[1. Создание динамических эффектов на html-странице: фиксация заголовка, замена текста на рисунок, увеличение размеров картинки 2](#_Toc119441744)

[2. Программа для обхода камерой вокруг установки в помещении 2](#_Toc119441745)

[3. Программа сдвига камеры с ограничениями вдоль и вглубь помещения с установкой 3](#_Toc119441746)

[4. Программа приближен. и удаления камеры относит. установки 3](#_Toc119441747)

[5. Правила создания и размещ. на экране элементов интерфейса Canvas 4](#_Toc119441748)

[6. Принцип обработки щелчка мышью по кнопке на Canvas 4](#_Toc119441749)

[7. Принцип обраб. надвиж. и ухода курсора мыши с кнопки на Canvas 5](#_Toc119441750)

[8. Процедура создания «всплывающего» текстового окна при наведении курсора на кнопку 5](#_Toc119441751)

[9. Программа «всплывающего» текстового окна при наведении курсора на кнопку 6](#_Toc119441752)

[10. Алгоритм выбора удобного ракурса для размещ. камеры на сцене 6](#_Toc119441753)

[11. Программирование выбора оптимального ракурса размещения камеры при щелчке мышью по кнопке 7](#_Toc119441754)

[12. Программирование подсветки элемента установки при наведении курсора мыши на кнопку 7](#_Toc119441755)

[13. Программа появления на экране текстового окна при щелчке по кнопке 8](#_Toc119441756)

[14. Настройка проекта для обработки щелчков мышью по 3D-объектам сцены 8](#_Toc119441757)

[15. Программа движения панели управления «показать-скрыть» при наведении-уходе курсора 9](#_Toc119441758)

[16. Программа создания массива выпадающего меню кнопок на Canvas10](#_Toc119441759)

[17. Правила создания анимации в среде Unity 10](#_Toc119441760)

[18. Код перехода к запуску анимации по нажатию клавиши 11](#_Toc119441761)

[19. Правила создания таблицы результатов эксперимента на симуляторе установки 11](#_Toc119441762)

[20. Программа занесения данных в таблицу наблюдений и проведения расчетов с данными в таблице 12](#_Toc119441763)

1. Создание динамических эффектов на html-странице: фиксация заголовка, замена текста на рисунок, увеличение размеров картинки

Фиксация заголовка:

для фиксации заголовка в html-файле мы должны поместить этот заголовок либо в тег, либо, если этот тег часто используется на странице, присвоить ему свой класс, либо прописать теге в свойстве style css настройки.

Для того чтобы сделать наш заголовок фиксированным нам нужно задать для него высоту, ширину и прописать position:fixed в файле css или в свойстве тега style.

html: <div>Фиксация заголовка</div> или <div class=”fix”> Фиксация заголовка</div> или <div style=” position:fixed; height:300px; width:100%;”> Фиксация заголовка</div>

css: div или .fix{ position:fixed; height:300px; width:100%;}

для того чтоб подключить css-файл в html-документ прописываем:

<link rel="stylesheet" href="путь\_к\_css\_файлу" />

Замена текста на картинку:

Прописываем функции срабатывающие при нажатии и уходе курсора, которые заменяют текст на изображение по id в файле с расширением .js, либо в теге script:

<script> function toim() {document.getElementById("img").innerHTML="<img src='univer.jpg' />"}

function totext() {document.getElementById( "img" ).innerHTML =" <span> БГТУ </span> " } </script>

<p>При щелчке по слову <span id="img" onmousedown="toim()" onmouseout="totext()">БГТУ</span> оно заменяется фото университета</p>

Увеличение размеров картинки:

<img src="logo.jpg" onmouseover="this.width=this.width\*5" onmouseout=" this.width = this.width/5 " />

1. Программа для обхода камерой вокруг установки в помещении

Для управления движения камеры относительно установки необходимо, прежде всего, выбрать центральную точку на установке, для чего в скрипте для управления движением камеры нужно перетянуть в добавленном скрипте в окне **Inspector** для камеры центральный объект установки в поле объектной переменной типа **Transform**, например, **targetPos** , что будет определять точку, вокруг которой будет вращаться и двигаться камера.

public class Scroll : MonoBehaviour

{ [SerializeField]

Transform targetPos;

//вращение вокруг центр точки установки с зажатой ПКМ

void Update()

{ if (Input.GetMouseButton(1))

{transform.RotateAround(targetPos.position, Vector3.up, Input.GetAxis("Mouse X") \* sensivity); }}

1. Программа сдвига камеры с ограничениями вдоль и вглубь помещения с установкой

public class Scroll : MonoBehaviour

{ float scrollSpeed = 10f;//скорость вращения и передвижения камеры в стороны относительно установки

[SerializeField]

Transform targetPos; //пределы передвижения камеры в стороны относительно установки, не заходя за стены помещения, где она находится

int sensivity = 3; int maxdistance = 6; int mindistance = 1;

//функция граничения пределов движения камеры

bool ControlDistance(float distance)

{if (distance > mindistance && distance < maxdistance) return true;

return false; }

void Update()

{ //движение камеры в стороны клавишами

float x = Input.GetAxis("Horizontal");

float y = Input.GetAxis("Vertical");

if (x != 0 || y != 0)

{Vector3 newpos = transform.position + (transform.TransformDirection(new Vector3(x, 0, 0)) + Vector3.up \* y) / sensivity;

if (ControlDistance(Vector3.Distance(newpos, targetPos.position))) transform.position = newpos; } }}

1. Программа приближения и удаления камеры относительно установки

public class Scroll : MonoBehaviour

{ float scrollSpeed = 10f;//скорость вращения и передвижения камеры в стороны относительно установки

[SerializeField]

//пределы передвижения камеры в стороны относительно установки, не заходя за стены помещения, где она находится

int sensivity = 3;

int maxdistance = 6;

int mindistance = 1;

//функция граничения пределов движения камеры

bool ControlDistance(float distance)

{if (distance > mindistance && distance < maxdistance) return true;

return false; }

void Update() {

//приближение и удаление камеы от установки прокруткой колеса мыши

if (Input.GetAxis("Mouse ScrollWheel") != 0)

{Vector3 newpos = transform.position + transform.TransformDirection(Vector3.forward \* Input.GetAxis("Mouse ScrollWheel") \* scrollSpeed);

if (ControlDistance(Vector3.Distance(newpos, targetPos.position))) transform.position = newpos; } }}

1. Правила создания и размещения на экране элементов интерфейса Canvas

**Холст** (**Canvas**) в редакторе Unity обозначает границы экрана, а размещаемые на холсте **элементы UI** появляются на этом экране в заданных точках. У всех объектов UI, размещаемых на холсте, существует **точка привязки**, отображаемая в редакторе в виде наклонного крестика с обводкой. Это инструмент для позиционирования элементов интерфейса. **Привязкой** (**Anchor**) объекта называется точка его присоединения к холсту или экрану, относительно этой точки указывается положение объекта.

При этом объект остается статичным относительно точки привязки, а сама она может перемещаться относительно холста. Точку привязки можно задать, например, как «центр экрана», и она будет оставаться в центре, когда экран меняет свой размер. Аналогично привязка к правой стороне экрана позволит объекту оставаться справа даже при изменении размеров.

Точки привязки позволяют подстраиваться не только под изменение положения, но и под изменение размера. Привязку можно отредактировать таким образом, что при увеличении размера экрана изображения будут растягиваться вместе с ним.

Способ наложения элементов **UI** друг на друга определяется порядком их следования на вкладке **Hierarchy**. Чтобы расположить всплывающее окно поверх остальных элементов **UI** необходимо перетащить окно в самый низ иерархического списка дочерних элементов холста, чтобы оно отображалось поверх всего остального.

1. Принцип обработки щелчка мышью по кнопке на Canvas

1) В сцене на основе CANVAS создается рабочий UI-объект и кнопка Button.

2) Затем к объекту добавляется сценарий, который будет вызываться при совершении событий, которые возможны для кнопок.

3) Добавить директиву using UnityEngine.UI.

4) Выбрать событие для кнопки и создать связь с UI-объектом, к которому присоединен соответствующий сценарий.

5) Выбрать для кнопки в качестве функции имя сценария для добавленного UI-объекта, и в раскрывающемся списке выбрать ту функцию, которая должна выполнить действие, когда будет взаимодействие с кнопкой.

По умолчанию у кнопки событие - OnClick() – щелчок мыши, для других событий необходимо к кнопке добавить компоненту Event Trigger и затем выбрать из списка нужное событие, например OnPointerEnter() – надвижение курсора мыши.

1. Принцип обработки надвижения и ухода курсора мыши с кнопки на Canvas

public void OnPointerEnter(PointerEventData eventData) – метод для обработки наведения курсора на кнопку

public void OnPointerExit(PointerEventData eventData) – метод для обработки ухода курсора с кнопки

1) В сцене на основе CANVAS создается рабочий UI-объект и кнопка Button.

2) Затем к объекту добавляется сценарий, который будет вызываться при совершении событий, которые возможны для кнопок.

3) Добавить директиву using UnityEngine.UI.

4) Выбрать событие OnPointerEnter() для кнопки и создать связь с UI-объектом, к которому присоединен соответствующий сценарий.

5) Выбрать для кнопки в качестве функции имя сценария для добавленного UI-объекта, и в раскрывающемся списке выбрать ту функцию, которая должна выполнить действие, когда будет взаимодействие с кнопкой.

По умолчанию у кнопки событие - OnClick() – щелчок мыши, для других событий необходимо к кнопке добавить компоненту Event Trigger и затем выбрать из списка нужное событие OnPointerEnter или OnPointerExit – наведение и увод мыши.

1. Процедура создания «всплывающего» текстового окна при наведении курсора на кнопку

Для того чтобы вывод сообщения в текстовое окно происходил по событию от пользователя необходимо создать в скрипте с именем, например PopupWindow, функцию для вывода нужного сообщения, например, OnPointerEnter(). При том объектная переменная message типа Text должна быть связана с объектом на Canvas – с соответствующим текстовым окном.

Затем в нижней части панели Inspector необходимо активизировать элемент кнопки On Enter(), щелкнуть на кнопке со значком + (плюс), добавить элемент кнопки, перетащите на ячейку объект-кнопки, выделите в меню строку с именем скрипта PopupWindow и выбрать в меню функцию OnPointerEnter() имеющуюся в скрипте PopupWindow.

using System.Collections; using System.Collections.Generic;

using UnityEngine;

public class PopupWindow : MonoBehaviour

{ public GameObject MyWindow;

void Start()

{ Close(); //ЗАКРЫТЬ ВСПЛЫВАЮЩЕЕ ОКНО ПРИ ЗАПУСКЕ ПРОГРАММЫ   }

public void OnOpenSettings()

{ Open(); //  ОТКРЫТЬ ВСПЛЫВАЮЩЕЕ ТЕКСТОВОЕ ОКНО  }

public void Open()

{ Panel.SetActive(true); //АКТИВИРОВАТЬ ОБЪЕКТ, ЧТОБЫ ОТКРЫТЬ ОКНО}

public void Close()

{Panel.SetActive(false); // ДЕАКТИВИРОВАТЬ ОБЪЕКТ, ЧТОБЫ ЗАКРЫТЬ ОКНО }}

1. Программа «всплывающего» текстового окна при наведении курсора на кнопку

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PopupWindow : MonoBehaviour

{ public GameObject MyWindow;

void Start()

{ Close(); //ЗАКРЫТЬ ВСПЛЫВАЮЩЕЕ ОКНО ПРИ ЗАПУСКЕ ПРОГРАММЫ }

public void OnOpenSettings()

{ Open(); //  ОТКРЫТЬ ВСПЛЫВАЮЩЕЕ ТЕКСТОВОЕ ОКНО  }

public void Open()

{ Panel.SetActive(true); //АКТИВИРОВАТЬ ОБЪЕКТ, ЧТОБЫ ОТКРЫТЬ ОКНО }

public void Close()

{ Panel.SetActive(false); //ДЕАКТИВИРОВАТЬ ОБЪЕКТ, ЧТОБЫ ЗАКРЫТЬ ОКНО}}

\*\*\* для вывода сообщения в текстовое окно по событию от пользователя – создать в скрипте(PopupWindow) функцию для вывода нужного сообщения, например, OnPointerEnter(). Объектная переменная должна быть связана с окном. В инспекторе необходимо активировать элемент кнопки, выбрав соответствующую функцию в скрипте.

1. Алгоритм выбора удобного ракурса для размещения камеры на сцене
2. Создать на сцене по **пустому объекту** для каждого просматриваемого из заданной точки элемента установки и переместить его в режиме сцены так, чтобы было удобно смотреть из него на один из подсвечиваемых элементов.
3. Затем выбрать этот пустой объект и в меню выполнить команду **Align With View** - пустой объект станет так, как настроен вид когда мы смотрим на объекты. Это нужно для фиксации позиции с которой должна будет смотреть камера на выбранный объект.
4. Сделать так для каждого элемента установки, и потом проинициализировать в UI-скриптах их **anchorObject** пустыми объектами для просмотра с камеры.
5. Программирование выбора оптимального ракурса размещения камеры при щелчке мышью по кнопке

Алгоритм реализации перемещения и поворота камеры к элементу установки:

1.Создать на сцене для каждого просматриваемого из заданной точки элемента установки и переместить его в режиме сцены так, чтобы было удобно смотреть из него на один из подсвечиваемых элементов.

2.Выбрать пустой объект и в голавном меню во вкладке GameObject выполнить команду Align With View - пустой объект станет так, как настроен вид того, как вы смотрите на объекты. Это нужно для того, чтобы зафиксировать позицию, с которой должна будет смотреть камера на выбранный объект.

3.Проделать то же самое для каждого элемента установки, после чего проинициализировать в UI-скриптах для соответствующих элементов их anchorObject пустыми объектами для просмотра с камеры.

void Start() {startPosition = transform.position;

startRotation = transform.rotation;

**//** *пример если позиция и поворот пустого объекта заданы значениями*

needPosition = new Vector3(5.45f, 8, -22.6f);

needRotaton = new Quaternion(0.7f, 0, 0, 0.7f);

}  void FixedUpdate(){ if(move){ offset+=speed;

transform.position = Vector3.Lerp(startPosition, needPosition, offset);

transform.rotation = Quaternion.Slerp(startRotation, needRotaton, offset);

if (offset >= 1){move = false;offset = 0;}}}

1. Программирование подсветки элемента установки при наведении курсора мыши на кнопку

Скрипт должен быть помещен на элемент установки на сцене, который нужно подсветить.

public class ChangeColor : MonoBehaviour { public void ChangCol()

{ GetComponent<Renderer> ().material.color = new Color (1, 0, 0); }

public void ChangCol1()

{ GetComponent<Renderer> ().material.color = new Color (1, 1, 1); } }

1. Программа появления на экране текстового окна при щелчке по кнопке

Для того чтобы вывод сообщения в текстовое окно происходил по событию от пользователя необходимо создать в скрипте с именем, например **Windows\_Up,** функцию для вывода нужного сообщения, например, **OnSettings().** При том объектная переменная **message** типа **Text** должна быть связана с объектом на **Canvas** – с соответствующим текстовым окном.

using UnityEngine;

using System.Collections;

public class Windows\_Up : MonoBehaviour {

[SerializeField]

Text message;

public void OnSettings() {//МЕТОД, ВЫЗЫВАЕМЫЙ ПОЛЬЗОВ. ЩЕЛЧКОМ ПО КНОПКЕ

message.text="Установка начальных значений";{}

Затем в нижней части панели **Inspector** необходимо активизировать элемент кнопки **On Click()**, щелкнуть на кнопке со значком + (плюс), добавить элемент кнопки (в листинге присутствуют как ячейка для объекта со скриптом , так и меню для вызываемой функции), перетащите на ячейку объект-кнопки, выделите в меню строку с именем скрипта **Windows\_Up** и выбрать в дополнительном меню справа функцию **OnSettings()** имеющуюся в скрипте **Windows\_Up.**

1. Настройка проекта для обработки щелчков мышью по 3D-объектам сцены

Для обработки события «щелчок мышью по объекту на сцене» необходимо:

1. добавить в иерархию объектов новый не отображаемый на сцене объект Create/UI/EventSystem

2.В сценарий для объекта необходимо добавить класс using UnityEngine.EventSystems

3.В базовый класс добавить новый интерфейс системы событий IPointerClickHandler

using UnityEngine;

using System.Collections;

using UnityEngine.EventSystems;

public class Script11 : MonoBehaviour, IPointerClickHandler {

void Start () {}

void Update () {} }

4.К камере необходимо предварительно добавить компоненту Physics RayCaster для согласования щелчков мыши по 2D-экрану со щелчками по 3D-объектам на сцене

1. Программа движения панели управления «показать-скрыть» при наведении-уходе курсора

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

public class PanelMove : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler {

RectTransform UIGameobject; // трансформация UI Панели

float width; // ширина панели

float changeX; // смещение панели

float speedPanel; // скорость закрытия панели

enum states { // перечисление состояний панели

open, close, opening, closing }

states state = states.open; // изначальное состояние закрытое

void Start() { //инициализация переменных

UIGameobject = gameObject.GetComponent<RectTransform>(); // объект панели

width = UIGameobject.sizeDelta.x; // определение ширины панели

speedPanel = 8; // скорость движения панели }

Для того чтобы панель управления с кнопками не заслоняла общую сцену с си-мулятором установки и скрывалась-открывалась на экране можно на панель до-бавить скрипт с кодом обработки событий надвижения-ухода курсора мыши с па-нели. В этом случае необходимо добавить в код скрипта два новых класса: IPointerEnterHandler, IPointerExitHandler.

В целом код управления плавным движением панели может иметь следующий вид:

void Update()

{if (state == states.closing)

{float x = UIGameobject.anchoredPosition.x;

float y = UIGameobject.anchoredPosition.y;

x += speedPanel;

changeX += speedPanel;

UIGameobject.anchoredPosition = new Vector2(x, y);

if (changeX > width)

{state = states.close;

changeX = 0; }}

if (state == states.opening)

{float x = UIGameobject.anchoredPosition.x;

float y = UIGameobject.anchoredPosition.y;

x -= speedPanel;

changeX += speedPanel;

UIGameobject.anchoredPosition = new Vector2(x, y);

if (changeX > width)

{state = states.open;

changeX = 0; }}}

public void OnPointerEnter(PointerEventData eventData)

{if (state == states.close) state = states.opening; }

public void OnPointerExit(PointerEventData eventData)

{if (state == states.open) state = states.closing;}}

1. Программа создания массива выпадающего меню кнопок на Canvas

using System.Collections; using System.Collections.Generic;

using UnityEngine; using UnityEngine.EventSystems; using UnityEngine.UI

**public class Dropdown : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler** {

public GameObject panelTrigger; // связать переменную с объектом

RectTransform triggerTransfrom; // переменная прямоуг.

public Button button; RectTransform buttonTransfrom; // кноп. и ее прямоуг.

Vector2 buttonSize; // 2мерный вектор для размещ. Кнопки

public GameObject dropdownList; // объект списка меню

RectTransform dropdownTransfrom; // прямоуг. для списка меню

Vector2 dropdownSize; // 2мерный вектор списка меню

public Sprite buttonNormalState; // для накладыв. нач. текстуры на кноп

public Sprite buttonHighlightedState; // для накладыв. конечн. текстуры на кноп.

**void Start()**{ if (dropdownList != null) {

triggerTransfrom = panelTrigger.GetComponent<RectTransform>();

buttonTransfrom = button.GetComponent<RectTransform>();

dropdownTransfrom = dropdownList.GetComponent<RectTransform>();

buttonSize = buttonTransfrom.sizeDelta;

dropdownSize = dropdownTransfrom.sizeDelta; // размер прямоуг списка

dropdownTransfrom.sizeDelta = new Vector2(0, 0);

triggerTransfrom.sizeDelta = buttonSize; // размер прямоуг. Кноп

dropdownList.SetActive(false); // активир. выпад. список}}

**public void OnPointerEnter(PointerEventData eventData) //навели курсор**

{button.GetComponent<Image>().sprite = buttonHighlightedState;

dropdownTransfrom.sizeDelta = dropdownSize;

triggerTransfrom.sizeDelta = buttonSize + dropdownSize; // общ. размер прямоуг.

**dropdownList.SetActive(true); //активир. список**}

**public void OnPointerExit(PointerEventData eventData) // убрали курсор**

{if (dropdownList != null) {

dropdownTransfrom.sizeDelta = new Vector2(0, 0);

triggerTransfrom.sizeDelta = buttonSize; button.GetComponent<Image>().sprite = buttonNormalState; **dropdownList.SetActive(false); // деактивир. список**}}}

1. Правила создания анимации в среде Unity

Анимацию можно создать, когда мы нажимаем на объект и, открыв панель Animation, нажимаем «Create». Unity предлагает сохр. анимацию с расширением .anim в папках проекта. В папках появится анимация и ее контроллер. Контроллер выглядит как граф состояний. Мы переходим к временной шкале и включаем режим авто ключевых кадров, наж красная круглая кноп. Далее переходим на нужные кадры и совершаем какие-либо манипуляции с объект. Анимацию можно проиграть кноп «Play», интерфейс временной шкалы стандартн, как и в Adobe Animate, 3ds max. Затем надо для целевого объекта задать компонент Add Component -> Animation для обычной анимации (можно задать проигрывание автоматически или циклич) или же Add Component -> Animator Controller и перетащить контроллер (контроллер позволяет руководить набором анимаций, переключ между ними, когда выполн нек условие, для работы с контроллером надо использ. код, а обычная анимация без управления может проигр по умолч). В контроллере есть разные состояния, Empty State - состояние по умолч (команда Create State)

1. Код перехода к запуску анимации по нажатию клавиши

**using UnityEngine.UI;**

**using UnityEngine.EventSystems; using UnityEngine;**

**public class ObjectAnimationController : MonoBehaviour**

{

[**SerializeField]** public GameObject myobject; //объект, у которого есть контроллер анимации

**public void Update()**

{ if(**Input.GetMouseButtonDown(0)**) StartAnimation (); }

**public void StartAnimation() //метод запуск аним, предварит нужен контроллер аним где есть boolean параметр played, если true - переходит в сост проигрывания, false не проигрыв анимац**

{if (myobject!= null){

Animator animator = myobject.GetComponent<Animator>();

if (animator != null)

{bool isPlayed = animator.GetBool("played");

animator.SetBool("played", !isPlayed); } } }

1. Правила создания таблицы результатов эксперимента на симуляторе установки

Табл состоит **из массива текстовых полей**, в поля запис р-ты работы с симул. Ячейки табл заполняются последовательно **частично вручную, частично автоматически** по формулам. Для работы с табл на информационной панели для практики должны быть предусмотрены **кнопки Button: кнопка для записи значения в таблицу, кнопка для отображения таблицы и кнопка для очистки**. Также на инф панели должно быть текстовое поле InputText для ввода полученных значений со шкалы прибора и занесения его в табл. В ходе выполнения лр измерения записываются в нужные ячейки табл по нажатию кноп «Записать», **для просмотра содержания табл использ событие наведение курсора на кнопку «Таблица», а для очистки таблицы от записей - наведение курсора на кнопку «Очистка».** Вся табл – **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».**

1. Программа занесения данных в таблицу наблюдений и проведения расчетов с данными в таблице

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).**

**using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.UI;**  
public class TableVal : MonoBehaviour  
{

**[SerializeField] ЭТО ПИШЕМ ПЕРЕД ВСЕМИ ПЕРЕМЕННЫМИ**

InputField textInput;  
Text u1, i1, u2, i2, v1, v2, ve;

//функция для кноп «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 ();}  
ve.text =((int.Parse (v1.text)+int.Parse (v2.text))/2).ToString ();}  
**textInput.text="";**}  
//функция для кноп «Очистить»

**public void Clean()**

 {u1.text = "-";  
u2.text = "-"; … и так далее все переменные}}