Лабораторная работа №1

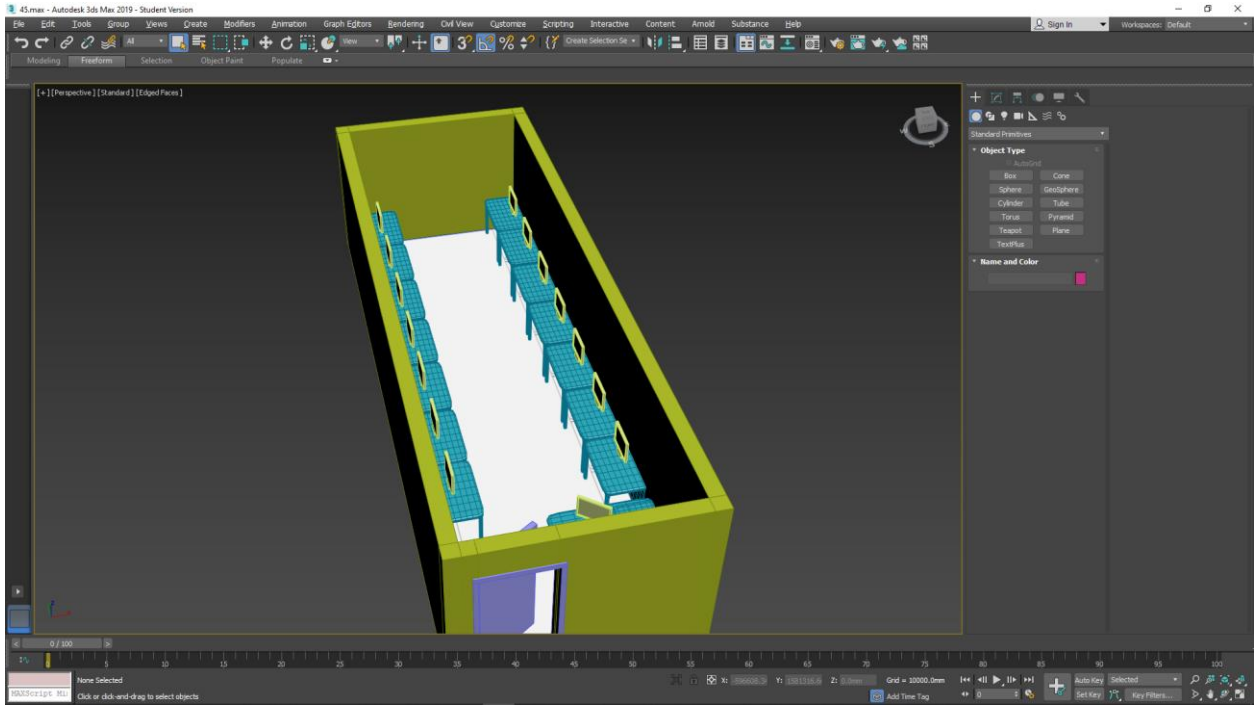
Цель работы: Создание виртуальной сцены с использованием компонентов физического взаимодействия.

Задачи для достижения поставленной цели:

1. Разработать модель для экспорта комнаты в 3ds Max
2. Импортировать разработанные модели в виртуальную среду Unity
3. Создать сцену в виртуальной среде Unity используя импортированные модели
4. Подключить компоненты физического взаимодействия (Mesh и Rigidbody)
5. Произвести назначение материалов на GameObject
6. Подключить FPS Controller
7. Собрать и запустить разработанный проект

Ход работы:

1. Создание необходимой модели комнаты, происходит опираясь на полученные знания прошлого семестра.



2. Экспортирование созданной модели (Формат FBX, OBJ)

В 3DS Max, это происходит в меню File>Export.

3. Импортирование моделей в Unity

Импортировать модели можно просто перенеся файл с экспортом, либо в папку Assets, а оттуда в окно сцены или иерархии, либо сразу в окно сцены/иерархии

4. Подключение компонентов взаимодействия

В текущей лабораторной работе потребуется подключение и настройка Rigidbody и различных коллайдеров.

Все это происходит на панели inspector, при нажатии по кнопке Add Component.

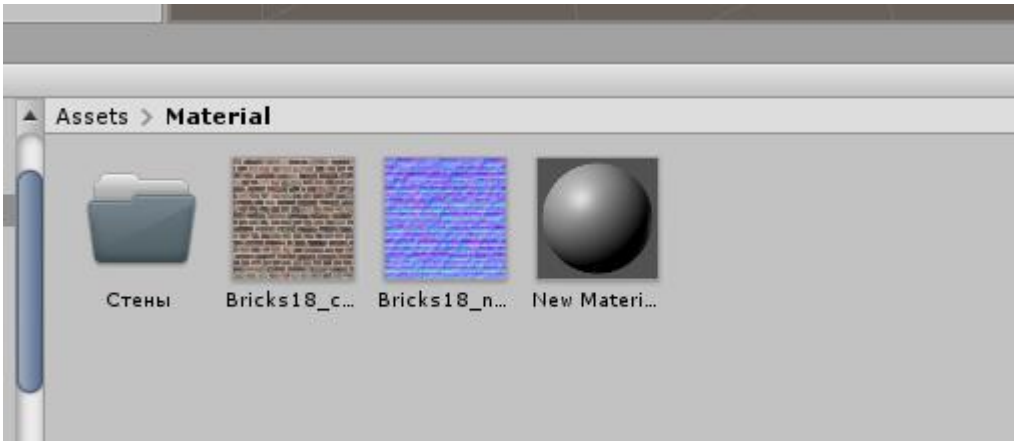
Коллайдеры (Colliders):

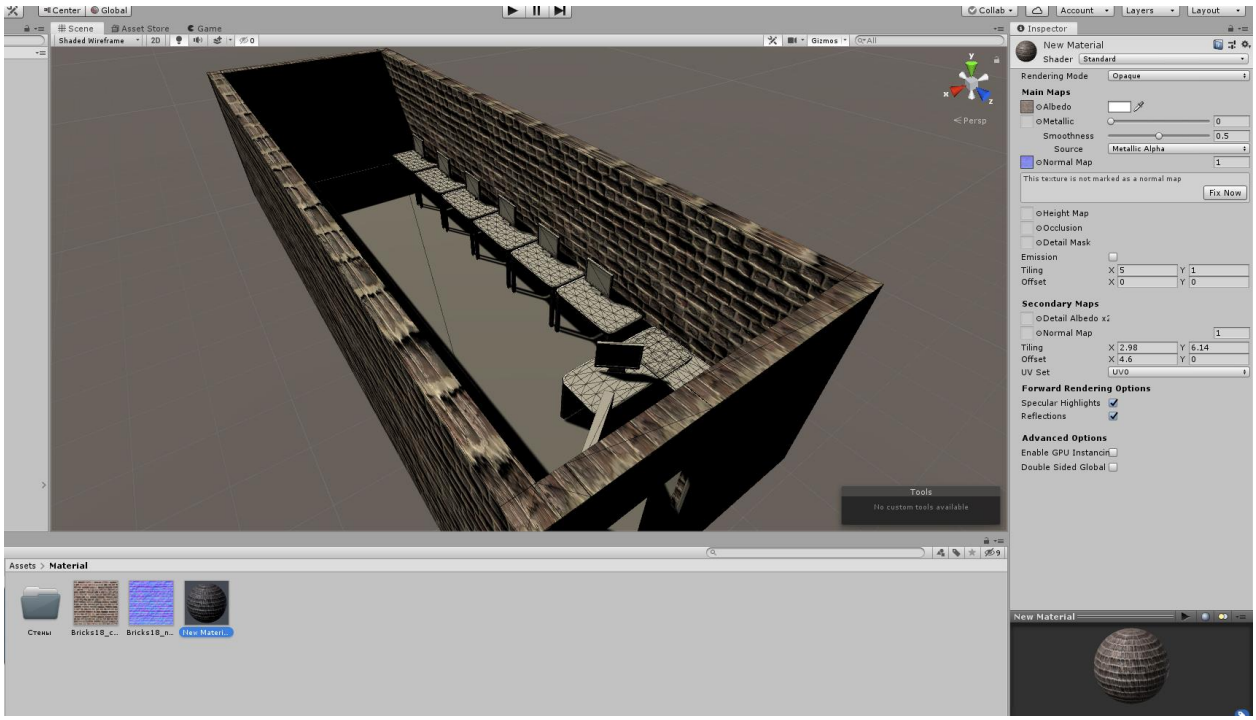
Компоненты коллайдера определяют форму объекта для целей физических столкновений.

В данной лабораторной работе они необходимы для создания твердых объектов.

Rigidbody - это основной компонент, подключающий физическое поведение для объекта. С прикреплённым Rigidbody, объект немедленно начнёт реагировать на гравитацию. Если добавлен один или несколько компонентов Collider, то при коллизиях (столкновениях) объект будет передвигаться.

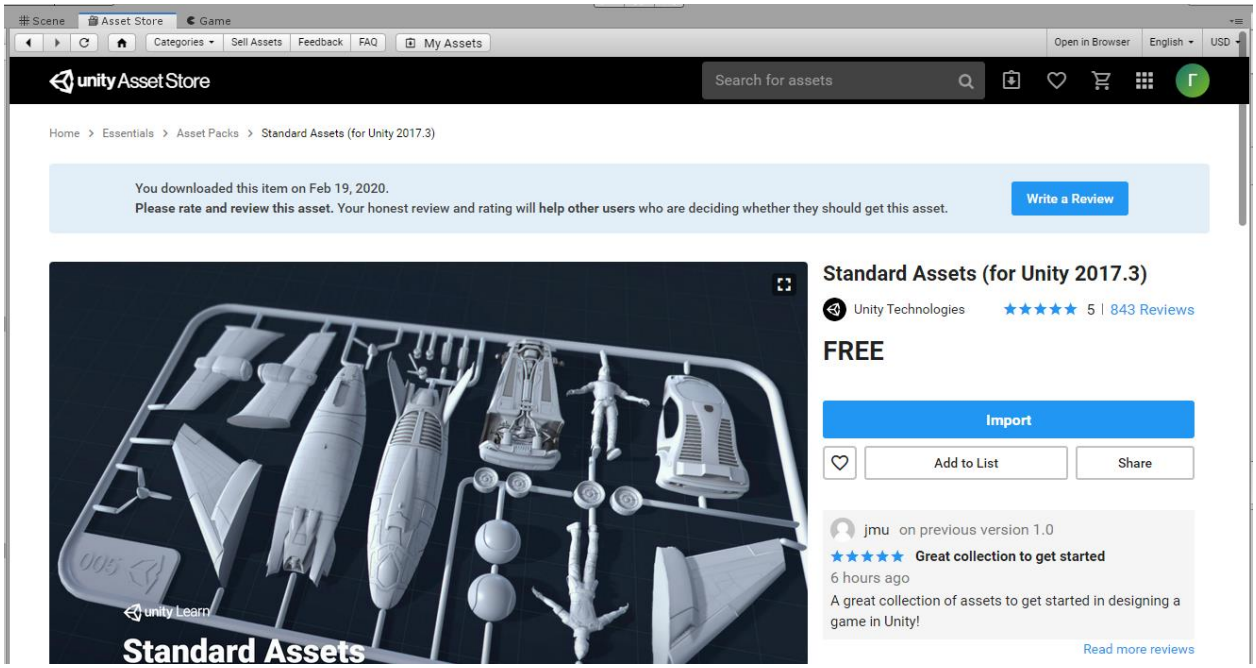
5. Для создания фотореалистичных текстур, создадим в Unity новый материал и подключим к нему две карты, основную карту цвета в Albedo и карту нормалей в Normal map





6. Подключение FPS controller

Подключить FPS controller, можно скачав и импортировав «Standart Assets» из вкладки AssetStore.



Сам FPS controller будет находиться по следующему пути в рабочей папке проекта: Assets>Standart Assets>Characters>FirstPersonCharacter>Prefabs

Лабораторная работа №2

Цель работы: Создание виртуальной сцены с использованием триггеров для интерактивного взаимодействия с виртуальными объектами.

Задачи для достижения поставленной цели:

1. Разработать модель комнаты в 3ds Max
2. Импортировать разработанные модели в виртуальную среду Unity
3. Создать сцену в виртуальной среде Unity используя импортированные модели
4. Подключить компоненты физического взаимодействия (Mesh и Rigidbody)
5. Написать скрипты физического взаимодействия с объектами
6. Произвести назначение материалов на GameObject
7. Подключить FPS Controller
8. Собрать и запустить разработанный проект

Ход работы:

Для создания скриптов физического взаимодействия первым делом потребуется создать пустой скрипт, нажав в меню Assets правой кнопкой мыши Create> C# Script

Создастся пустой скрипт

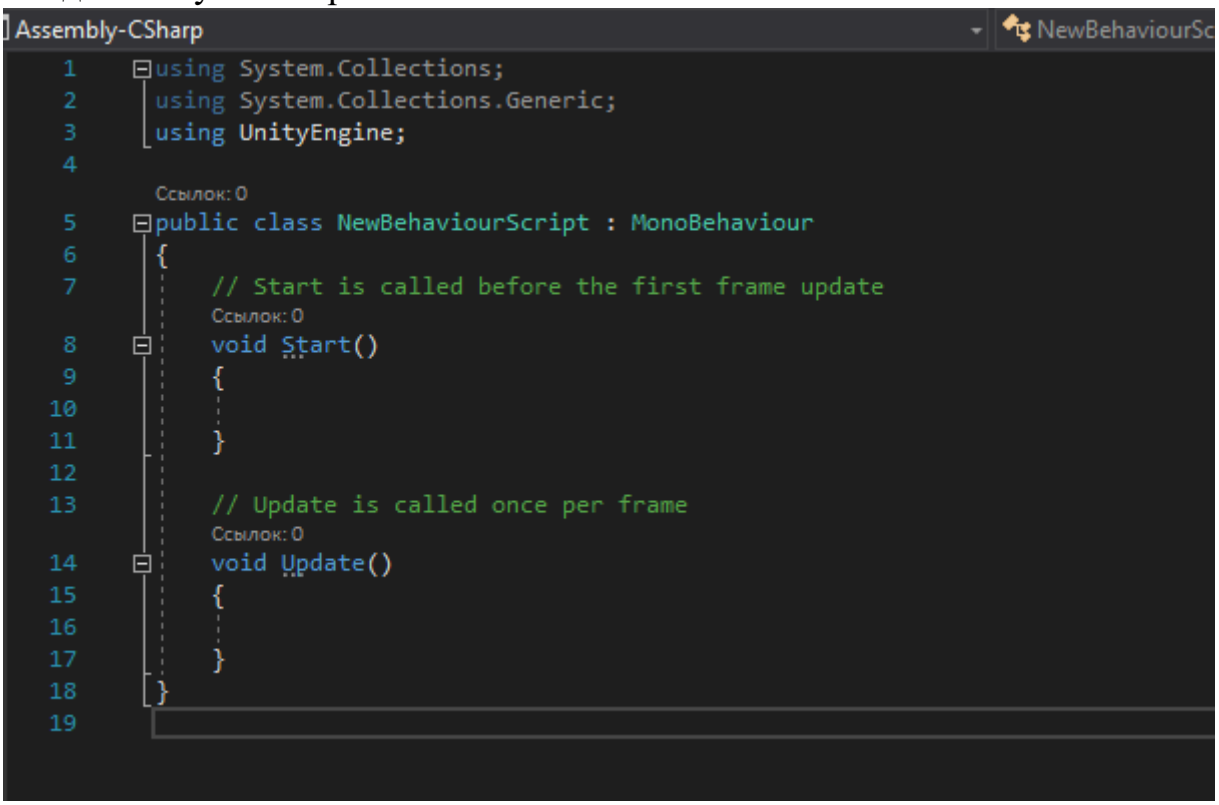


Рис.1 пустой скрипт

Далее требуется создать различные скрипты и назначить их на объекты. При необходимости на компоненте коллайдер требуется включать режим IsTrigger, если данный коллайдер будет являться триггером, и настраивать размеры коллайдера в меню Edit Collider

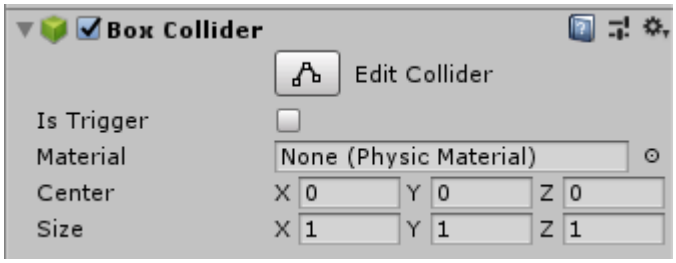


Рис.2 Настройки коллайдера

Элементарный скрипт на открытие двери:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Ссылок: 0
public class NewBehaviourScript : MonoBehaviour
{
    public GameObject door; // Игровой объект
    Ссылок: 0
    private void OnTriggerStay(Collider other) // метод выполняемый при нахождении объекта в коллайдере
    {
        if (Input.GetKeyDown(KeyCode.E))// Ввод с клавиатуры
            door.transform.Rotate(0f, 90f, 0f);// Разворот объекта
    }

    Ссылок: 0
    private void OnTriggerExit(Collider other) // метод выполняемый при выходе из коллайдера
    {
        door.transform.Rotate(0f, 270f, 0f);
    }

}
```

Рис.3 Скрипт на открытие двери

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    Ссылок: 0
5   public class NewBehaviourScript : MonoBehaviour
6   {
7       private Light MyLight;
8
        Ссылок: 0
9       private void Start()
10      {
11          {
12              MyLight = GetComponent<Light>();
13          }
14      }
        Ссылок: 0
15      void Update()
16      {
17          if (Input.GetKey(KeyCode.L))
18          {
19              MyLight.enabled = !MyLight.enabled;
20          }
21      }
22  }
23
```

Рис.4 Скрипт включения/выключения глобального освещения.

Пример скрипта, на включение/выключение глобального освещения:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Ссылок: 0
public class NewBehaviourScript : MonoBehaviour
{
    public GameObject Obj;
    Ссылок: 0
    private void Update()
    {

        if (Input.GetKey(KeyCode.R))
        {
            gameObject.GetComponent<Renderer>().enabled = false;
        }


    }
}
```

Рис.5 Скрипт на выключение текстур у объекта

Пример выполнения скрипта на выключение текстур(Рис.6)

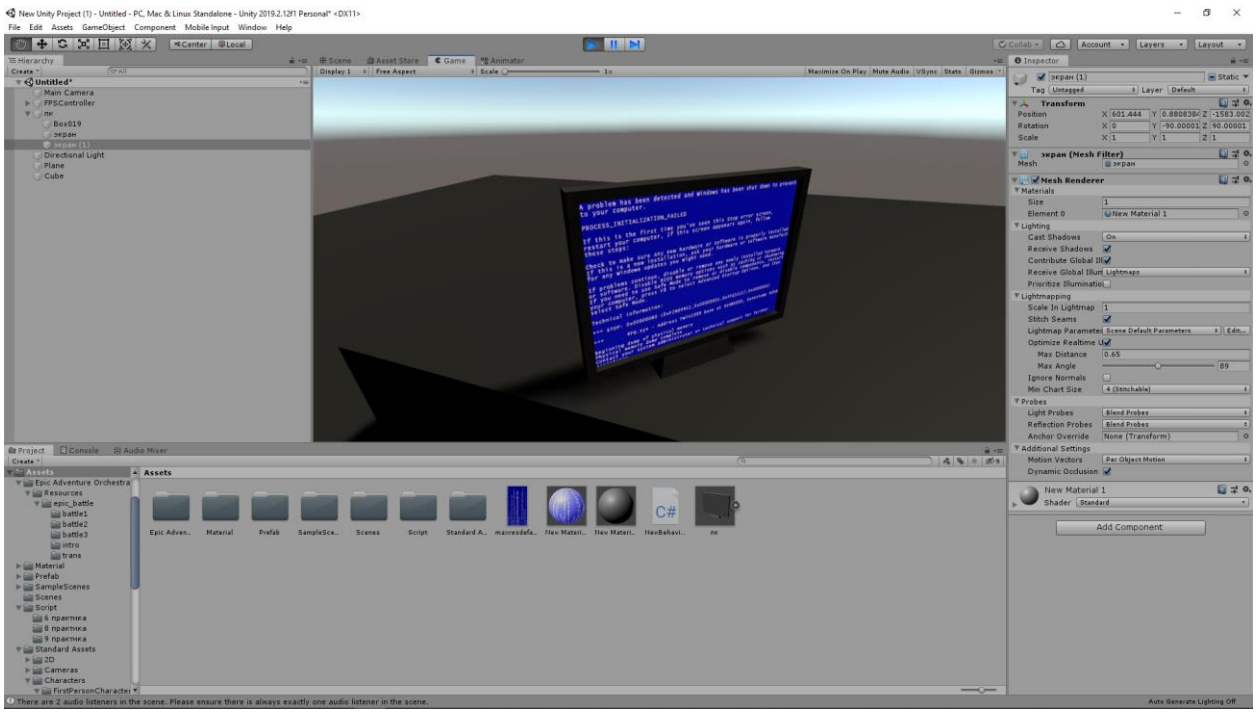


Рис.6 Пример выполнения скрипта на выключение текстур

При нажатии на кнопку R текстура, наложенная на экран, пропадет(Рис.7)

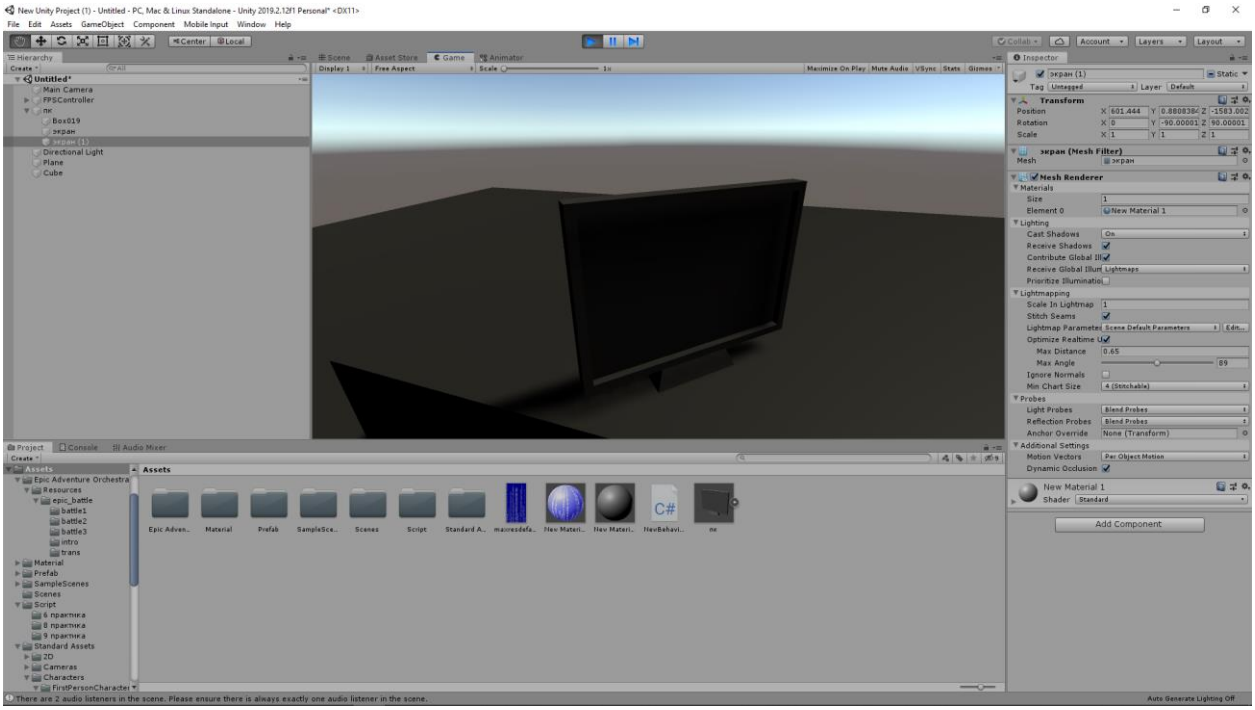


Рис.7 Пример выполнения скрипта на выключение текстур

Лабораторная работа №3

Цель работы: Разработка графического интерфейса для приложения виртуальной реальности

Задачи для достижения поставленной цели:

1. Скачать и импортировать в проект Fibrum sdk
2. Настроить компонент Canvas
3. Создание объекта «информационного щитка»
4. Создание объекта «перекрестия»
5. Создание объекта «Информационное лобовое стекло»
6. Создание объекта «Игровой элемент пользовательского интерфейса»
7. Создание объекта «Выноска»
8. Собрать и запустить разработанный проект

Ход работы:

Скачать FibrumSDK - http://www.fibrum.com/sdk/#sdk_download 3

Для импортирования FibrumSDK в меню нажмите “Assets” -> “Import package” ->”Custom package” и выберете скачанный ранее файл FibrumSDK.unitypackage. В появившемся окне нажмите “Import” (Рис. 1).

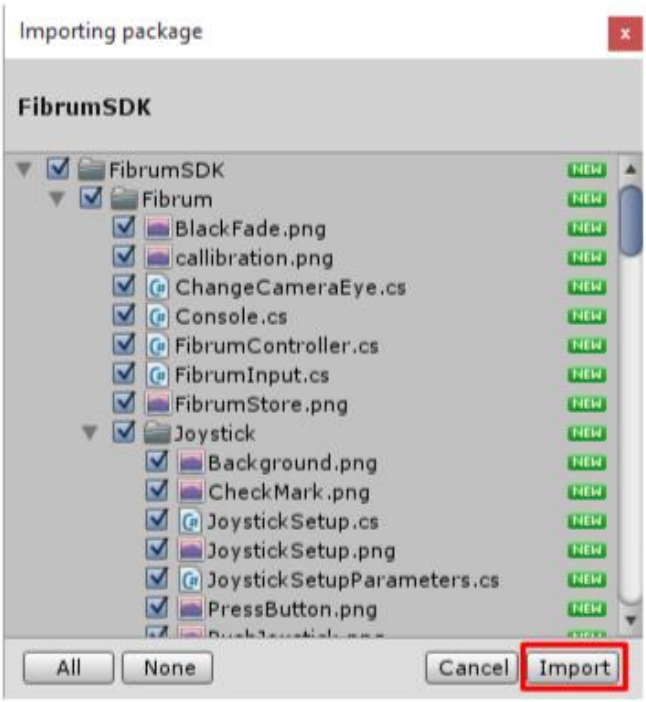


Рис. 1. (Ипорт Fibrum sdk)

Удалите (ПКМ -> Delete) все объекты из окна “Hierarchy” (1). В окне “Progect” откройте папку “Assest -> FibrumSDK -> Perfabs” (2). Перетащите файл VR_Camera (3) в окно “Hierarchy” (1) – Рис. 2.

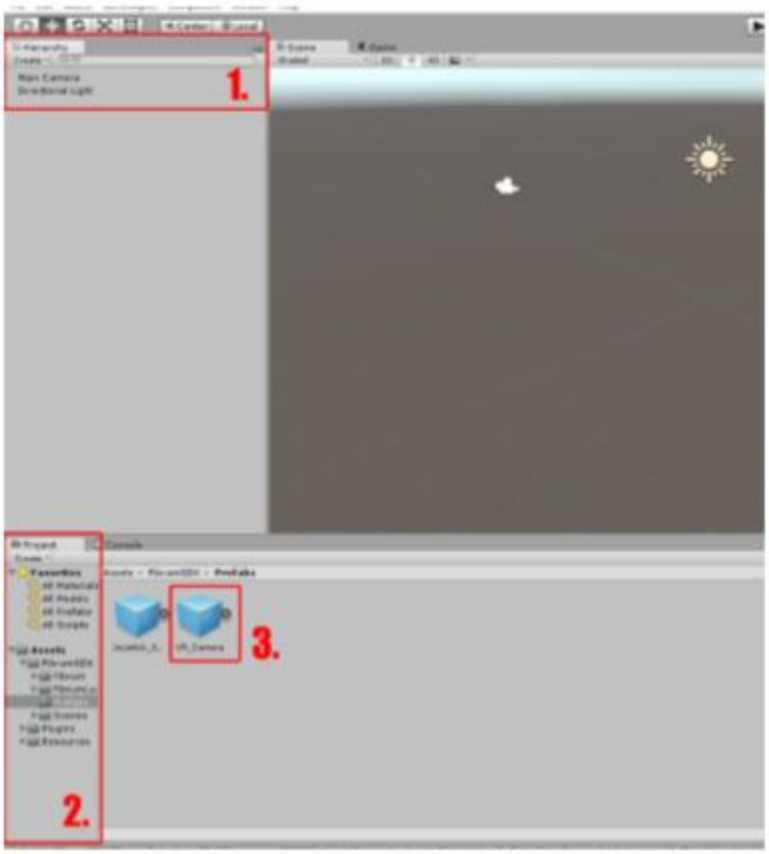


Рис. 2. Первичная настройка сцены Unity с Fibrum sdk

В Unity 5 элементы пользовательского интерфейса всегда помещают в компонент Canvas (Холст). В руководстве Unity он описывается следующим образом:

Компонент Canvas представляет собой абстрактную область, в которой размещен UI. Все элементы должны быть дочерними по отношению к объекту игры, к которому присоединен компонент Canvas.

В лабораторной работе будет описан ряд способов реализации UI в виртуальном пространстве. Подробно будут рассмотрим перечисленные ниже методы и примеры использования каждого из них:

- Информационный щиток. При использовании информационного щитка (HUD-щитка) пользовательский интерфейс располагается на одном и том же месте перед глазами независимо от перемещений головы.
- Курсоры в виде перекрестья. Перекрестье, то есть указатель курсора в форме мишени, во многом схоже с информационным щитком, но используется для выбора какого-либо элемента в сцене.
- Информационное лобовое стекло. Представляет собой всплывающую в 3D-пространстве панель, похожую на лобовое стекло кабины.
- Игровой элемент пользовательского интерфейса. Отображается в сцене как часть проекта,
- Выноска. UI-сообщение, присоединенное к объектам в сцене, похожее на реплику в облачке, парящем над головой персонажа.

Компонент Unity **Canvas** (Холст) предоставляет множество настроек и параметров для обеспечения гибкости размещения разных видов графики

Необходимо создать новый объект Canvas и изменить его свойство **Render Mode** (Режим отображения) на **world space** (глобальное пространство):

1. Выберите **GameObject ~ UI ~ Canvas** (Игровой объект ~ Пользовательский интерфейс ~ Холст).
2. Переименуйте вновь созданный объект Canvas в DefaultCanvas. (Рис.3)

3. Установите его свойство **Render Mode** (Режим отображения) в **world space** (глобальное пространство).(Рис.4)

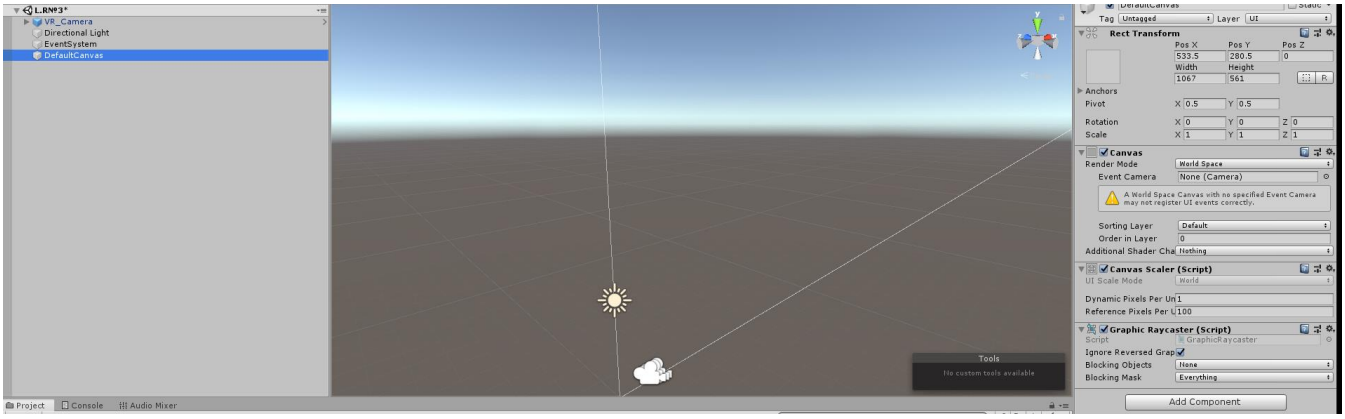


Рис.3 (Default Canvas)

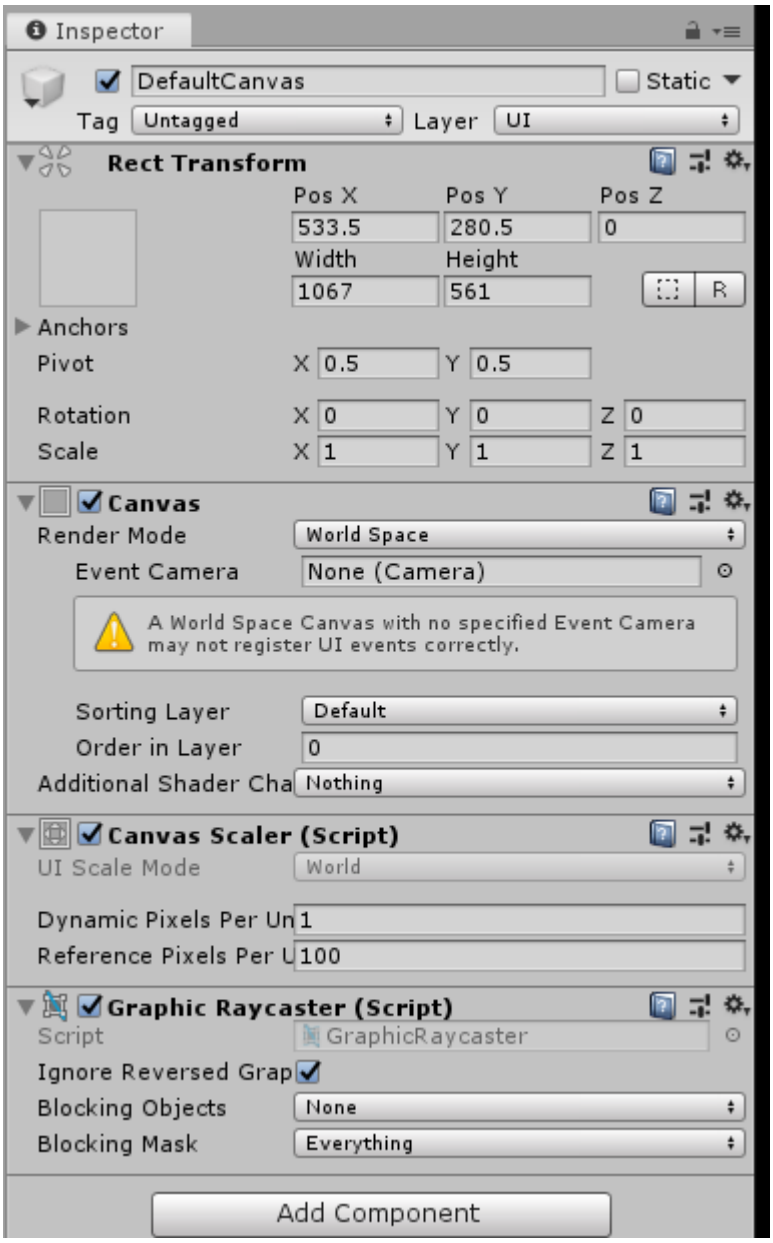


Рис.4. (Первичная настройка Canvas)

Компонент **Rect Transform** (Прямоугольное преобразование) определяет систему разметочной сетки, похожую на разлиновку на листе миллиметровки. Она служит для размещения элементов пользовательского интерфейса. Выберем для нее размеры 64Ox480 с соотношением сторон 0. 7 5. Ширина и высота компонента **Rect Transform** отличаются от размеров объекта Canvas в сцене. Произведите следующие настройки:

1. В компоненте **Rect Transform** установите **Width** = 640 **Height** = 480.
2. В разделе **Scale** (Масштаб) установите **X, У, Z** равными (0.00135, 0.00135, 0.00135). Это размер одного пикселя в единицах измерения игрового пространства.
3. В компоненте **Rect Transform** установите для позиций **Pos X, Pos У, Pos Z** значения (0, 1.325, 0).(Рис.5)

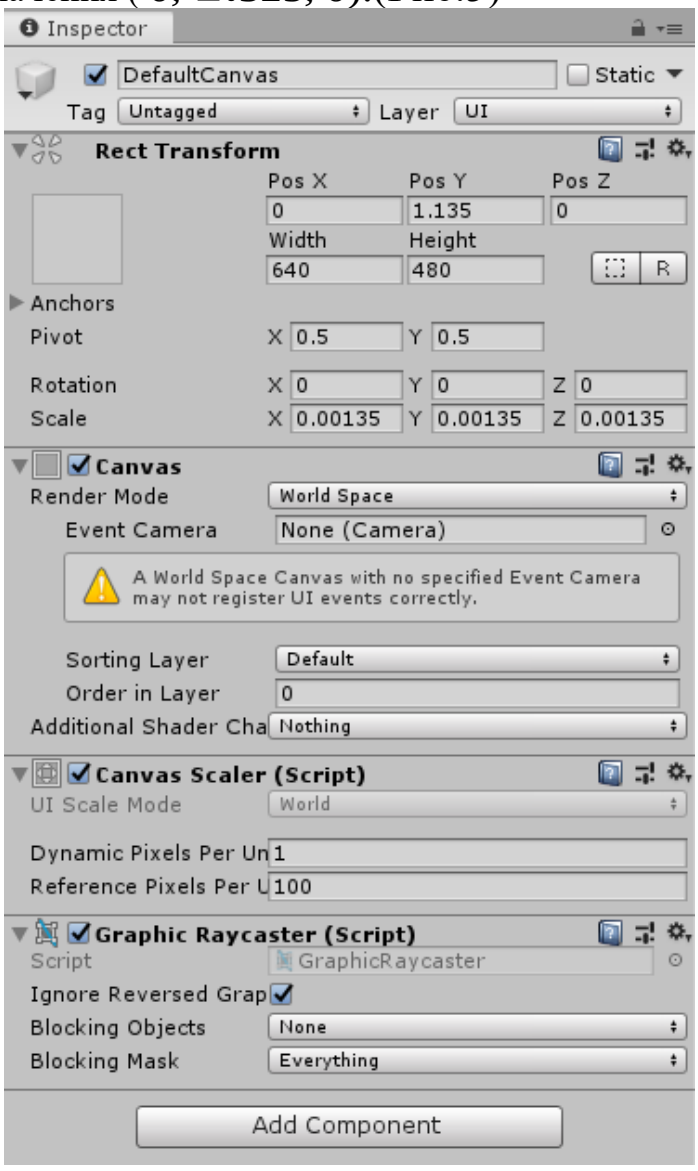


Рис.5.(Настройка компонента Rect Transform)

Далее потребуется добавить пустой элемент **Image** (Изображение) с белым фоном. При его отключении фон будет прозрачным, а при включении - непрозрачным. В этих же целях можно использовать элемент пользовательского интерфейса **Panel** (Панель):

1. При выделенном объекте DefaultCanvas выберите пункт **GameObject ~ Image** (Игровой объект ~ Пользовательский интерфейс ~ Изображение). Вновь созданный объект должен являться дочерним по отношению к объекту DefaultCanvas;
2. При выделенном объекте Image в верхнем левом углу секции компонента **Rect Transform** находится кнопка **anchor presets** (преустановки якоря), (Рис.6). При щелчке по ней откроется диалоговое окно **anchor presets.** Нажмите и удерживайте клавишу **Alt** для того, чтобы сделать видимыми настройки **stretch** (растяжение) и **position** (позиционирование), выберите расположенный в нижнем правом углу пункт **stretch - stretch.** Теперь пустое изображение заполняет весь холст. (Рис.7)

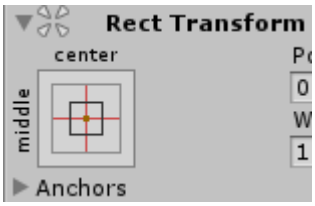


Рис.6. (Anchor)

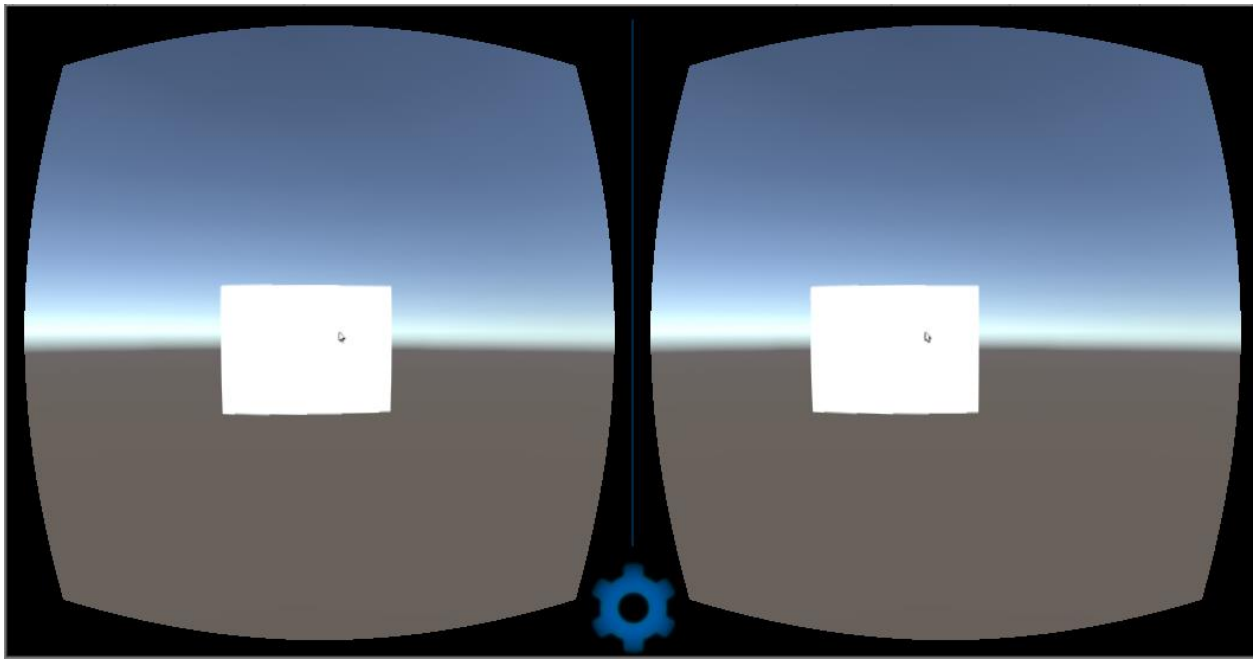


Рис.7. (UI Image)

Добавим элемент **Text** (Текст):

1. При выделенном объекте DefaultCanvas выберите пункты **GameObject ~ UI~Text** (Игровой объект~ Пользовательский интерфейс ~ Текст). Созданный объект должен являться дочерним по отношению к объекту DefaultCanvas). На холсте должны появиться слова **New Text** (Новый текст).(Рис.8)
2. При выделенном объекте **Text** установите **Alignment** (Выравнивание) в **Center Align** (По центру) и **Middle Align** (Посередине) и установите **Vertical Overflow** (Переполнение) в **Overflow** (Переполнение). Установите значение **Scale** (Масштаб) равным (4, 4, 4).
3. При выделенном объекте **Image** установите его кнопку **anchor presets** в **stretch - stretch** с помощью виджета в верхнем левом углу раздела **Rect Transform.**

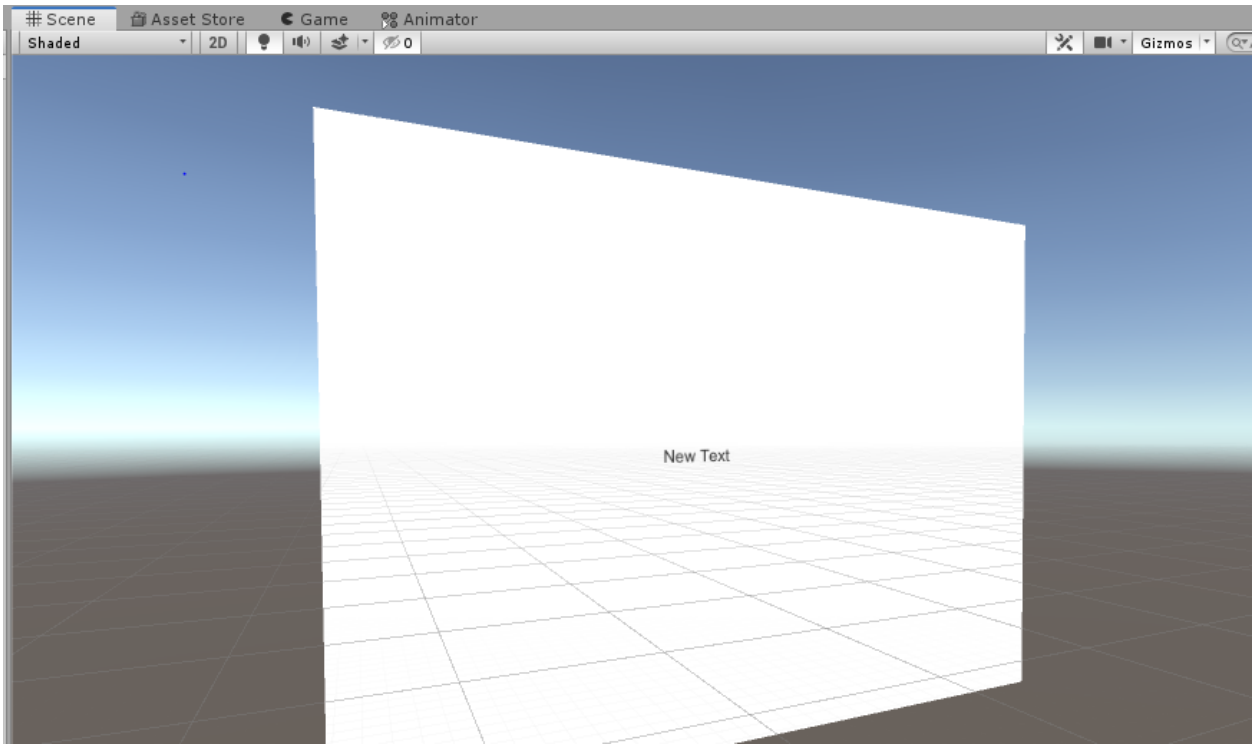


Рис.8. (UI Text)

Увеличьте пиксельное разрешение для придания ясности шрифту текста: оставаясь в DefaultCanvas, выберите и установите в **Canvas Scaler ~ Dynamic Pixels Per Unit** (Масштабирование холста ~ Динамических пиксел ей на единицу) значение 10. (Рис.9.)

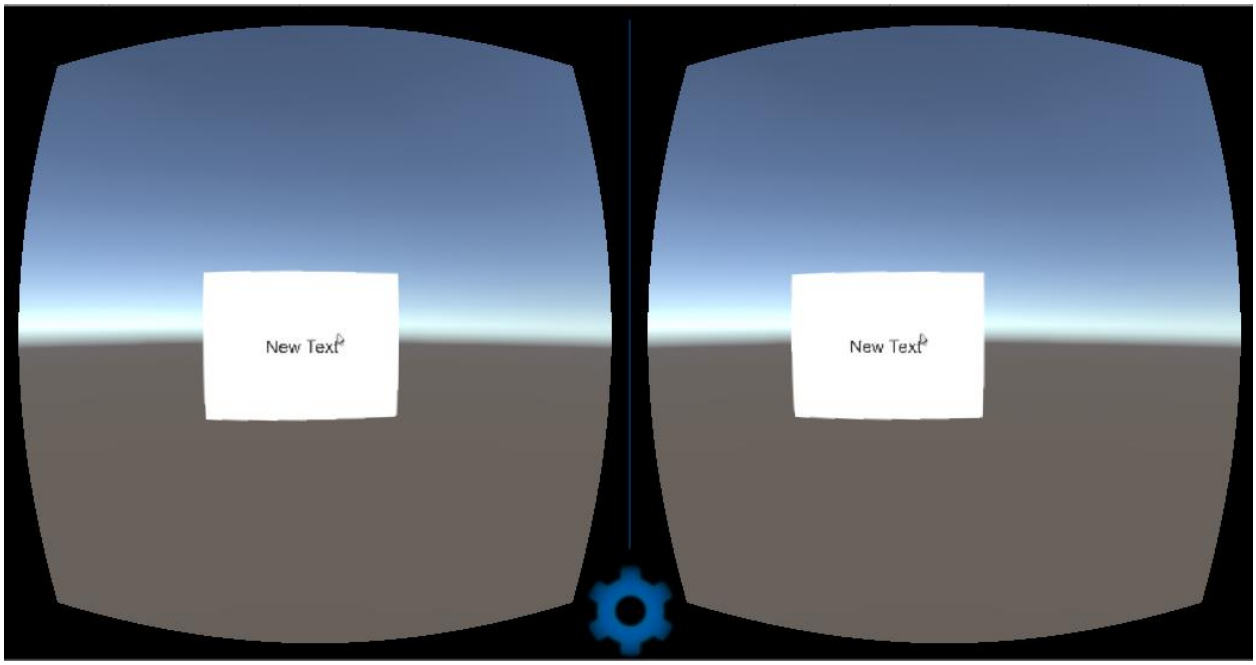


Рис.9. (Итоговые настройки Canvas)

И наконец, сохраните работу как предварительно подготовленный объект, который можно будет повторно использовать (префаб):

1. При необходимости в папке Project Assets создайте новую папку Prefabs.

2. Перетащите объект DefaultCanvas в папку Project Assets/Prefabs для создания предварительно подготовленного объекта

3. Удалите экземпляр DefaultCanvas на панели **Hierarchy** (Иерархия).

Информационный щиток

Индикатор важной информации, или *HUD*, представляет собой видимый холст, перекрывающий игровую сцену. Создадим HUD-щиток.

1. На панели **Hierarchy** (Иерархия) разверните объект VR_Camera

2. Перетащите предварительно подготовленный объект DefaultCanvas с панели **Project** (Проект) на объект VR_Camera так, чтобы он стал его дочерним объектом.
3. Убедитесь, что объект Canvas на панели **Hierarchy** выделен, и переименуйте его в VisorCanvas.
4. На панели **Inspector** (Инспектор) для объекта Canvas измените значения свойства **Rect** компонента Transform **Pos X, Pos У, Pos Z** на

(0,0,1)

5. Раскройте объект VisorCanvas и выберите дочерний объект **Text.**
6. На панели **Inspector** измените текст **Default Text** (Текст по умолчанию)
7. Измените цвет текста на более яркий, например, зеленый.
8. Отключите объект **Image,** чтобы отображался только текст, сбросив флажок **Enable** (Включить) на панели **Inspector.**
9. Сохраните сцену и проверьте ее в VR (Рис.10.)

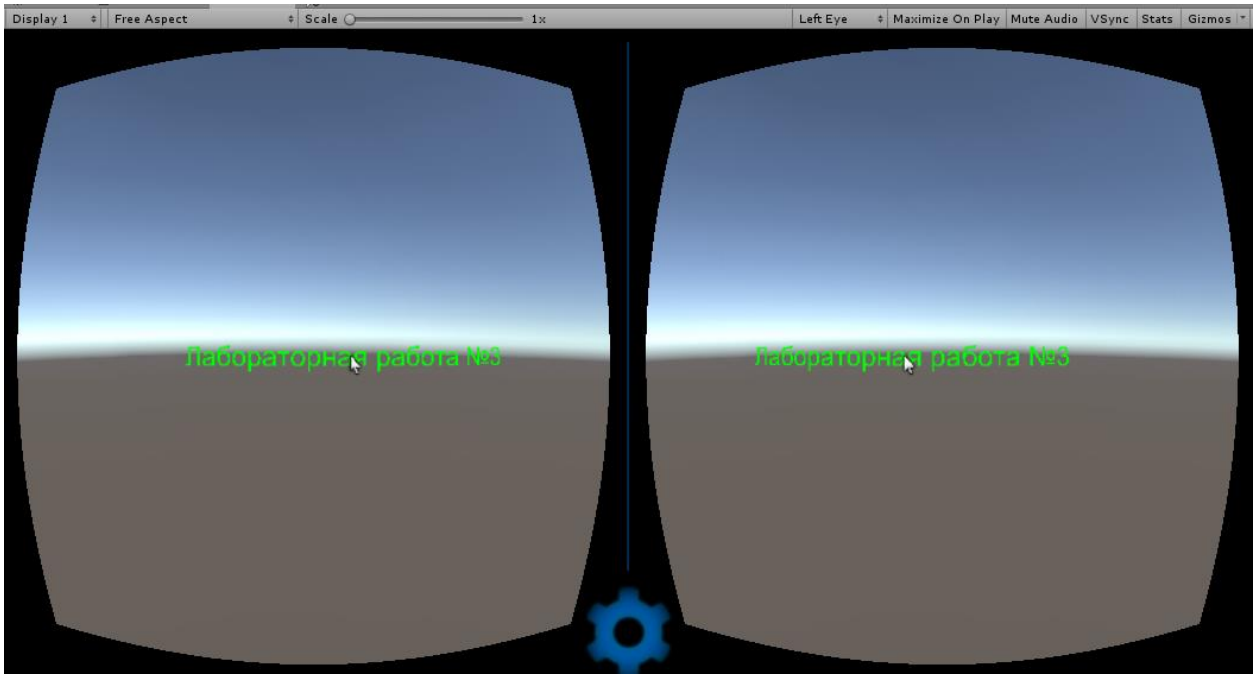


Рис.10. (HUD-щиток)

Курсор в виде перекрестья

1. Найдите объект **VR_Camera** в панели **Hierarchy** (Иерархия).
2. Перетащите с панели **Project** (Проект) предварительно подготовленный объект DefaultCanvas на объект камеры, чтобы сделать его дочерним объектом камеры. Назовите его ReticleCursor.
3. Установите значения свойств компонента Rect Transform **Pos X, Pos У, Pos Z** равными (0,0,1)
4. Удалите его дочерние объекты Image и Text.
5. Добавьте ему дочерний объект Raw Image, выбрав в главном меню

GameObject ~ UI ~ Raw Image (Игровой объект ~ Пользовательский интерфейс ~ Необработанное изображение) и удостоверьтесь, что вновь добавленный объект является дочерним объектом ReticleCursor.

В разделе **Rect Transform** панели для объекта **Raw Image** установите значения **Pos X, Pos У, Pos Z** равными (о, о, о) и значения **Width, Height** (Ширина, Высота) - (22, 22). За тем выберите какой-нибудь бросающийся в глаза цвет, например красный, в свойстве **Color** (Цвет) компонента **Raw Image (Script).**

6. Сохраните сцену и проверьте ее в VR

Лабораторная работа №3.2

Цель работы: Разработка графического интерфейса для приложения виртуальной реальности

Задачи для достижения поставленной цели:

1. Первичная настройка сцены
2. Настройка VR камеры и контролера движения FibrumSDK
3. Создание и настройка объекта Canvas
4. Создание кнопки выключения «проводника»
5. Создание кнопки на переход в другую папку
6. Создание кнопки на открытие настроек локального диска
7. Собрать и запустить разработанный проект

Ход работы:

Создадим новую сцену и подготовим её удалив объект main camera и добавив plane.

Из префабов FibrumSDK возьмем объект Joystick_Simple_character и добавим его на сцену переместив в окно иерархии.

В префабе Joystick_Simple_character отключим его дочерний объект VR_Camera убрав «галочку» рядом с именем объекта (Рис.1) и в его настройках уберем Bullet Prefab, нажав на кружок справа от настроек и выбрав None (Рис.2)

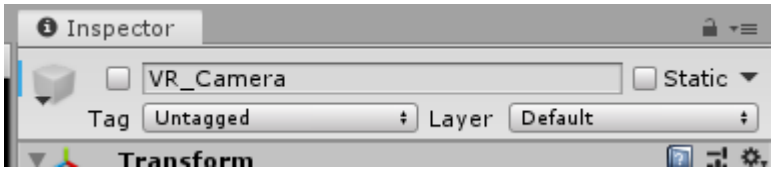


Рис.1 настройка Joystick_Simple_character

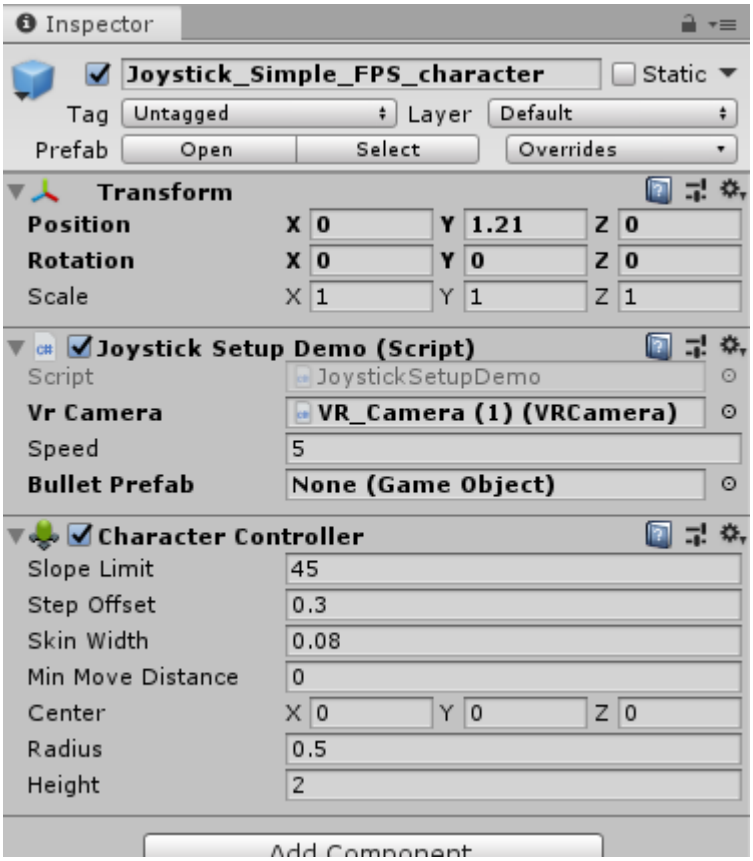


Рис.2 настройка Joystick_Simple_character

Префаб VR_Camera, таким же образом перемещаем на сцену, сделав его дочерним объекта Joystick_Simple_character.

Добавим на сцену объект Canvas, в настройках Render mode выбрав World Space, затем добавим UI объект RawImage, сделав его дочерним Canvas

Далее необходимо сделать скриншот проводника, сохранить его и переместить в настройки RawImage (RawImage (Script) ~ Texture) и настроить размеры этих объектов используя меню Rect Transform (Рис.3)

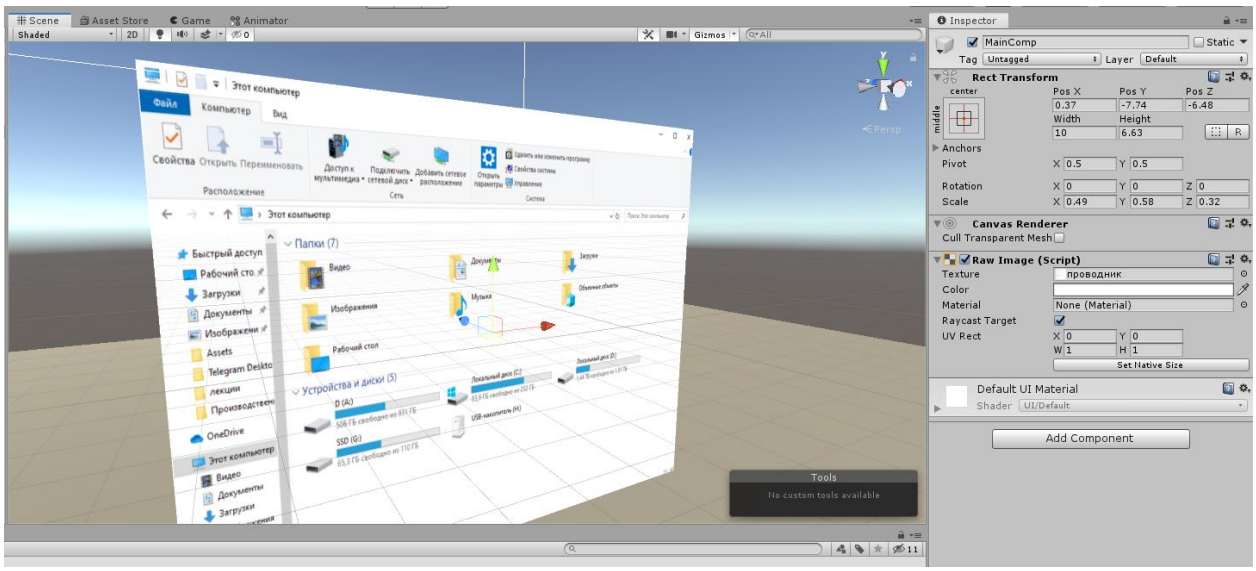


Рис.3 Настройка RawImage

Добавим новый UI объект Button который будет закрывать наше меню. Необходимо сделать его дочерним RawImage.

После этого переместим его на кнопку закрытия проводника, предварительно настроив его размер(Рис.4)

Так-же сделаем кнопку прозрачной, выбрав настройки цвета и зададим альфа каналу значение в 0.

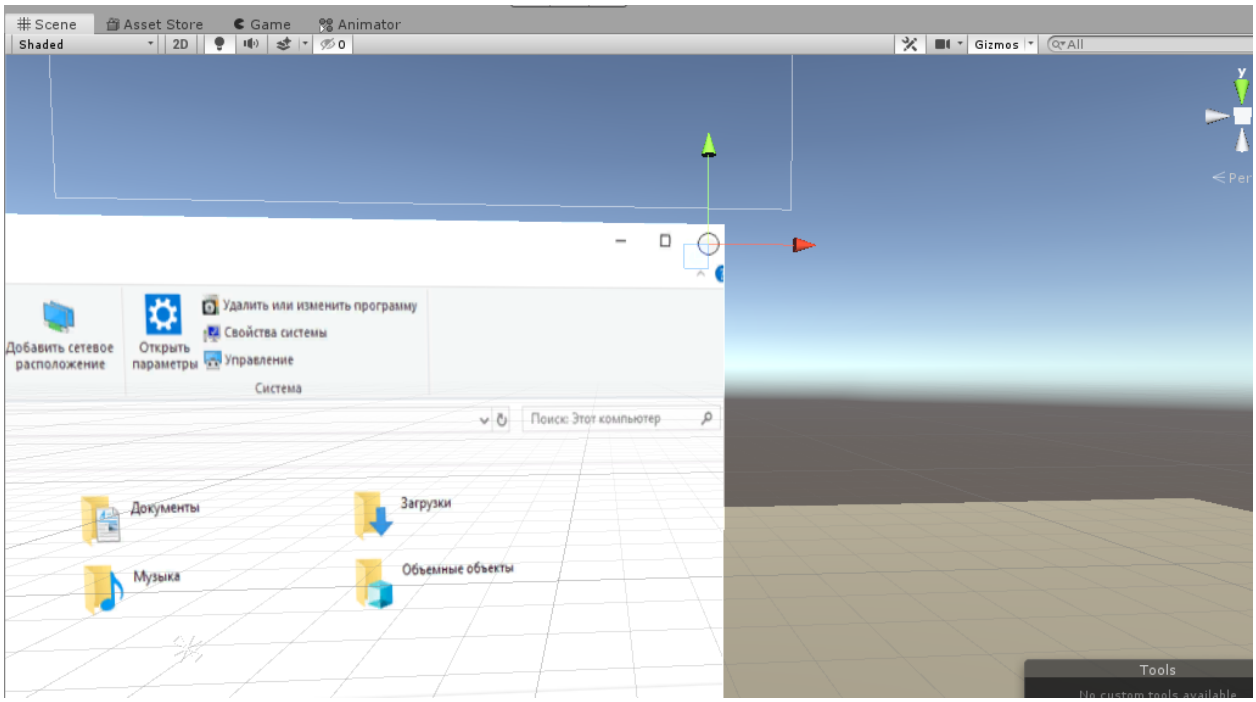


Рис.4 Настройка UI объекта Button

В настройках инспектора в меню On Click () нажмем на «плюс» и создадим новый сценарий закрытия объекта Canvas (Рис. 5)

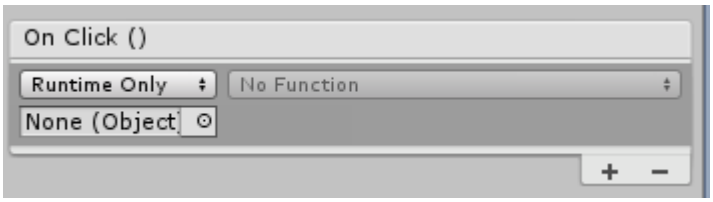


Рис.5 Создания сценария на закрытие объекта Canvas

В нижнее левое меню необходимо перетащить объект Canvas, либо же выбрать его через меню, нажав на круг.

В настройках функции выбираем GameObjeckt SetActive (bool)

Теперь при наведении и нажатии на кнопку проводник закроется(Рис.6)

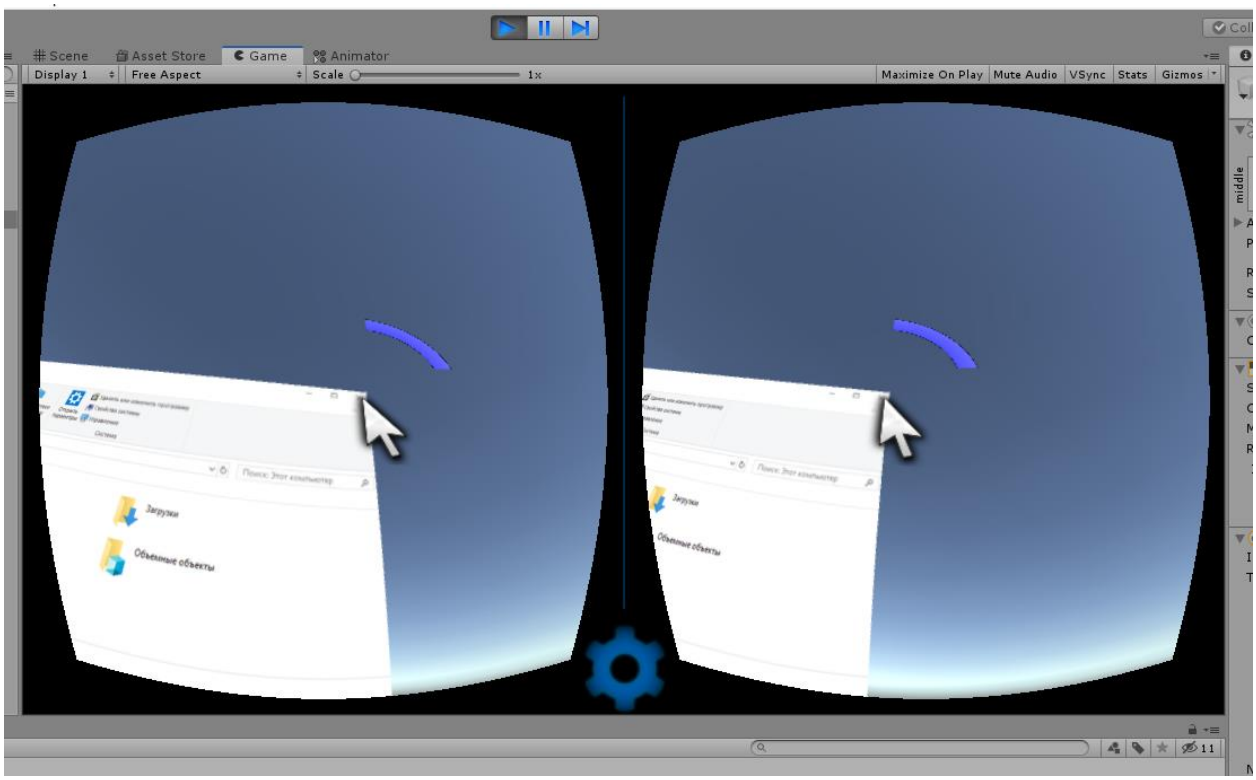


Рис. 6 Закрытие проводника.

По такому же принципу сделаем кнопку на открытие проводника, используя другой Canvas.

Создадим еще две кнопки и два RawImage и разместим их на локальных дисках C и D соответсвенно.

Выключим созданные RawImage.

Далее требуется написать скрипт на открытие другого окна(Рис.7)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

Ссылок: 0
public class ToC: MonoBehaviour
{
    Ссылок: 0
    public void Open(GameObject Obj)
    {
        Obj.SetActive(true);
    }
}
```

Рис.7 Скрипт на открытие GameObjeckt

Добавляем этот скрипт на объект Canvas

В настройках добавленных кнопках в меню On Click () теперь появиться новый метод, написанный нами для открытия других объектов. (Рис.8)

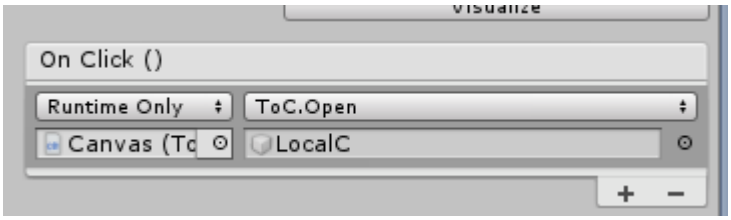


Рис.8(Новый метод открытия объектов)

Вывод: Была проведена настройка контролера движения для VR камеры, были изучены варианты создания графического пользовательского интерфейса в среде виртуальной реальности