается скрипт управления и ускорения, и вызывается метод скрипта GameManager который переносит игрока в начало уровня.

ЗАКЛЮЧЕНИЕ

Игра завершена и готова, были исправлены ошибки. За относительно короткие сроки был написан небольшой проект, который хоть и не имеет большого коммерческого потенциала или оригинальности, обладает потенциалом в дальнейшем развитии: Добавление новых уровней или фигур занимает минимальное время (можно даже использовать метод процедурной генерации уровней), а также можно добавить другие играбельные фигуры, текстуры, музыку, анимации итп. В целом этот проект дал большой опыт в использовании игровых движков, создании игр, а также расширил понятия интерактивных графически систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Webscketches. 2D ИГРА НА UNITY. ПОДРОБНОЕ РУКОВОДСТВО [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: http://websketches.ru/blog/2d-igra-na-unity-podrobnoye-rukovodstvo-p1 свободный
2. Unity. Coroutines [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ru/2019.4/Manual/Coroutines.html свободный
3. Unity. Collider2D [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ru/2019.4/Manual/Collider2D.html свободный
4. Unity. Rigidbody2D [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ru/2019.4/Manual/class-Rigidbody2D.html свободный
5. Unity. Resources.Load [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ScriptReference/Resources.Load.html свободный
6. Unity. Renderer [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ScriptReference/Renderer.html свободный
7. Unity. TextAsset [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ScriptReference/TextAsset.html свободный
8. Unity. Input [Электронный ресурс] // Электрон. текстовый документ – Режим доступа: https://docs.unity3d.com/ScriptReference/Input.html свободный