МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

к.т.н., доцент		Н.Н. Решетникова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Изучение физических свойств объектов и их взаимодействия в Unity

по дисциплине: Специальные разделы мультимедиа

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № Z0440M 18.01.2022 Митрофанов Д.А.

Санкт-Петербург 2022

1 Цель работы

Знакомство с физическими свойствами объектов на примере сцены, созданной в ЛР №3. Реализация физики твёрдого тела для взаимодействия объектов друг с другом на игровом движке Unity.

2 Выполнение работы

2.1 Настройка коллайдеров

Для демонстрации работы физического движка на сцену были установлены бочки с разной массой:

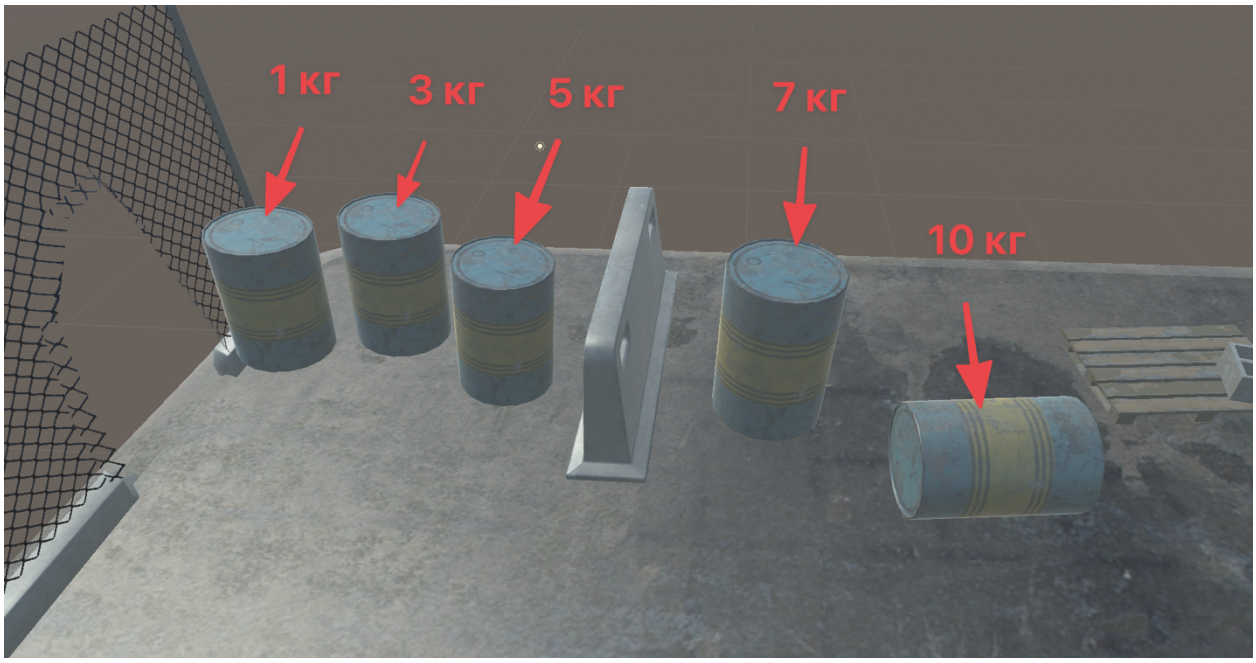

Рисунок 1 – Внешний вид бочек на сцене и их характеристики

На каждый из них был назначен модификатор *RigidBody* (данный модификатор указывает на то, что объект является твердым телом и подчиняется законам физического движка) и по два коллайдера: *Box Collider* и *Capsule Collider.* Такая комбинация идеально подходит для цилиндрических объектов, коими являются бочки. Настройки модификаторов приведены на Рисунке 2.

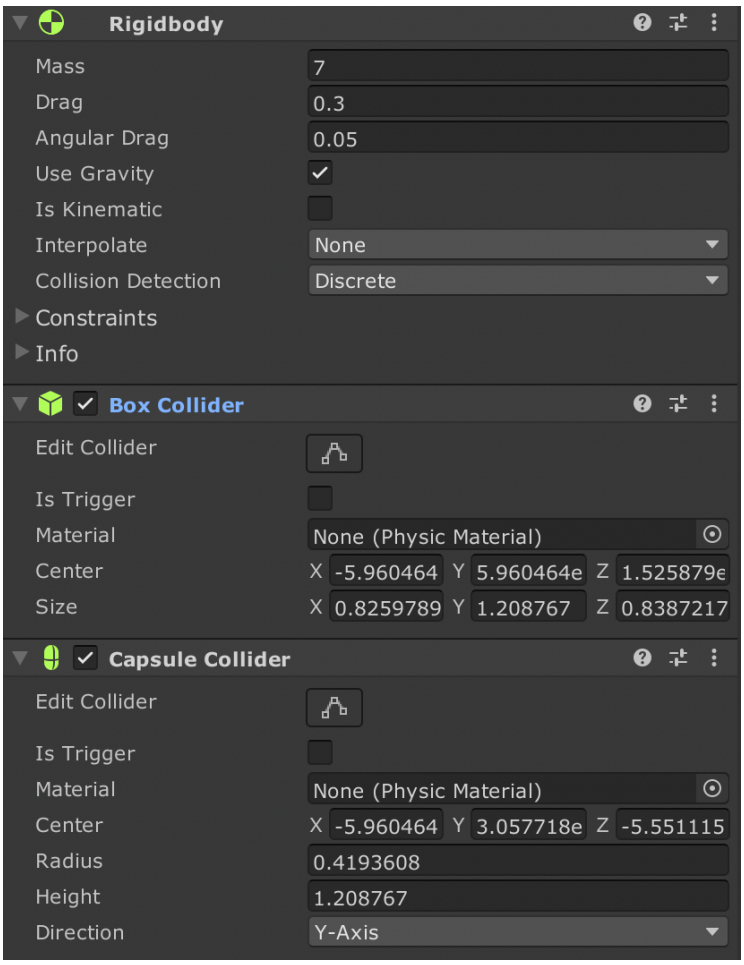

Рисунок 2 – Настройки модификаторов бочек

Также на сцену был введен футбольный мяч. К нему был прикреплен сферический коллайдер с физическим материалом и модификатор *RigidBody (Рисунок 3)*.

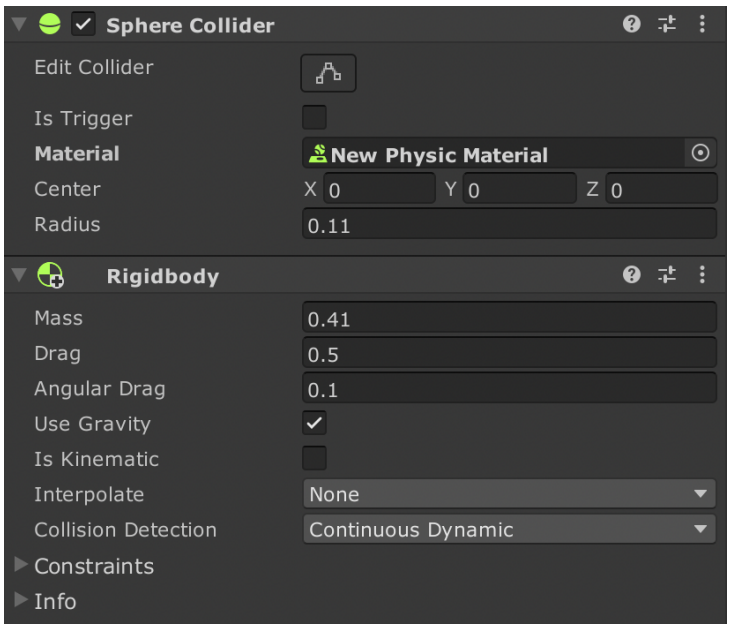

Рисунок 3 – Настройки модификаторов мяча

Физический материал определяет то, как объект будет взаимодействовать с другими коллайдерами. В данном случае настраиваются параметры трения и упругости мяча (Рисунок 4).

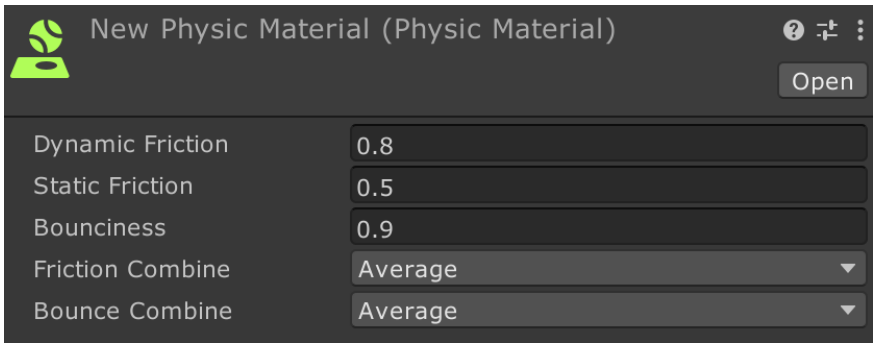

Рисунок 4 – Настройки материала мяча

2.2 Настройка игрока

Для того, чтобы игрок смог бросать мяч, нужно создать соответствующий скрипт (Листинг 1) и привязать его к персонажу

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ThrowBall : MonoBehaviour {
    public GameObject ball;
    public Transform spawnBalls;
    public float throwForce;
    private float rate = 0.9f;
    private float nextThrow = 0.0f;
    void FixedUpdate()
    {
if (Input.GetButton("Fire1") && Time.time > nextThrow)
{
nextThrow = Time.time + rate;
GameObject newBall = Instantiate(ball);
newBall.transform.position = spawnBalls.transform.position +
spawnBalls.transform.forward;
newBall.GetComponent<Rigidbody>().AddForce(spawnBalls.forward * throwForce,
ForceMode.Impulse);
} }
}
```

Листинг 1 – Код используемого контроллера

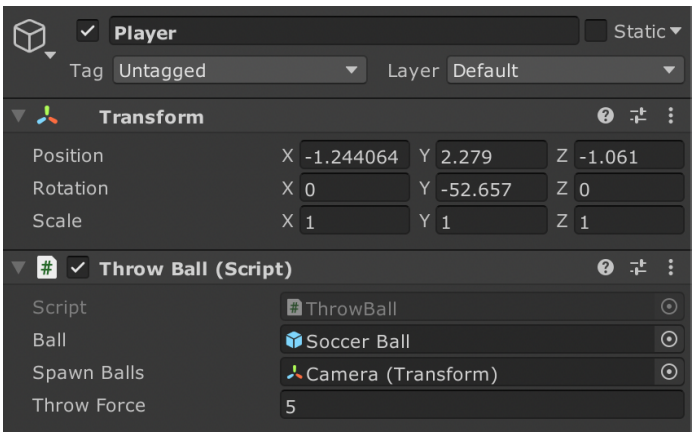

Рисунок 5 - Привязка скрипта к персонажу

2.3 Сборка сцены

Работа выполнялась на платформе macOS с процессором m1, режим сборки соответствующий:

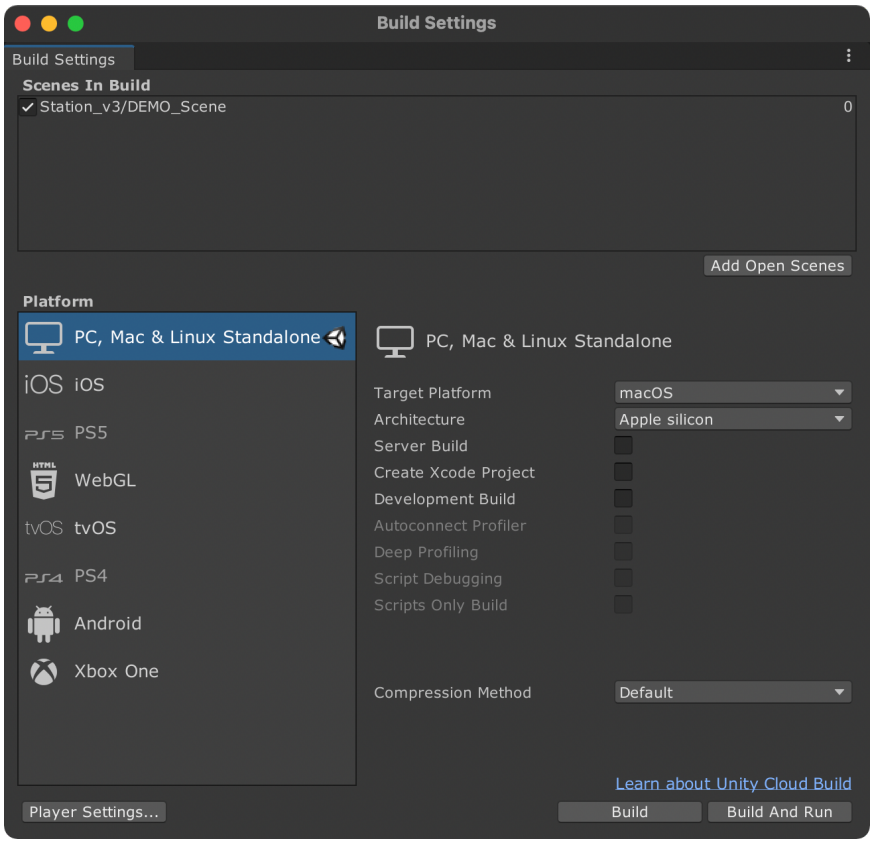

Рисунок 6 – Настройка сборки

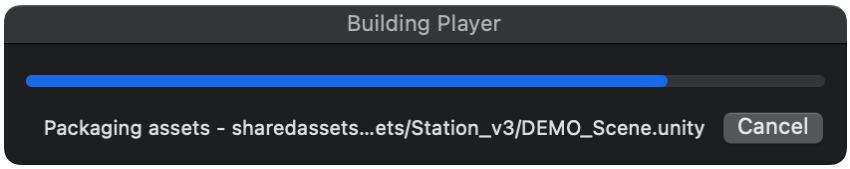

Рисунок 7 - Процесс сборки

После сборки в указанной папке появляется исполняемый файл (в случае с macOS, с расширением .app):

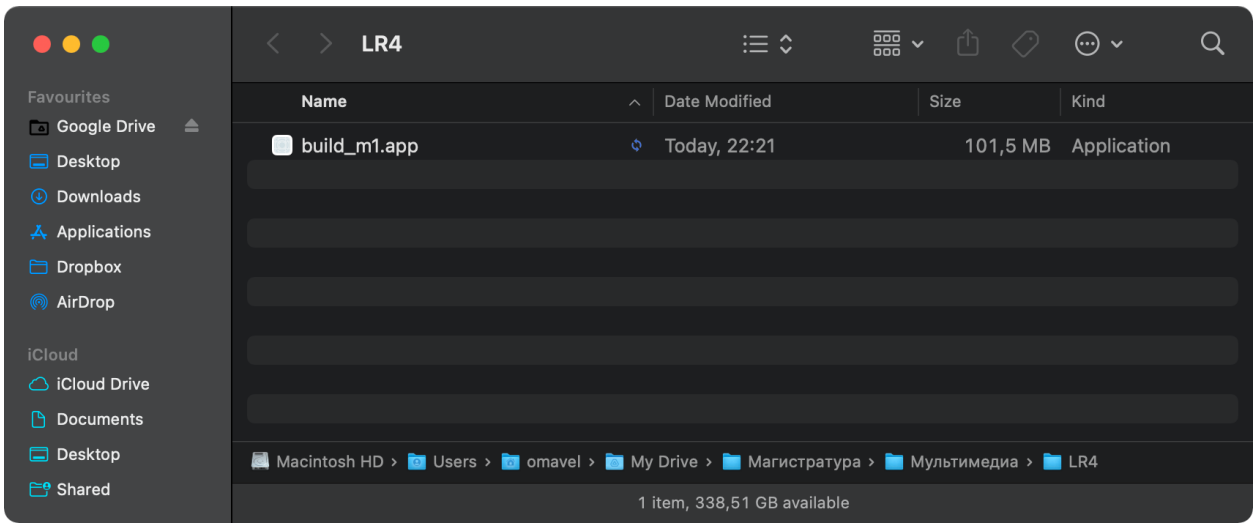

Рисунок 8 – Исполняемый файл

Выводы

Были изучены настройки физического движка Unity, реализована физика твёрдого тела для взаимодействия объектов друг с другом, изучены методы сборки проекта.

Видеодемонстрация доступна по ссылке: https://youtu.be/_afjW5djz-w