**РАЗРАБОТКА РАБОЧЕГО ИНТЕРФЕЙСА ДЛЯ СИМУЛЯТОРА УСТАНОВКИ**

**АЛГОРИТМ РАБОТЫ УСТАНОВКИ**

Для программирования действий любой установки, прежде всего, необходимо продумать и составить **СХЕМУ ВЗАИМОДЕЙСТВИЯ** **ОБЪЕКТОВ**, отражающую взаимодействие объектов внутри этой установки, при составлении которой необходимо пройти следующие этапы:

* ***Выделить события***, которые могут происходить (включение/выключение установки, поворот или перемещение какого-нибудь объекта, нажатие кнопки и т. д.).
* ***Выделить объекты***, которые должны инициировать эти события (например, нажатие по переключателю инициирует событие по включению/выключению установки).
* ***Выбрать элементы***, которые должны реагировать на то или иное событие (например, на включение/выключение установки может реагировать большое количество элементов, начиная от отображения показаний дисплеев до запуска механизмов).
* ***Указать*,** при каких условиях должны происходить события и какие методы при этом будут вызываться.

Например, в работе симулятора лабораторной установки «НАБЛЮДЕНИЕ КОЛЕЦ НЬЮТОНА» используются следующие события:

* включение установки,
* выбор цвета света вращением барабана,
* переход к окуляру,
* заполнение таблицы результатами наблюдения,
* обработка в таблице результататов наблюдения,
* возврат к выбору нового цвета и повторение наблюдения,
* просмотр таблицы,
* очистка таблицы,
* получение итоговых результатов работы установки.

**СИСТЕМА ЗАДАНИЙ**

Первое задание при начале работы с симулятором это, как правило, задание «Включить установку». После щелчка по кнопке «ПРАКТИКА» 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 (см. рисунок), которое должно выполняться в соответствии с появляющимся текстовым сообщением и т. д. вплоть до окончания работы с симулятором.

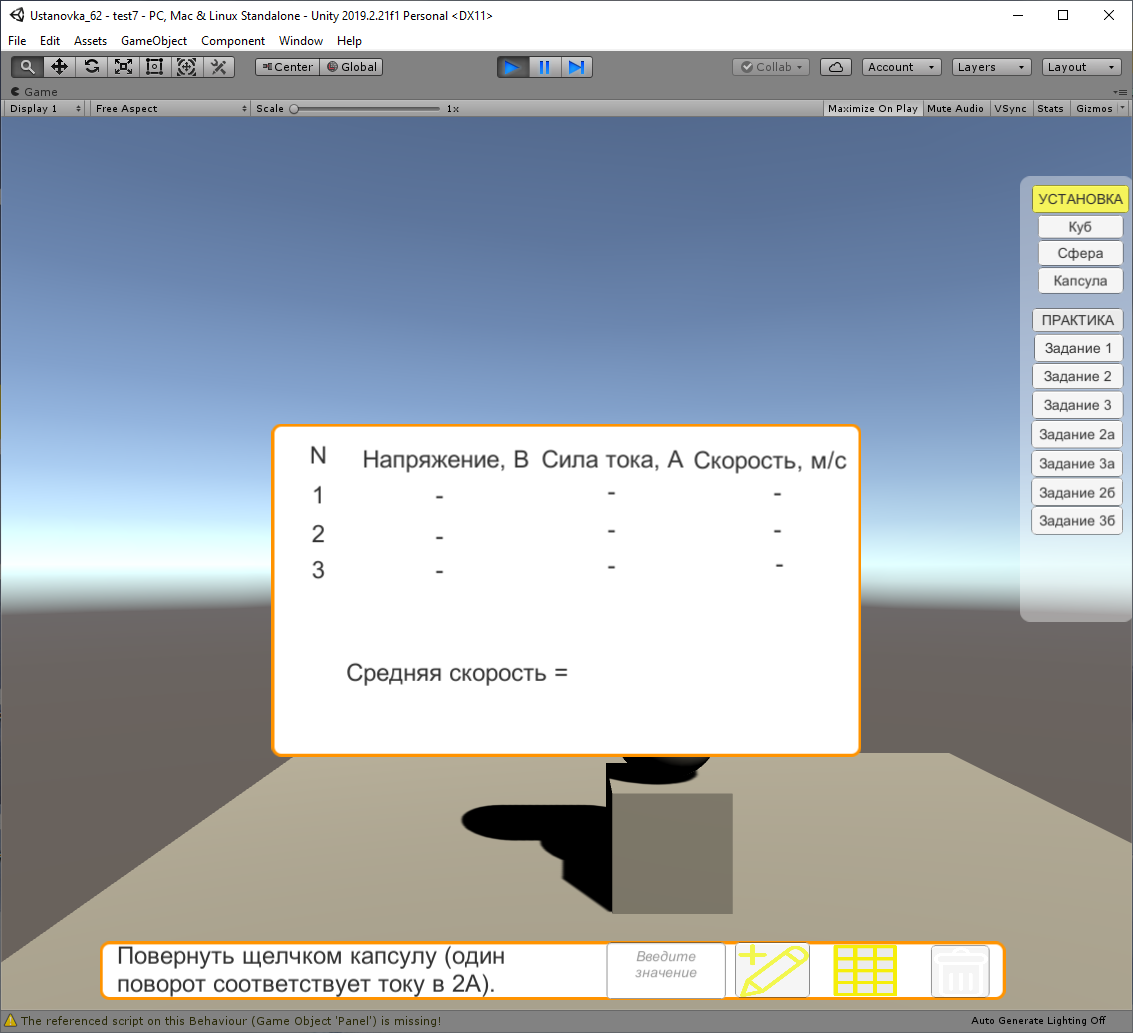


Результатом работы с симулятором установки являются полученные по ходу выполнения заданий измеренные значения, показания приборов и т. п., которые последовательно заносятся в таблицу с выполнением в ней предусмотренных в каждом конкретном эксперименте вычислений.

**СОЗДАНИЕ ПРОГРАММЫ ВЫПОЛНЕНИЯ ЭТАПОВ РАБОТЫ УСТАНОВКИ**

Для подключения функционала установки при работе с симулятором необходимо создать соответствующие **скрипты**, добавляемые на объект симулятора, с последовательным вызовом из него функций, выполняющихся при нажатии на кнопки выполнения этапов работы с симулятором.

При этом скрипт можно добавить на элемент интерфейса Table, к которому будут обращаться кнопки «ПРАКТИКИ» для выполнения заданий, соответствующих этапам работы на симуляторе, выводя соответствующие сообщения в текстовое окно в нижней части экрана левее кнопок для работы с таблицей, как показано на рисунке:



Примерный код такого скрипта приведен ниже:

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.UI;   
  
public class Tasks : MonoBehaviour  
{  
    [SerializeField]   
    Text message1;   
    public void spher()   
    {message1.text = "Включить установку, щелкнув сферу"; }  
  
    [SerializeField]   
    Text message2;  
    public void cub()   
    {message2.text =

"Повернуть щелчком куб (один поворот соответствует напряже- нию в 20В). Затем занести заданное напряжение в таблицу и перейти к Заданию 3.";

}

...

}

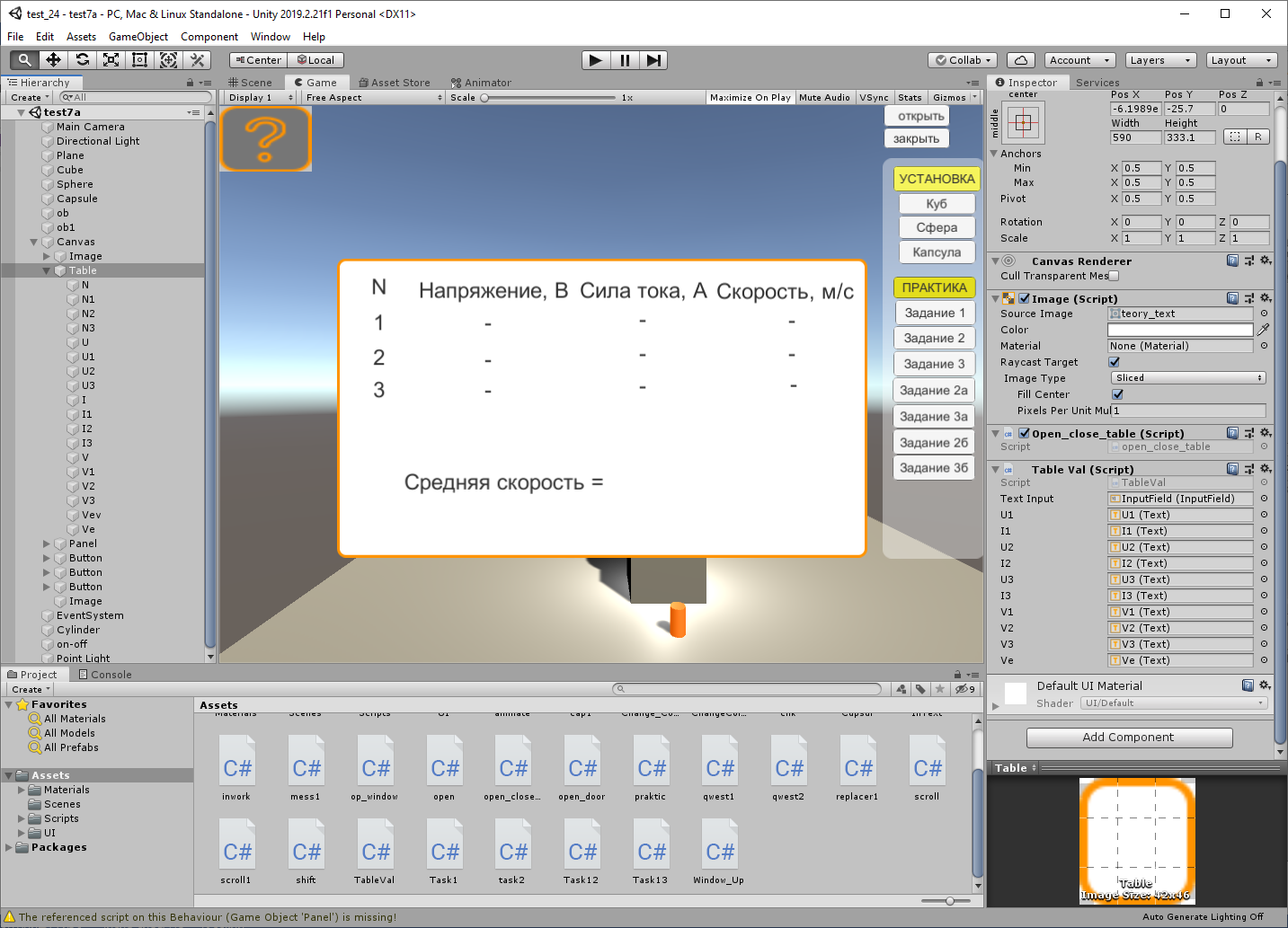
А для того, чтобы выполнить указанные в Задании действия с объектом симулятора на сцене, необходимо добавить скрипт на этот объект. При этом, если используется щелчок мышью по объекту, то необходимо в скрипте учесть правила обработки щелчков мышью с добавлением в скрипт директивы *using UnityEngine.EventSystems;* и класса *IpointerClickHandler* к базовому классу *MonoBehaviour.* Например, для поворотов куба при щелчке по нему мышью код может выглядеть следующим образом:

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.EventSystems;  
  
public class task2 : MonoBehaviour, IPointerClickHandler  
{  
             
    public void OnPointerClick(PointerEventData eventData)  
    {           
        transform.Rotate (0, 10, 0);  
    }  
      
}

**ВЫВОД ЗНАЧЕНИЙ В ТАБЛИЦУ**

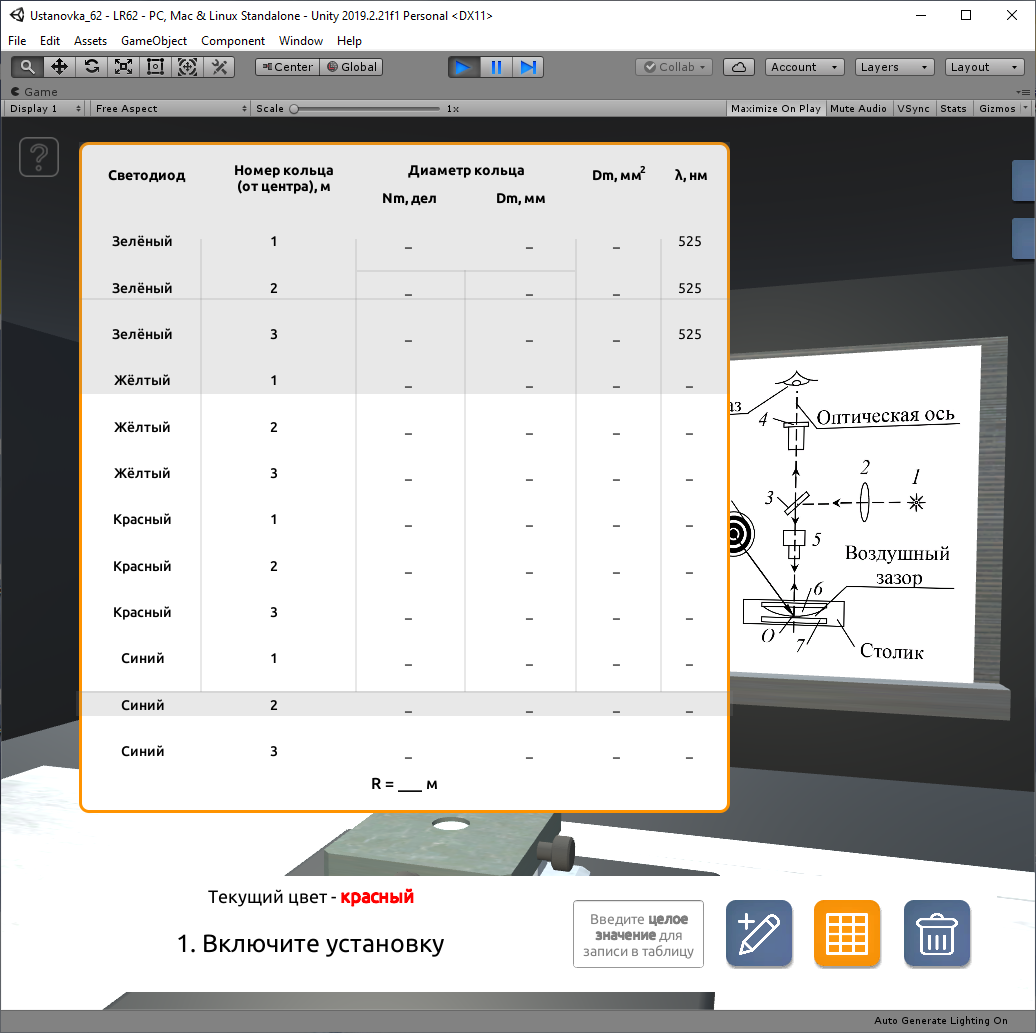
Для сохранения результатов работы с симулятором необходимо создать таблицу, в которую будут заносится результаты измерений показаний приборов установки или значений характеристик величин, получаемых при работе с установкой.

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 с именем **name1** конструкция

**name1.text.ToString()**,

а для вычислений, использующих записи в текстовых объектах, например, с именем **name1** используется конструкция

**int.Parse(name1.text).**

Ниже приведен пример кода заполнения и очистки таблицы результатов работы

**using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.UI;  
  
public class TableVal : MonoBehaviour  
{  
//определение используемых переменных**

**[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;**

**public void WriteValue() //функция для кнопки «Записать»  
    {  
        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 = "-";  
    }  
  
}**