**ТЕМЫ ДЛЯ ОБСУЖДЕНИЯ НА КОЛЛОКВИУМЕ**

1. **ПРОКОММЕНТИРОВАТЬ КОД ВРАЩЕНИЯ КАМЕРЫ ВОКРУГ УСТАНОВКИ**

**public class Scroll : MonoBehaviour**

**{**

**[SerializeField]**

**Transform targetPos;**

**int sensivity = 3;**

**int maxdistance = 20;**

**int mindistance = 1;**

**void Update()**

**{**

**if (Input.GetMouseButton(1))**

**{**

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

**}**

**}**

Данный код отвечает за вращение игрового объекта, на котором он висит, вокруг заданной точки.

Точка, вокруг которой будет происходить вращение объявляется в поле targetPos, типа Transform.

Помимо этого, есть приватные поля, отвечающие за некоторую настройку проекта:

Sensitivity – поле, отвечающее за настройку (настройкой это назвать сложно, просто захардкоженая циферка) чувствительность.

maxDistance и minDistance (этот восхитительный нейминг…) конкретно к этому блоку кода не относятся, тем не менее, просто скажу, что это поля отвечающие за минимальную и максимальную дистанцию приближения.

1. **ПРОКОММЕНТИРОВАТЬ КОД ДВИЖЕНИЯ КАМЕРЫ В ГРАНИЦАХ ПОМЕЩЕНИЯ**

**…**

**bool ControlDistance (float distance)**

**{**

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

**return false;**

**}**

**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;**

**}**

Данный участок кода предоставляет метод проверки дистанции на то, чтобы текущий зум был в пределах конкретных рамок (minDistance, maxDistance объявленные и инициализированные выше).

После чего, максимально где и куда вставлен код, который в теории должен быть в Update, но окей. Там мы получаем вертикальный и горизонтальный ввод с клавиатуры, проверяя не равны ли они нулю. В случае же, если хотя бы одна не равна нулю, мы создаем новую позицию с помощью структуры Vector3 где просто к текущей позиции добавляем некоторую сформированный вектор направления передвижения (**transform.TransformDirection(new Vector3(x, 0, 0)) + Vector3.up \* y) / sensivity).**

Дальше просто проверяем, не выходит ли новая позиция за рамки допустимы и

в случае, если не выходит – присваиваем позиции нашего объекта новое положение

1. **ПРОКОММЕНТИРОВАТЬ КОД ПРИБЛИЖЕНИЯ И УДАЛЕНИЯ КАМЕРЫ ОТ УСТАНОВКИ**

**…**

**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;**

**}**

Представленный выше код считывает, есть ли прокрутка колеса мыши, после чего формирует вектор новой позиции, проверяет, не выходит ли он за пределы и если не выходит – присваивает новую позицию

1. **КАК СОЗДАТЬ ФАЙЛ ДЛЯ АНИМАЦИИ ОБЪЕКТА В СРЕДЕ UNITY**

Для этого, в любой папке нашего проекта, нажимаем правую кнопку мыши>create>animation

1. **КАК СОЗДАТЬ И НАСТРОИТЬ КОНТРОЛЛЕР АНИМАЦИИ ДЛЯ УПРАВЛЕНИЯ АНИМАЦИЕЙ ОБЪЕКТА В СРЕДЕ UNITY**Правая кнопка мыши>crate>animator (также можно просто добавить анимацию на игровой объект, на котором еще нет аниматора, после чего он сам создастся).

После чего, дважды нажимаем по аниматору. У нас открывается окно настройки аниматора, где можно добавить анимации, переходы, почитайте тут в моей [статье по аниматору](https://telegra.ph/Vvedenie-v-Animator-Unity-3D-07-30). Там все подробно описано, как пользоваться аниматором, переходы, смешивания, условия и т.п.

1. **ПРОКОММЕНТИРОВАТЬ СКРИПТ ДЛЯ УПРАВЛЕНИЯ АНИМАЦИЕЙ**

**Animator anim;**

**void Start()**

**{**

**anim = GetComponent<Animator>();** **}**

**void Update()**

**{**

**If(Input.GetKeyDown(KeyCode.Q))**

**{**

**anim.SetBool(“run”, true);**

**}**

**If(Input.GetKeyDown(KeyCode.W))**

**{**

**anim.SetBool(“run”, false);**

**}**

**}**

Представленный выше код выполняет задачу максимально приметивного контроля над анимациями.  
В методе Start (Вызывается при старте, перед отрисовкой первого кадра) мы получаем ссылку на компонент аниматора с игрового объекта, на котором висит данный скрипт. Далее в Update (КАЖДЫЙ КАДР, НЕ 50 РАЗ В СЕКУНДУ) идет проверка, не нажаты ли кнопки Q или W, в зависимости от которых в аниматоре изменяем состояние параметров типа bool c помощью метода setBool(string, bool), где первый параметр – имя параметра в аниматоре, второй параметр – новое значение.

1. **АЛГОРИТМ ПРОЦЕДУРЫ ПРОГРАММИРОВАНИЯ ИНТЕРАКТИВНОГО UI С ИСПОЛЬЗОВАНИЕМ КНОПОК BUTTON**
2. **Создание канваса**
3. **Создание кнопки на нем**
4. **Создание скрипта с методом, который будет вызываться по нажатию**
5. **Добавление в компонент Button новый слушатель событий**
6. **…**
7. **… (использование других событий для кнопки?)**
8. **ПРОКОММЕНТИРОВАТЬ SCRIPT ВЫВОДА СООБЩЕНИЯ В ТЕКСТОВОЕ ПОЛЕ ПОСЛЕ ЩЕЛЧКА ПО КНОПКЕ НА CANVAS**

**using System.Collections;**

**using UnityEngine;**

**using UnityEngine.UI;**

**public class InText : MonoBehaviour {   
[SerializeField]**

**Text message;**

**public void OnPointer() {**

**message.text = "Установить начальные значения";**

**}**

**}**

Данный код просто выводит заданный текст в текстовое поле при вызове метода OnPointer()

1. **АЛГОРИТМ ПРОЦЕДУРЫ СОЗДАНИЯ «ВСПЛЫВАЮЩЕГО» ТЕКСТОВОГО ОКНА ПРИ НАВЕДЕНИИ КУРСОРА НА КНОПКУ**

**1. …**

**2. …**

**3. …**

**4. …**

1. **ПРОКОММЕНТИРОВАТЬ SCRIPT ОТКРЫТИЯ-ЗАКРЫТИЯ «ВСПЛЫВАЮЩЕГО» ОКНА ПРИ НАВЕДЕНИИ-УХОДА КУРСОРА С КНОПКИ НА CANVAS**

**using UnityEngine;**

**using System.Collections;**

**public class SettingsPopup : MonoBehaviour {**

**void Start() {**    
**Close();    
    }   
  
    public void OnOpenSettings() {    
        Open();    
    }  
  
    public void Open() {  
        gameObject.SetActive(true);    
    }    
    public void Close() {  
        gameObject.SetActive(false);**

**}**

**}**

Данный компонент содержит три метода, два из которых просто меняют текущее состояние игрового объекта, на котором висит этот компонент – Open, Close.

Метод OnOpenSettings вызывает метод Open.

1. **ПРОКОММЕНТИРОВАТЬ SCRIPT ПОДСВЕТКИ ЭЛЕМЕНТА УСТАНОВКИ ПРИ НАВЕДЕНИИ КУРСОРА МЫШИ НА КНОПКУ С НАЗВАНИЕМ ЭЛЕМЕНТА В СПИСКЕ**

**using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
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); }  
}**

Компонент ChangeColor (глаголом называть класс конечно да😊) который управляет цветом материала у игрового объекта. Где метод ChangeCol – отвечает за дефолтный цвет, а ChangeCol1 – за подсветку объекта.

1. **АЛГОРИТМ ПРОЦЕДУРЫ ВЫБОРА УДОБНОГО РАКУРСА ДЛЯ КАМЕРЫ НА СЦЕНЕ**
2. **…**
3. **…**
4. **…**
5. **…**
6. **ПРОКОММЕНТИРОВАТЬ SCRIPT ПЕРЕДВИЖЕНИЯ КАМЕРЫ В ЗАДАННУЮ ТОЧКУ**

**public class Replacer : MonoBehaviour**

**{ bool move = true;**

**Vector3 startPosition;**

**Vector3 needPosition;**

**float speed = 0.01f;**

**float offset = 0;**

**Quaternion startRotation;**

**Quaternion needRotaton;**

**public void Move1()   
        {  move = true;  
            startPosition = transform.position;  
            startRotation = transform.rotation;  
            needPosition = new Vector3(277.0f, 251, 6);  
            needRotaton = Quaternion.AngleAxis(-180, new Vector3(0,1,0));**

**}  
 void Update()**

**{**

**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;**

**}**

**}**

**}**

**}**

Этот класс необходим для возврата к исходной позиции и вращению (да, начальная позиция захардкожена в методе Move1() ☹)

Пройдемся по полям:

Move – поле, отвечает за то, в движении ли сейчас объект.

startPosition и startRotation – начальная позиция и вращение игрового объекта (начальная в данном контексте – позиция, которая была на момент вызова метода Move1())

speed – некоторая скорость, с которой двигается объект.

Offset – прогресс передвижения от начальной точки к конечной.

needPosition и needRotation – позиция и вращение, в которые необходимо переместить текущий объект.