**ФУНКЦИОНАЛ СИМУЛЯТОРА УСТАНОВКИ**

**ЗАПОЛНЕНИЕ ТАБЛИЦЫ РЕЗУЛЬТАТОВ**

**ПУБЛИКАЦИЯ НА САЙТЕ**

**СИСТЕМА ЗАДАНИЙ**

Первое задание при начале работы с симулятором это, как правило, задание «Включить установку».

После щелчка по кнопке «РАБОТА УСТАНОВКИ» выполняется событие вывода в текстовое окно сообщения: «Включите установку».

Первое задание выполняется при совершении события включения установки, когда пользователь щелкнет мышью на объект «Переключатель», что и приведет к выполнению первого задания.

После выполнения первого задания программа должна выдать в информационное окно следующее задание, которое должно выполняться в соответствии с появляющимся текстовым сообщением и т. д. вплоть до окончания работы с симулятором.



Результатом работы с симулятором установки являются полученные по ходу выполнения заданий измеренные значения, показания приборов и т. п., которые последовательно заносятся в таблицу с выполнением в ней предусмотренных в каждом конкретном эксперименте вычислений.

**ФРАГМЕНТ КОДА ДЛЯ ВЫПОЛНЕНИЯ ЗАДАНИЙ ДЛЯ СИМУЛЯТОРА «КОЛЬЦА НЬЮТОНА»**

public class Tasks : MonoBehaviour

private void Start()

{

taskText.enabled = false;

taskText.text = "Включите установку";

}

void WorkingControl(bool work) //постоянно отслеживаем включена ли установка

{

isWorking = work;

}

void Task1(bool w) //первое задание вызовется после включения установки

{

if (w)

{

taskText.text = "Поворотом барабана, установите <color=#008000ff><b>зеленый</b></color> светодиод";

}

}

void Task2(Color color) //второе задание вызовется после выполнения первого задания

{

if (color == Color.green) // проверка условия, что включен именно зеленый цвет

{

taskText.text = "Посмотрите в окуляр микроскопа";

}

}

void Task3(bool mainView) //третье задание вызовется после выполнения второго задания

{

if (!mainView && isWorking) //проверка сразу двух условий

{

taskText.text = "Впишите в таблицу радиус первого кольца в малых делениях при <color=#008000ff><b>зелёном</b></color> свете";

}

bool Task4()  
    {  
        if (TaskControl(Color.green))  
        {  
            taskText.text = ++numberOfTask + ". " + "Впишите в таблицу радиус второго кольца в малых делениях при <color=#008000ff><b>зелёном</b></color> свете";  
  
        }

//…

//и т.д. в зависимости от количества заданий (этапов) работы с симулятором установки

// ( всего в шаблоне симулятора «Кольца Ньютона» 18 заданий)

}

**ВЫВОД ЗНАЧЕНИЙ В ТАБЛИЦУ**

Для сохранения результатов работы с симулятором необходимо создать таблицу, в которую будут заносится результаты измерений показаний приборов установки или значений характеристик величин, получаемых при работе с установкой.

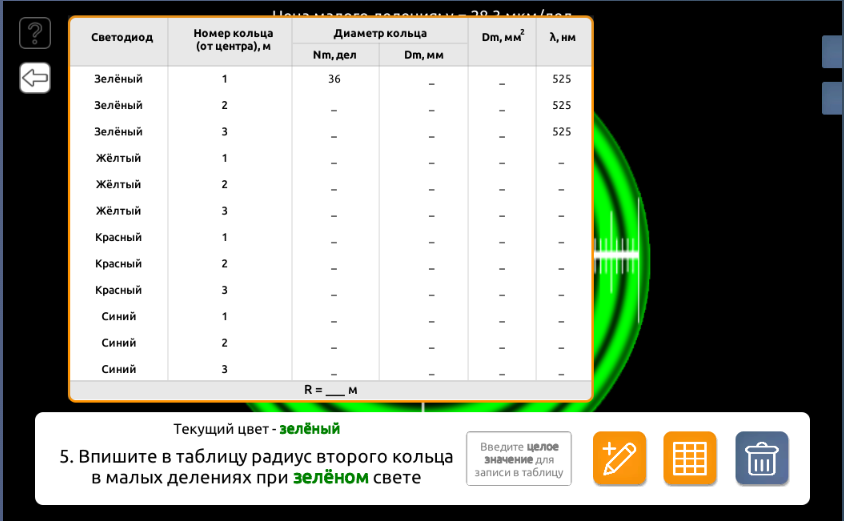
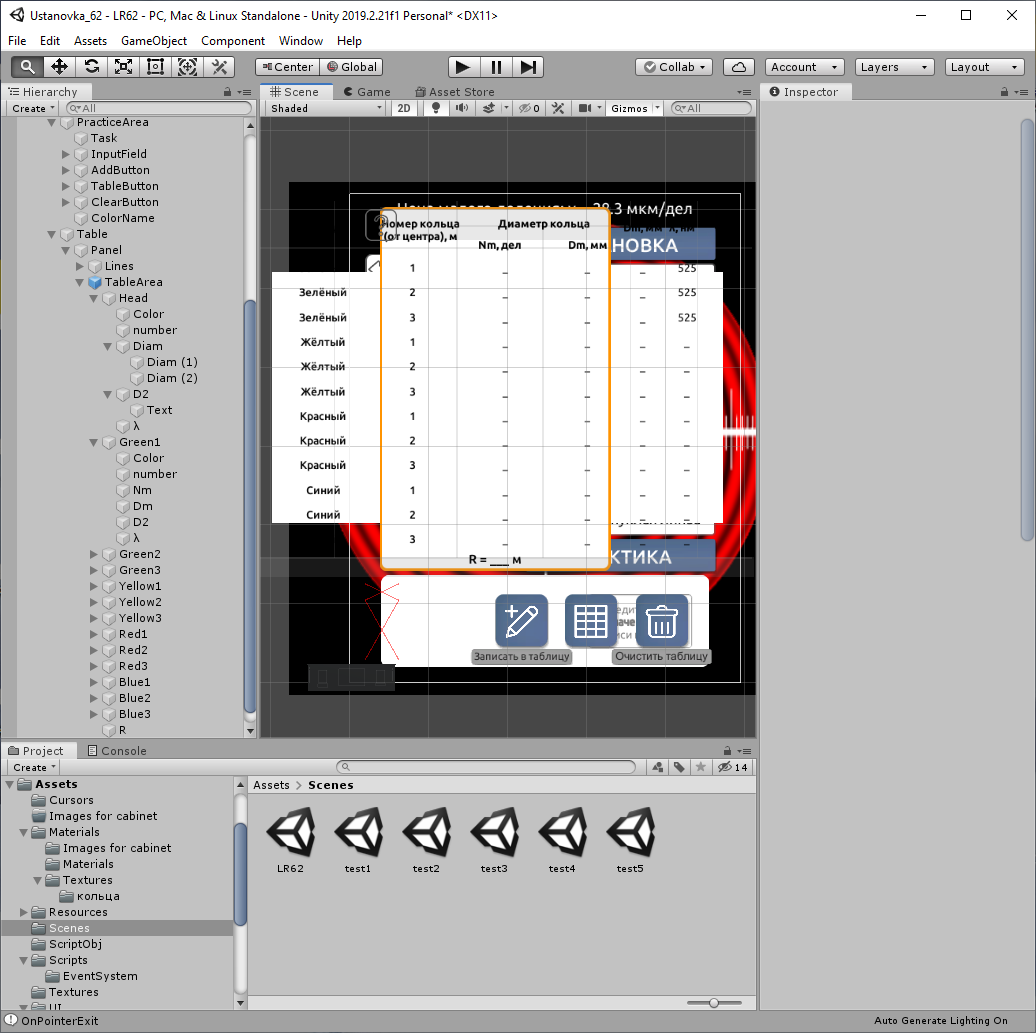
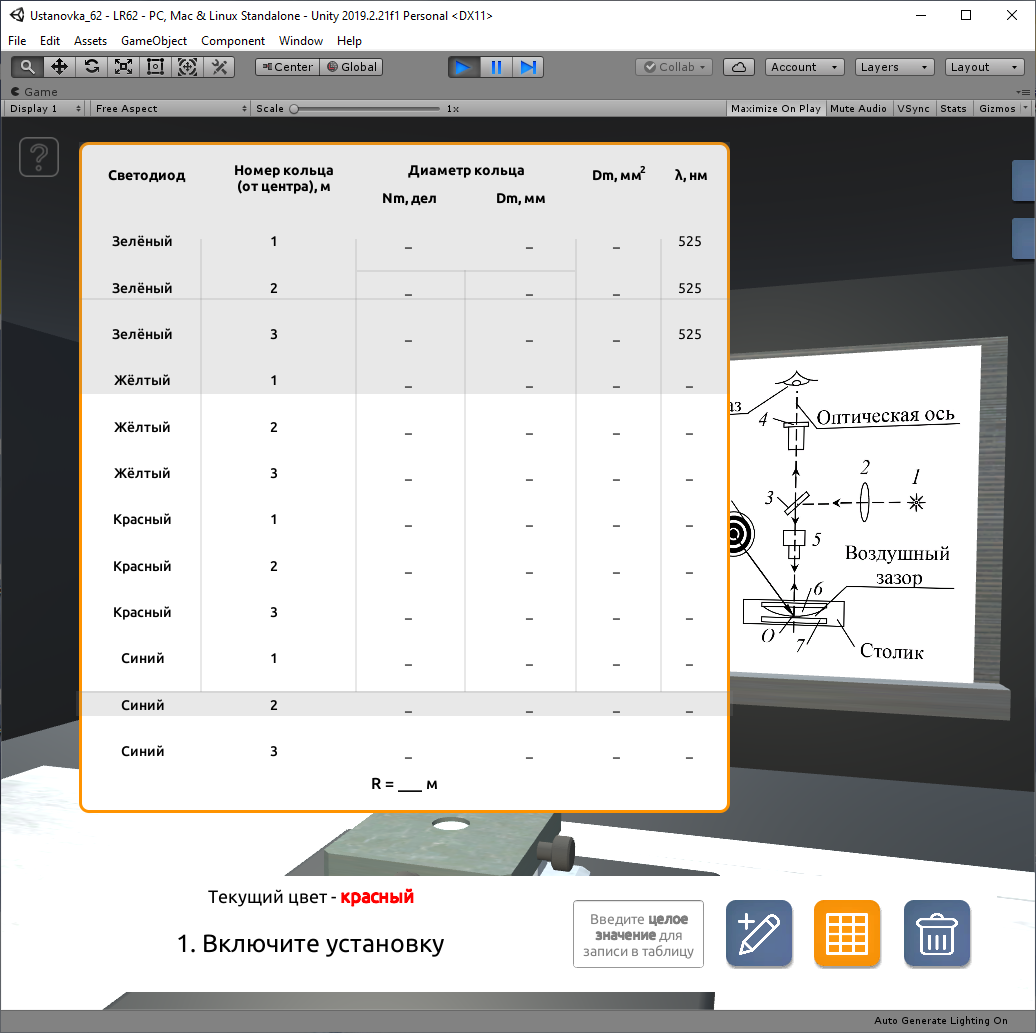


Таблица состоит из массива текстовых полей, в которые записываются результаты работы с симулятором.



Ячейки таблицы заполняются последовательно частично вручную, частично автоматически по формулам.

Для работы с таблицей на информационной панели для практики должны быть предусмотрены кнопки **Button:** кнопка для записи значения в таблицу, кнопка для отображения таблицы и кнопка для очистки таблицы (на рисунке ниже для них использованы соответствующие изображения).



Также на информационной панели должно присутствовать текстовое поле **InputText** для ввода полученных значений со шкалы прибора и занесения его в таблицу.

В ходе выполнения лабораторной работы снятые измерения с установки записываются в нужные ячейки таблицы по нажатию кнопки «Записать», для просмотра содержания таблицы используется событие наведение курсора на кнопку «Таблица», а для очистки таблицы от записей - наведение курсора на кнопку «Очистка».

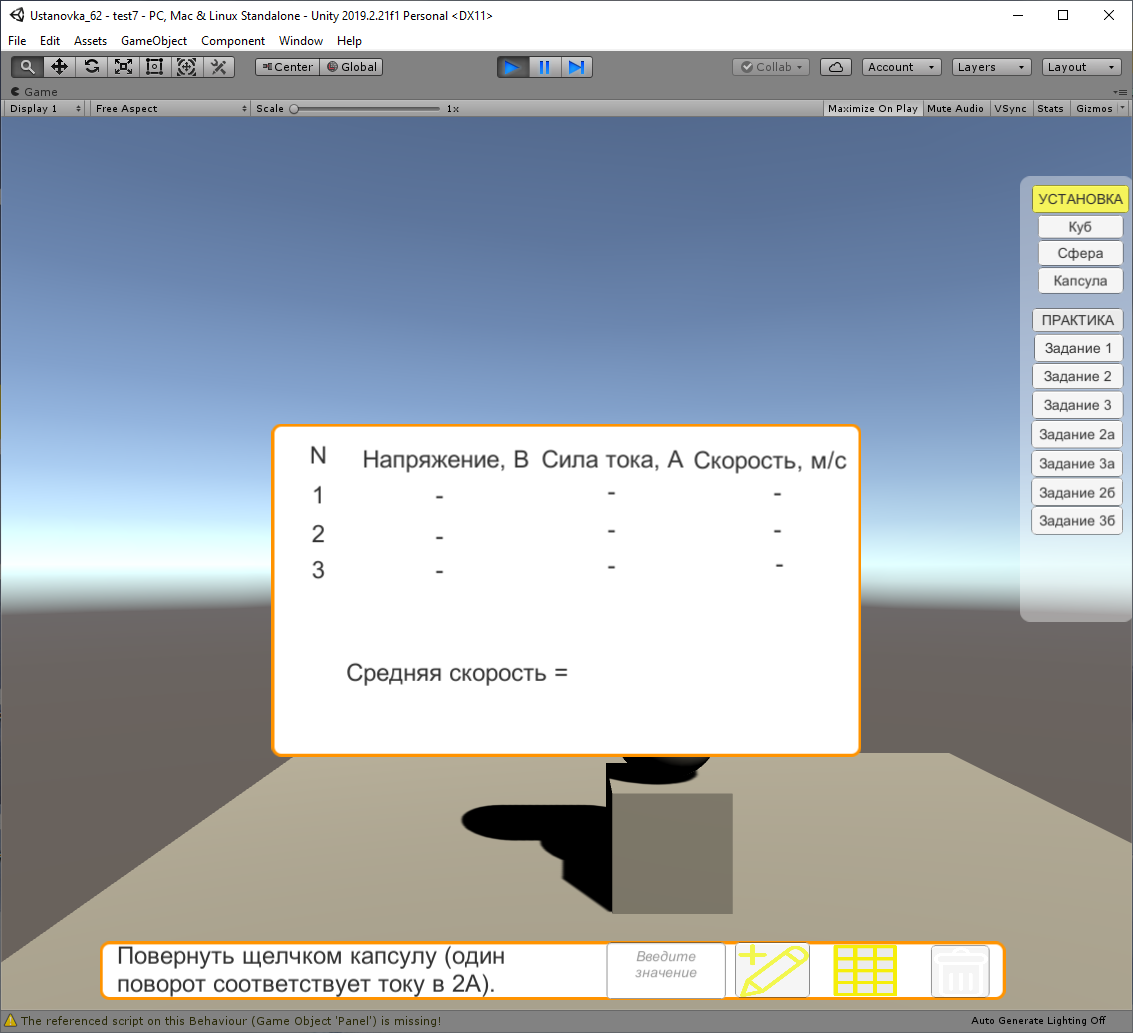
После осуществления всех необходимых заданий из раздела «Практика» выводится соответствующее сообщение о завершении работы с симулятором установки.

**СОЗДАНИЕ И ЗАПОЛНЕНИЕ ТАБЛИЦЫ РЕЗУЛЬТАТОВ РАБОТЫ**

Вся таблица – заголовки, названия полей и т.п. строится из текстовых объектов UI соответствующего размера с фиксированным текстом, а для ячеек, куда нужно записывать результаты эксперимента, используется символ подчеркивания или минуса, чтобы было проще находить нужные ячейки таблицы при записи в них значений по нажатию кнопки «Записать».

Для записи в текстовые объекты (ячейки таблицы) значений используется, например, для текстового объекта **name1** конструкция **name1.text.ToString()**, а для вычислений с записями в текстовых объектах используется конструкция **int.Parse(name1.text).**

СТРУКТУРА ТАБЛИЦЫ ДЛЯ ТЕСТОВОГО ПРИМЕРА УСТАНОВКИ



КОД ЗАПОЛНЕНИЯ И ОЧИСТКИ ТАБЛИЦЫ ДЛЯ ТЕСТОВОГО ПРИМЕРА УСТАНОВКИ

**using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.UI;  
  
public class TableVal : MonoBehaviour  
{  
    [SerializeField]  
    InputField textInput;  
  
    [SerializeField]  
    Text u1;  
    [SerializeField]  
    Text i1;  
    [SerializeField]  
    Text u2;  
    [SerializeField]  
    Text i2;  
    [SerializeField]  
    Text u3;  
    [SerializeField]  
    Text i3;  
    [SerializeField]  
    Text v1;  
    [SerializeField]  
    Text v2;  
    [SerializeField]  
    Text v3;   
    [SerializeField]  
    Text ve;**

**//функция для кнопки «Записать»  
    public void WriteValue()  
    {  
        if (u1.text == "-")  
            u1.text = textInput.text.ToString ();  
        else if (i1.text == "-") {  
            i1.text = textInput.text.ToString ();  
            v1.text = (int.Parse (u1.text) \* int.Parse (i1.text)).ToString ();  
        } else if (u2.text == "-")  
            u2.text = textInput.text.ToString ();  
        else if (i2.text == "-") {  
            i2.text = textInput.text.ToString ();  
            v2.text = (int.Parse (u2.text) \* int.Parse (i2.text)).ToString ();  
        } else if (u3.text == "-")  
            u3.text = textInput.text.ToString ();  
         else if (i3.text == "-") {  
            i3.text = textInput.text.ToString ();  
            v3.text = (int.Parse (u3.text) \* int.Parse (i3.text)).ToString ();  
            ve.text = ((int.Parse (v1.text) + int.Parse (v2.text)+ int.Parse (v3.text))/3).ToString ();  
        }  
        textInput.text="";  
    }  
//функция для кнопки «Очистить»  
    public void Clean()  
    {  
        u1.text = "-";  
        u2.text = "-";  
        u3.text = "-";  
        i1.text = "-";  
        i2.text = "-";  
        i3.text = "-";  
        v1.text = "-";  
        v2.text = "-";  
        v3.text = "-";  
    }  
  
}**

**МИНИМУМ**

**ЭЛЕМЕНТОВ ИНТЕРФЕЙСА СИМУЛЯТОРА УСТАНОВКИ**

**Обязательные объекты симулятора установки:**

1. Main Camera.
2. Directional Light и (или) дополнительное освещение.
3. Помещение (3 стены окружения с тематическими изображениями и стол-подставка для установки).
4. Установка (все статичные и функциональные элементы установки с соответствующими текстурами).
5. Canvas с элементами UI.

**Обязательные скрипты:**

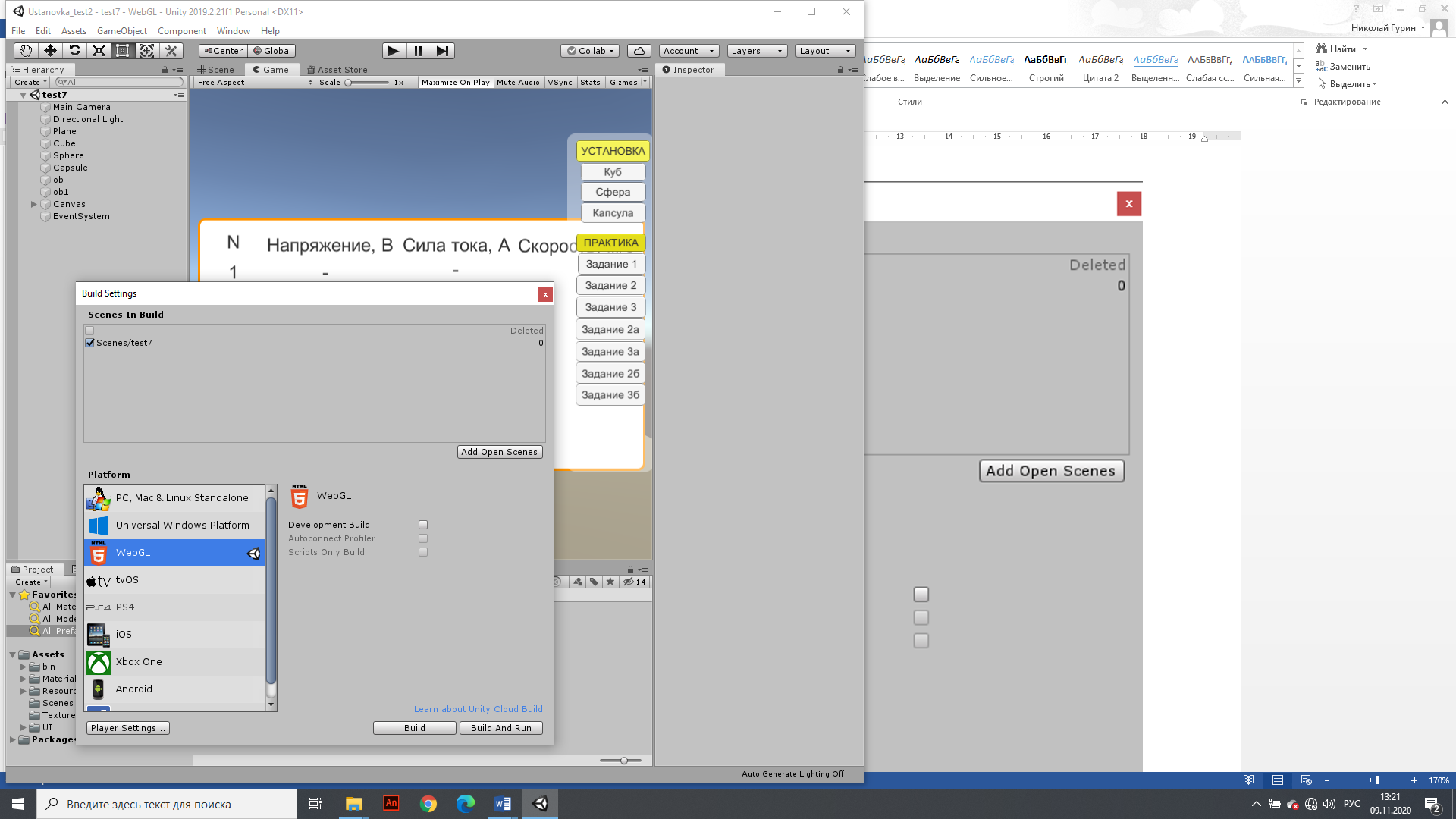
1. Управление камерой.
2. Общая справка-инструкция по работе с установкой.
3. Подсветка выбранного элемента установки.
4. Перемещение и поворот камеры к выбранному элементу установки.
5. Движение панели управления (выдвинуть-задвинуть).
6. Система заданий (последовательное выполнение этапов работы с учетом функциональных зависимостей между элементами установки от момента включения до ее выключения).
7. Ввод значений в таблицу и обработка результатов работы установки.

**Обязательные функции кнопок:**

1. Вывод подсказок в текстовые окна Text на фоне изображения Image (инструкции по использованию симулятора на полный экран, назначения элементов установки с их подсветкой и очередность выполнения заданий – в информационное окно внизу экрана).
2. Перевод камеры в удобный ракурс для просмотра функционального элемента установки.
3. Запись результатов работы в таблицу.
4. Показ таблицы.
5. Очистка таблицы.

**ПРОЦЕДУРА ПУБЛИКАЦИИ ПРИЛОЖЕНИЯ UNITY**

1. При публикации обратить внимание, чтобы в путях к проекту, папки для публикации и к файлам **WebGL** не было пробелов и русских букв.
2. Чтобы в публикации отображался русский текст, нужно выбрать нужный шрифт (с расширением **ttf**), скопировать его в папку Assets проекта, а затем применить в качестве шрифта на компонент с текстом.
3. Для публикации проекта выполнить команду **File/Build Settings**. Далее выбрать платформу **WebGL** и нажать **Switch Platform** (можно предварительно задать более подробные настройки публикации в **Player Settings**).



1. Если в поле **Scene In Build** сразу не добавилась сцена, нужно нажать кнопку **Add Open Scenes**.
2. Нажать кнопку **Build** и дождаться окончания процедуры построения публикации проекта (возможно потребуется несколько десятков минут).
3. После завершения построения будут созданы необходимые папки и файл **index.html**. На этом этапе уже можно запустить проект в браузере ( требует запуска с http сервера), поэтому можно воспользоваться, например, браузером **Mozilla Firefox**.
4. Для того, чтобы встроить Unity-плеер с проектом в уже существующий html, достаточно в требуемое место в существующем html-документе скопировать и вставить фрагмент из только что созданного **index.html** (а также созданные папки при сборке проекта):

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<link rel="shortcut icon" href="TemplateData/favicon.ico">

<link rel="stylesheet" href="TemplateData/style.css">

<script src="TemplateData/UnityProgress.js"></script>

<script src="Build/UnityLoader.js"></script>

<script>

var unityInstance = UnityLoader.instantiate("unityContainer", "Build/student.json", {onProgress: UnityProgress});

</script>

</head>

<body>

<div class="webgl-content">

<div id="unityContainer" style="width: 960px; height: 600px"></div>

</div>

</body>

Для создания ***независимого exe-файла*** разработанного приложения для симулятора установки, выполняемого вне среды **Unity** и без использования плеера **WebGL** на html-странице, необходимо, действуя аналогично предыдущей процедуре, выполнить в пункте 3 процедуры команду **File/Build Settings/ Universal Windows Platform**.

В результате будут созданы необходимые для запуска файлы и папки, а также exe-файл для запуска приложения.

*При этом нужно учитывать, что возможно при установке Unity соответствующий модуль для публикации разработанного приложения в формате WebGL или exe-файла не был установлен и его потребуется догрузить*.