**ФУНКЦИОНАЛ СИМУЛЯТОРА УСТАНОВКИ**

**ЗАПОЛНЕНИЕ ТАБЛИЦЫ РЕЗУЛЬТАТОВ**

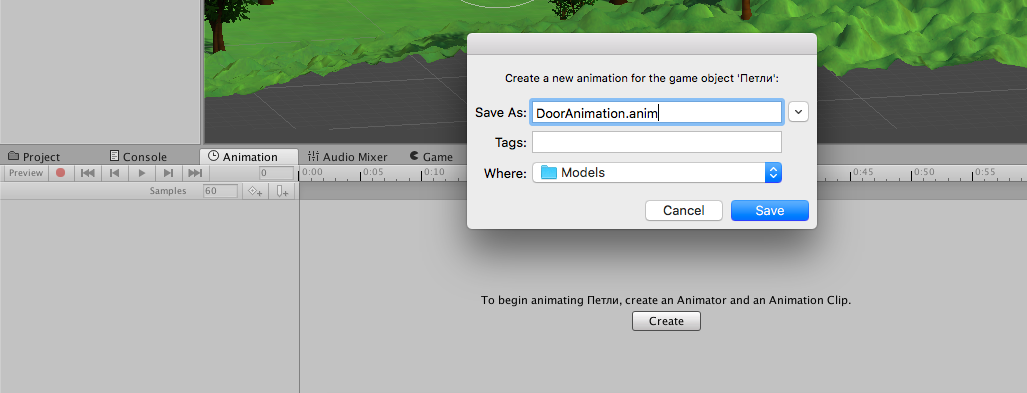
**ПУБЛИКАЦИЯ ПРИЛОЖЕНИЯ**

**СОЗДАНИЕ АНИМАЦИИ ОБЪЕКТА**

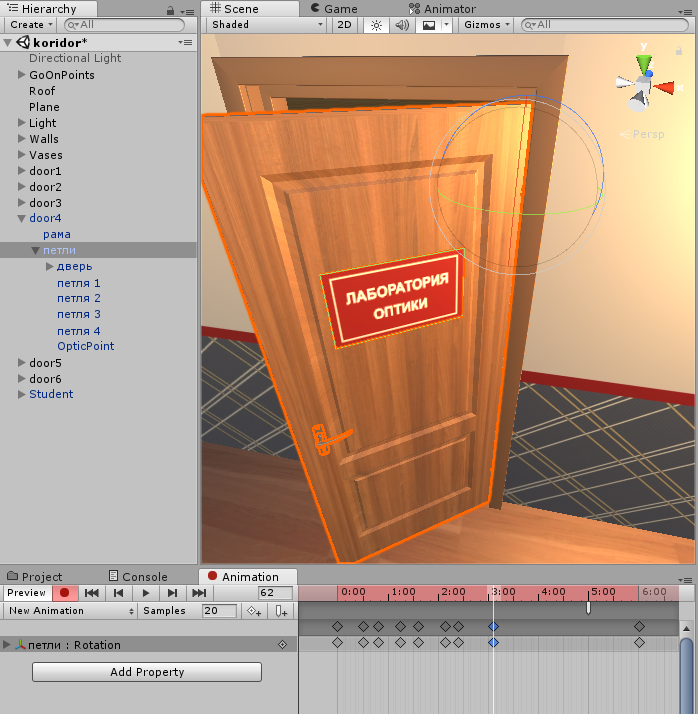
**В СРЕДЕ UNITY**

Для создания анимации в редакторе **Unity** нужно выбрать объект, который будет анимироваться, открыть для него вкладку **Window/Animation/Animation** и создать, прежде всего, отдельный файл анимации в открывающемся окне, нажав кнопку **Create**.

После сохранения файла анимации с типом **\*.anim** в папке проекта **Assets** создается в дальнейшем содержание анимации объекта.



На временной шкале в нижней части окна редактора **Unity** нажать кнопку с красным кружком (на рисунке) и в соответствующих ключевых кадрах записать требуемую анимацию для объекта.

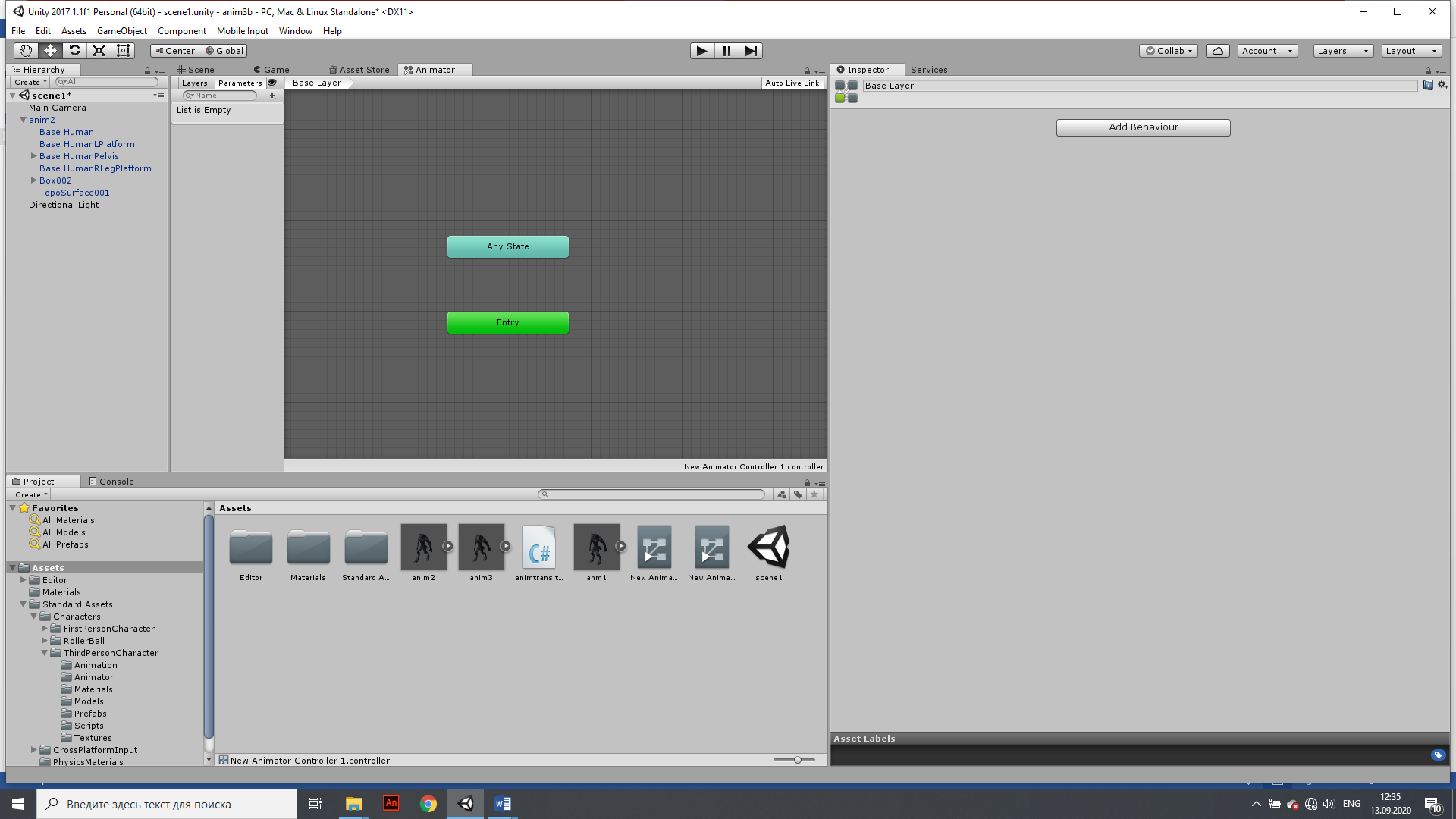


*При создании анимации необходимо учесть, что, если объект создан в среде* ***3dsMAX,*** *то при его анимируемом повороте, например открытии дверки духового шкафа, точка центра вращения* ***Pivot*** *по умолчанию будет расположена по центру. Чтобы объект вращался правильно вокруг края дверки, нужно в* ***Unity*** *создать пустой объект типа* ***Empty Object*** *(невидимый при проигрывании) и в окне* ***Hierarchy*** *сделать анимируемую дверку дочерним элементом этого пустого объекта, при этом разместить предполагаемую точку вращения в центре пустого объекта.*

**СОЗДАНИЕ КОНТРОЛЛЕРА АНИМАЦИИ**

Затем нужно добавить к объекту, контроллер анимации **Animator** – кликнуть правой кнопкой мыши по свободной области в папке **Assets**, выполнить команду **Create->Animator Controller** и дать имя контроллеру. Дважды кликнув на созданный контроллер, перейти в окно редактора.

В начале есть два блока, это **Entry**, с которого будут начинать проигрываться анимации и **Any State** – особое состояние, которое всегда существует и позволяет перейти от любого состояния к конкретному.



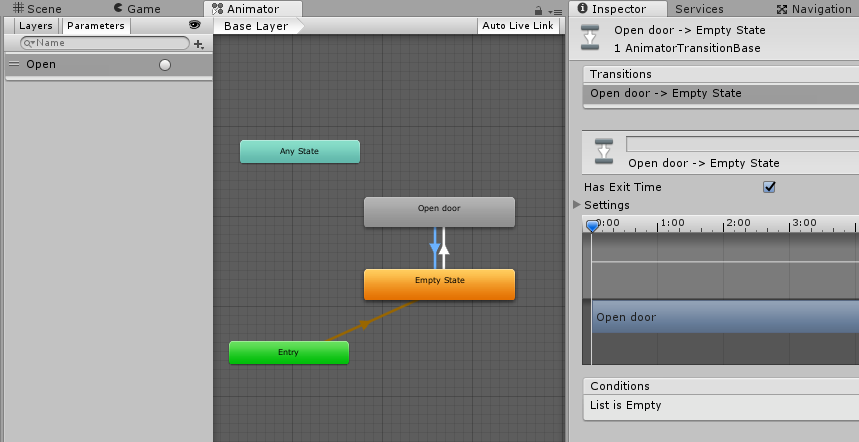
Создать **Empty State** как состояние по умолчанию, щелкнув правой кнопкой мыши в свободной части окна аниматора и выполнив команду **Create State/Empty,** в качестве состояния покоя у двери – это состояние добавится в окно **Animator** и будет по умолчанию оранжевого цвета.

Затем нужно перетащить в окно **Animator** из папки **Assets** созданную анимацию, которая будет представлена серым прямоугольником с именем анимации как отдельное состояние.

Переходы к состоянию анимации открытия двери и обратно должны происходить по булевой переменной **Bool**, причем обратно - по условию установки режима **Has Exit Time**. При этом предварительно для созданной анимации двери необходимо *снять зацикливание* **Loop Time** в контроллере анимации, после того, как она будет выбрана в папке **Assets**.

Для регулировки скорости воспроизведения анимации необходимо для состояния анимации в окне **Animator** необходимо задать ее величину в окне режима **Speed**, используя движение по нему курсором мыши.

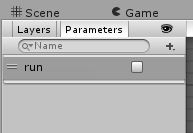
Для создания в окне **Animator** перехода от одного состояния к другому (отображается на экране стрелкой перехода) необходимо щелкнуть правой кнопкой мыши по начальному состоянию и выбрать в контекстном меню команду **Make Transition,** затем навести кусор мыши на прямоугольник другого состояния и щелкнуть по нему мышью, аналогичным образом можно создать обратный переход.



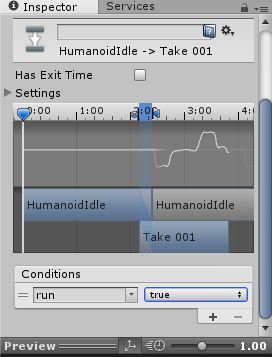
**УПРАВЛЕНИЕ АНИМАЦИЕЙ В UNITY НА ЯЗЫКЕ C#**

Для программного управления анимацией естественно необходимо будет создать программный код на языке C# в скрипте, который должен быть довавлен к анимируемо-му объекту.

Однако, прежде всего, в окне аниматора необходимо **создать переменную** для перехода к анимации, открыввкладку **Parameters,** затем нажав плюс (добавить), затем выбрав тип создаваемой переменной, например, **Bool** (булевского типа) и, наконец, задав имя переменной, например **run**.



После этого выбрать в окне редактора щелчком мыши стрелку перехода от исходного состояния покоя к состоянию анимации движения. Затем в **Inspector** для этого перехода снять галочку для режима **Has Exit Time** и в списке **Conditions** и добавить в него переменную **run** со значением **true**, нажав кнопку **+**.



Чтобы создать обратный переход к состоянию покоя от анимации необходимо выбрать щелчком мыши стрелку обратного перехода, но теперь установить галочку **Has exit time**, а затем в списке **Conditions** добавить (нажав значок +) переменную **run**, но уже со значением **false**.

**ДОБАВИТЬ СКРИПТ ДЛЯ ПЕРЕКЛЮЧЕНИЯ МЕЖДУ АНИМАЦИЯМИ**

В окне инспектора **Inspector** добавить **Add Component** для объекта новый скрипт **New Script** и в редакторе кода создатьследующий код:

**Animator anim;** //переменная типа **Animator** для ссылки на анимацию

**void Start(){**

**anim = GetComponent<Animator>();** //инициализация контроллера анимации

**}**

**void Update()**

**{**

**If(Input.GetKeyDown(KeyCode.Q))** //если нажата клавиша **Q**

**{**

**anim.SetBool(“run”, true);** // переменная, отвечающая за переход имеет значение **true**

**}**

**If(Input.GetKeyDown(KeyCode.E))** //если нажата клавиша **E**

**{**

**anim.SetBool(“run”, false);** // переменная, отвечающая за переход имеет значение **false**

**}**

**}**

**ПРОИГРАТЬ АНИМАЦИЮ ИЗ ЛЮБОГО СОСТОЯНИЯ**

Чтобы проиграть какую-либо анимацию можно было из любого состояния, необходимо:

1. создать переход от **Any State** к этой анимации, например с именем **Hit**. Для возвращения в состояние ожидания необходимо создать переход от новой анимации **Hit** к анимации по умолчанию.
2. создать еще одну переменную типа **Trigger** (например **hitten**), которая будет отвечать за щелчок мышью для запуска анимации. Добавим к переходу от **Any State** эту переменную, а для обратного перехода оставим галочку **Has Exit Time**, чтобы переход совершался автоматически (без использования переменной) один раз после проигрывания анимации.
3. добавить в метод **Update()** код проигрывания новой пользовательской анимации по клику мыши:

**if (Input.GetMouseButtonDown(0))  {  anim.SetTrigger("hitten"); }**

**СИСТЕМА ЗАДАНИЙ**

Первое задание при начале работы с симулятором это, как правило, задание «Включить установку». После щелчка по кнопке «ПРАКТИКА» выполняется событие открытие всплывающего окна **информационной панели** и 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 окноследующее 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.

Для подключения функционала установки при работе с симулятором необходимо создать соответствующий **скрипт**, добавляемый на объект симулятора, с последовательным вызовом из него функций, выполняющихся при последовательном нажатии на кнопки выполнения этапов работы с симулятором.

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

public class Tasks : MonoBehaviour

void Task0()

{ taskText.text = "Включите установку"; } //вывод сообщения в текстовое окно на информационной панели

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

{ if (w)

{ taskText.text = "Поворотом барабана, установите зеленый 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 зелёном свете"; }

bool Task4()  
    {  if (TaskControl(Color.green))  
        { taskText.text = "Впишите в таблицу радиус второго кольца в малых делениях при зелёномсвете";      }

//…

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

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

//…

}

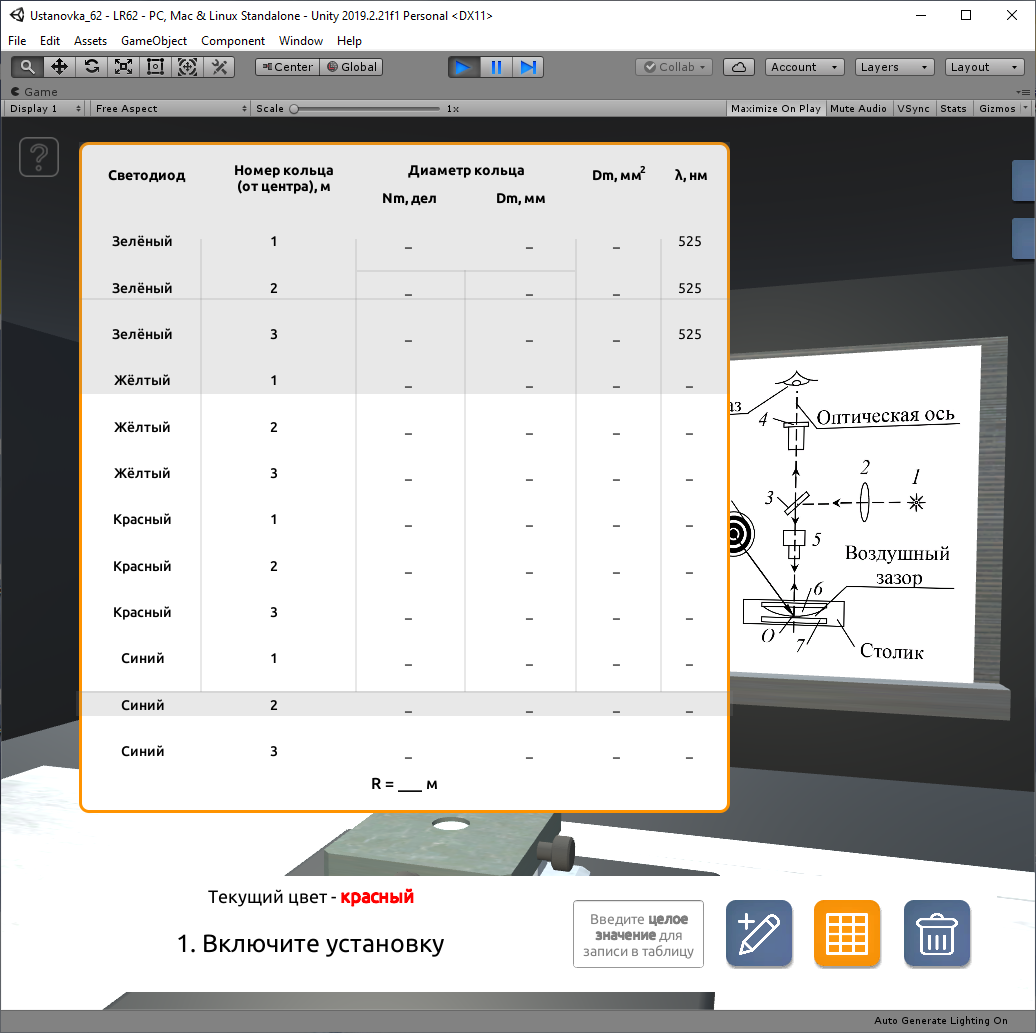
**ВЫВОД ЗНАЧЕНИЙ В ТАБЛИЦУ**

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 текстовое поле **Text** для вывода подсказок о содержании очередного этапа работы, кнопки **Button:** кнопка для записи значения в таблицу, кнопка для отображения таблицы и кнопка для очистки таблицы, а также текстовое поле **InputText** для ввода полученных значений со шкалы прибора и занесения его в таблицу (на рисунке ниже для них использованы соответствующие изображения).



.

В ходе выполнения работы на симуляторе 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» для чего для объекта Image таблицы создается скрипт с соответсвующими функциями для обработки событий, совершаемых с кнопками на панели управления работой установки «ПРАКТИКА» и информационной панели.

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

**МИНИМУМ**

**ЭЛЕМЕНТОВ ИНТЕРФЕЙСА СИМУЛЯТОРА УСТАНОВКИ**

**Обязательные объекты симулятора установки:**

1. Main Camera.
2. Directional Light и (или) дополнительное освещение.
3. Помещение (3 стены окружения: во фронт, слева, справа 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 в списке.
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:**

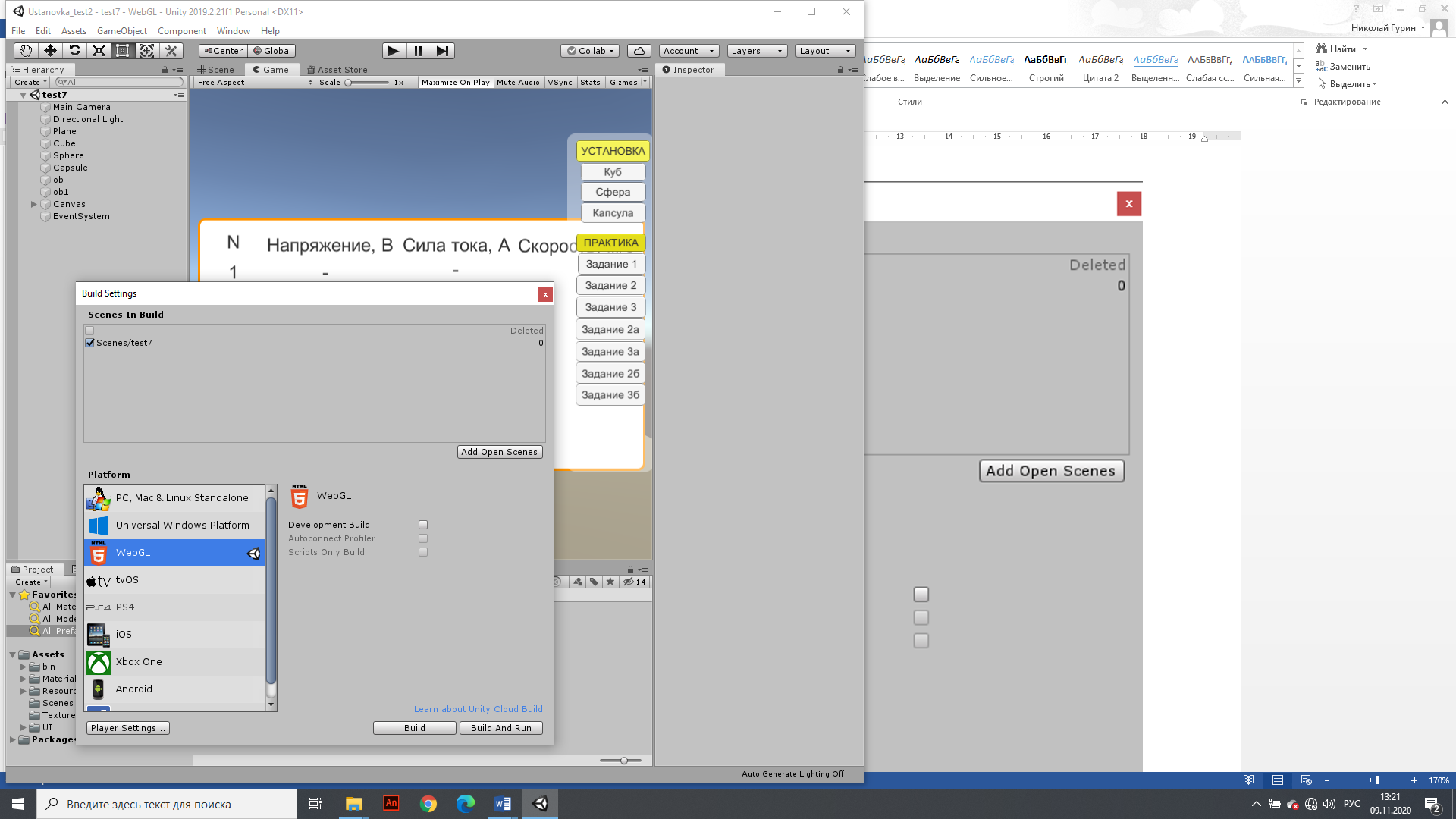
1. Вывод подсказок в текстовые окна Text на фоне изображения Image (инструкции по использованию симулятора на полный экран, назначения элементов установки с их подсветкой и очередность выполнения заданий – в информационное окно внизу экрана).
2. Перевод камеры в удобный ракурс для просмотра выбранного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

**ПРОЦЕДУРА ПУБЛИКАЦИИ ПРИЛОЖЕНИЯ UNITY**

1. Для публикации проекта в виде независимого от редактора **Unity** исполнимого **exe-файла** выполнить команду **File/Build Settings** и выбрать опцию **Universal Windows Platform**.

*Если в поле* ***Scene In Build*** *сразу не добавилась сцена, нужно нажать кнопку* ***Add Open Scenes****. При публикации обратить внимание, чтобы в путях к проекту и папки для публикации не было пробелов и русских букв. 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.*

1. Нажать кнопку **Build** и дождаться окончания процедуры построения публикации проекта (возможно потребуется несколько десятков минут). После завершения построения публикации будут созданы вспомогательные папки и файлы с используемыми данными, а также исполнимый **exe-файл** для запуска опубликованного приложения.
2. Для публикации проекта на **html-странице** необходимо, действуя аналогичным образом, в окне **Build Settings** выполнить опцию **WebGL** (можно также предварительно задать более подробные настройки публикации в **Player Settings**).



1. После завершения построения публикации будут созданы необходимые папки и файл **index.html**, на Web-странице которого будет размещен симулятор установки.

*При выполнении процедуры публикации приложения 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*.