**ТЕМЫ ДЛЯ ОБСУЖДЕНИЯ НА КОЛЛОКВИУМЕ**

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

**public class Scroll : MonoBehaviour** // Определяет класс Scroll, который наследуется от базового класса MonoBehaviour. Это позволяет использовать функции Unity, такие как Update()

**{**

**[SerializeField]**

**Transform targetPos;** // Ссылка на целевую позицию (объект), вокруг которого будет вращаться камера

**int sensivity = 3;** // Чувствительность вращения камеры (больше значение – быстрее вращение)

**int maxdistance = 20;** // Максимальное расстояние от камеры до целевой позиции (пока не используется в коде)

**int mindistance = 1;** // Минимальное расстояние от камеры до целевой позиции (пока не используется в коде)

**void Update()**

**{**

**if (Input.GetMouseButton(1))** // Проверяем, удерживается ли правая кнопка мыши

**{**

// Вращаем камеру вокруг целевой позиции (targetPos) по оси Y

// Вращение зависит от движения мыши по горизонтали (Mouse X) и умножается на чувствительность (sensivity)

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

**}**

**}**

**}**

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

**…**

**bool ControlDistance (float distance)**

**{**

// Проверяет, находится ли расстояние в допустимых пределах (между mindistance и maxdistance)

**if (distance > mindistance && distance < maxdistance) return true;** // Если расстояние в пределах, возвращает true

**return false;** //иначе false

**}**

**float x = Input.GetAxis("Horizontal");** // Получает значение горизонтального ввода (стрелки влево/вправо или A/D)

**float y = Input.GetAxis("Vertical");** // Получает значение вертикального ввода (стрелки вверх/вниз или W/S)

**if (x != 0 || y != 0)** // Проверяет, есть ли ввод с клавиатуры (движение)

**{**

// Вычисляет новую позицию камеры на основе текущей позиции и ввода, учитывая чувствительность

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

// Проверяет, находится ли новая позиция в допустимых пределах от целевой позиции (targetPos)

**if (ControlDistance(Vector3.Distance(newpos, targetPos.position)))**

**{**

**transform.position = newpos;** // Если в пределах, обновляет позицию камеры

**}**

**}**

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

**…**

**if (Input.GetAxis("Mouse ScrollWheel") != 0)** // Проверяет, было ли использовано колесо прокрутки мыши

**{**

// Вычисляет новую позицию камеры, добавляя вектор движения вперед, умноженный на значение прокрутки мыши и скорость прокрутки

**Vector3 newpos = transform.position +**

**transform.TransformDirection(Vector3.forward \* Input.GetAxis("Mouse ScrollWheel") \* scrollSpeed);**

// Проверяет, находится ли новая позиция в допустимых пределах от целевой позиции (targetPos)

**if (ControlDistance(Vector3.Distance(newpos, targetPos.position)))**

**transform.position = newpos;** // Если в пределах, обновляет позицию камеры

**}**

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

* Запуск проекта в Unity
* Создать объект для анимации
* Открыть окно анимации: **Window > Animation > Animation.**
* В окне анимации выбрать объект
* Нажать кнопку **Create**
* Выбрать куда сохранить. Получится файл с расширением .anim
* Проверить, чтобы был включен режим анимации (кнопка с красной точкой)
* Изменить св-ва объекта в нужный момент времени
* Изменить св-ва и Unity автоматически создаст ключевой кадр
* Ключевой кадр можно создать самому (маленький круг рядом со свойством)
* Нажать на кнопку записи

1. **КАК СОЗДАТЬ И НАСТРОИТЬ КОНТРОЛЛЕР АНИМАЦИИ ДЛЯ УПРАВЛЕНИЯ АНИМАЦИЕЙ ОБЪЕКТА В СРЕДЕ UNITY**

* Запуск проекта
* Создание анимации или её выбор
* Правой кнопкой мыши в папке проекта и **Create > Animator Controller**
* Дать название
* Дважды щёлкнуть на созданный контроллер
* Перетащить анимационный файл из папки проекта в окно анимации
* Щелкните правой кнопкой мыши на одном из состояний анимации и выберите **Make Transition**
* Перетащите стрелку перехода к другому состоянию анимации
* Щелкните на стрелку перехода, чтобы открыть параметры в Inspector
* В разделе **Conditions** можно добавить условия для перехода
* Для этого создайте параметры в разделе **Parameters** в **Inspector Animator** (+)
* Установите условия для переходов
* Выберите объект, которому вы хотите добавить анимацию, и добавьте компонент **Animator**
* В Inspector укажите созданный **Animator** **Controller** в поле **Controller**

1. **ПРОКОММЕНТИРОВАТЬ СКРИПТ ДЛЯ УПРАВЛЕНИЯ АНИМАЦИЕЙ**

**Animator anim;** // Объявляем переменную для хранения ссылки на компонент Animator

**void Start()**

**{**

// Получаем компонент Animator, прикрепленный к текущему игровому объекту, и сохраняем его в переменной anim

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

**}**

**void Update()**

**{**

// Проверяем, нажата ли клавиша Q. Если да, то устанавливаем параметр "run" в true, что запускает анимацию бега

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

**{**

**anim.SetBool("run", true);** // Устанавливаем параметр "run" в true, вызывая анимацию бега

**}**

// Проверяем, нажата ли клавиша W. Если да, то устанавливаем параметр "run" в false, что останавливает анимацию бега

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

**{**

**anim.SetBool("run", false);** // Устанавливаем параметр "run" в false, останавливая анимацию бега

**}**

**}**

1. **АЛГОРИТМ ПРОЦЕДУРЫ ПРОГРАММИРОВАНИЯ ИНТЕРАКТИВНОГО UI С ИСПОЛЬЗОВАНИЕМ КНОПОК BUTTON**
2. **Создание UI Canvas:** Для этого щелкните правой кнопкой мыши в иерархии и выберите **UI > Canvas**
3. **Добавление кнопки:** Щелкните правой кнопкой мыши на Canvas и выберите **UI > Button**
4. **Настройки кнопки:** Выберите кнопку в иерархии и настройте её свойства в инспекторе
5. **Создание скрипта для обработки нажатий:** создать и прикрепить к кнопке
6. **Связывание кнопки с методом:** в инспекторе кнопки найти компонент Button. В разделе **On Click()** нажмите на кнопку **+,** чтобы добавить новое событие. Перетащите объект, к которому прикреплен скрипт, в поле, которое появится. В выпадающем списке выберите метод **UIButtonHandler.OnButtonClick**.
7. **Использование других событий для кнопки:** Кнопки в Unity поддерживают различные события, такие как:

* **On Pointer Enter:** событие, которое срабатывает, когда курсор мыши входит в область кнопки.
* **On Pointer Exit:** событие, которое срабатывает, когда курсор мыши покидает область кнопки.
* **On Pointer Down:** событие, которое срабатывает, когда кнопка нажата.
* **On Pointer Up:** событие, которое срабатывает, когда кнопка отпущена.

Эти события можно обрабатывать, создавая интерфейсы, такие как **IPointerEnterHandler**, **IPointerExitHandler**, **IPointerDownHandler**, **IPointerUpHandler**,и реализуя соответствующие методы в вашем скрипте.

1. **ПРОКОММЕНТИРОВАТЬ SCRIPT ВЫВОДА СООБЩЕНИЯ В ТЕКСТОВОЕ ПОЛЕ ПОСЛЕ ЩЕЛЧКА ПО КНОПКЕ НА CANVAS**

**using System.Collections;** // Подключение пространства имен для работы с коллекциями (в данном случае не используется)

**using UnityEngine;** // Подключение основного пространства имен Unity, необходимого для работы с игровыми объектами и компонентами

**using UnityEngine.UI;** // Подключение пространства имен для работы с UI-элементами, такими как текстовые поля

**public class InText : MonoBehaviour {**

**[SerializeField]**

**Text message**; // Ссылка на компонент Text, который будет использоваться для отображения сообщения. Это поле будет доступно в инспекторе Unity

**public void OnPointer() {** // Метод, который будет вызываться при взаимодействии (например, при нажатии на кнопку)

**message.text = "Установить начальные значения";** // Устанавливает текстовое сообщение в текстовом поле, когда метод вызывается

**}**

**}**

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

**1. Создать UI Canvas:** Для этого щелкните правой кнопкой мыши в иерархии и выберите **UI > Canvas**

**2. Добавление кнопки:** Щелкните правой кнопкой мыши на Canvas и выберите **UI > Button**

**3. Создать текстовое окно:** Добавьте текстовый элемент, который будет служить всплывающим окном. Щелкните правой кнопкой мыши на **Canvas** и выберите **UI > Text**. Настроить текст.

**4. Создать скрипт для управления:** создать и прикрепить к кнопке

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

**using UnityEngine;** // Подключение основного пространства имен Unity, необходимого для работы с игровыми объектами и компонентами

**using System.Collections;** // Подключение пространства имен для работы с коллекциями (в данном случае не используется)

**public class SettingsPopup : MonoBehaviour {**

**void Start() {**

**Close();** // Вызываем метод Close() при старте, чтобы скрыть всплывающее окно

**}**

**public void OnOpenSettings() {**

**Open();** // Метод, который будет вызываться для открытия окна настроек

**}**

**public void Open() {**

**gameObject.SetActive(true);** // Устанавливает активность текущего игрового объекта (всплывающего окна) в true, открывая его

**}**

**public void Close() {**

**gameObject.SetActive(false);** // Устанавливает активность текущего игрового объекта (всплывающего окна) в false, закрывая его

**}**

**}**

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

**using System.Collections;** // Импорт пространства имен для работы с коллекциями. В данном скрипте не используется

**using System.Collections.Generic;** // Импорт пространства имен для работы с обобщенными коллекциями. В данном скрипте не используется

**using UnityEngine;** // Импорт основного пространства имен Unity, необходимого для работы с игровыми объектами и компонентами

**public class ChangeColor : MonoBehaviour**

**{**

// Метод для изменения цвета объекта на красный, когда курсор наводится на кнопку

**public void ChangCol()**

**{**

// Получает компонент Renderer текущего объекта и изменяет его цвет на красный

**GetComponent<Renderer>().material.color = new Color(1, 0, 0);** // RGB: (1, 0, 0) - красный цвет

**}**

// Метод для восстановления цвета объекта на белый, когда курсор уходит с кнопки

**public void ChangCol1()**

**{**

// Получает компонент Renderer текущего объекта и изменяет его цвет на белый

**GetComponent<Renderer>().material.color = new Color(1, 1, 1);** // RGB: (1, 1, 1) - белый цвет

**}**

**}**

1. **АЛГОРИТМ ПРОЦЕДУРЫ ВЫБОРА УДОБНОГО РАКУРСА ДЛЯ КАМЕРЫ НА СЦЕНЕ**
2. **Определить цель камеры**
3. **Создать пустой объект для управления камерой:** Создайте пустой объект **(Empty GameObject)** в сцене, который будет служить родителем для камеры.

Для этого щелкните правой кнопкой мыши в иерархии и выберите **Create Empty.**

1. **Настройте положение камеры:** Поместите камеру как дочерний объект к пустому объекту. Настройте положение и вращение камеры относительно пустого объекта, чтобы получить желаемый ракурс. Вы можете использовать инструменты перемещения и вращения в **Scene View** для настройки угла обзора**.**
2. **Скрипт**
3. **ПРОКОММЕНТИРОВАТЬ SCRIPT ПЕРЕДВИЖЕНИЯ КАМЕРЫ В ЗАДАННУЮ ТОЧКУ**

**public class Replacer : MonoBehaviour**

**{ bool move = true;** // Флаг, указывающий, нужно ли перемещать камеру

**Vector3 startPosition;** // Начальная позиция камеры

**Vector3 needPosition;** // Конечная позиция, в которую нужно переместить камеру

**float speed = 0.01f;** // Скорость перемещения камеры

**float offset = 0; /**/ Смещение для интерполяции

**Quaternion startRotation;** // Начальная ориентация камеры

**Quaternion needRotaton;** // Конечная ориентация, в которую нужно переместить камеру

// Метод, вызываемый для начала перемещения камеры

**public void Move1()**

**{**

**move = true;** // Устанавливаем флаг перемещения в true, чтобы начать движение

**startPosition = transform.position;** // Сохраняем текущую позицию камеры как начальную

**startRotation = transform.rotation;** // Сохраняем текущую ориентацию камеры как начальную

**needPosition = new Vector3(277.0f, 251, 6);** // Устанавливаем целевую позицию для перемещения камеры

**needRotaton = Quaternion.AngleAxis(-180, new Vector3(0, 1, 0));** // Устанавливаем целевую ориентацию камеры с поворотом на 180 градусов по оси Y

**}**

**void Update()**

**{**

// Проверяем, нужно ли перемещать камеру

**if (move)**

**{**

**offset += speed;** // Увеличиваем смещение на заданную скорость

**transform.position = Vector3.Lerp(startPosition, needPosition, offset);** // Интерполируем позицию камеры от начальной к целевой

**transform.rotation = Quaternion.Slerp(startRotation, needRotaton, offset);** // Интерполируем ориентацию камеры от начальной к целевой

// Проверяем, достигли ли мы целевой позиции

**if (offset >= 1)**

**{**

**move = false;** // Останавливаем движение, когда достигли целевой позиции

**offset = 0;** // Сбрасываем смещение для возможных будущих перемещений

**}**

**}**

**}**

**}**