**КОНТРОЛЬНЫЕ ВОПРОСЫ по теме UNITY**

1. Использование ключевых слов “Horizontal”, “Vertical” для программирования движения 3D-объектов с клавиатуры.

Используется для организации возможности управления объектом, используя клавиатуру

**horizontalInput = Input.GetAxis("Horizontal");**

**verticalInput = Input.GetAxis("Vertical");**

1. Использование ключевых слов “Mouse X”, “Mouse Y” для программирования поворотов 3D-объектов с помощью мышью.

Используются для программирования поворотов 3d-объектов с помощью мыши

**public float sensitivity = 1f;  
rotationX += Input.GetAxis("Mouse X") \* sensitivity;**

**rotationY -= Input.GetAxis("Mouse Y") \* sensitivity;**

1. Создание и использование шаблонов Prefabs. Создание экземпляров объектов из шаблона Prefab методом Instantiate().

В Unity шаблоны Prefab используются для создания и повторного использования объектов.

**public GameObject prefab; // ссылка на шаблон Prefab**

**// Создание экземпляра объекта из шаблона Prefab**

**GameObject instance = Instantiate(prefab);**

1. Создание и использование триггеров. Назначение и использование функции OnTriggerStay().

В Unity триггеры используются для обнаружения столкновений и взаимодействия между объектами на сцене. Триггером может быть обыкновенный игровой объект на сцене, которому задано свойство IsTrigger.

При входе объекта в триггер и при нахождении в нем функция OnTriggerStay() будет вызываться каждый кадр, пока объект находится в пределах триггера (подобно Update).

**void OnTriggerStay(Collider other)**

**{**

**if (other.CompareTag("Player")) {**

**// Действия, выполняемые при взаимодействии с игроком**

**}**

**}**

1. Программа запуска движения 3D-объекта в горизонтальной плоскости X-Z

**// Создаем вектор движения, используя значения ввода и скорость**

**Vector3 movement = new Vector3(horizontalInput, 0f, verticalInput) \* speed \* Time.deltaTime;**

**// Перемещаем объект в соответствии с вектором движения**

**transform.Translate(movement);**

transform.Translate() используется для перемещения объекта в соответствии с вектором движения.

1. Программа запуска вращения 3D-объекта с помощью кватернионов Quaternion вокруг произвольной оси.

**void Update()**

**{**

**// Вычисляем угол вращения на основе времени и скорости**

**float rotationAngle = 90f \* Time.deltaTime;**

**// Создаем кватернион вращения вокруг заданной оси (задать угол и ось вращения)**

**Quaternion rotationQuaternion = Quaternion.AngleAxis(rotationAngle, Vector3.up);**

**// Применяем вращение к текущему вращению объекта**

**transform.rotation \*= rotationQuaternion;**

**}**

1. Программа генерации 3D-объекта со свойствами твердого тела и падением его на плоскость при нажатии определенной клавиши клавиатуры.

**public GameObject objectPrefab; // Шаблон объекта**

**void Update()**

**{**

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

**{**

**// Создаем экземпляр объекта из шаблона**

**GameObject obj = Instantiate(objectPrefab, transform.position, Quaternion.identity);**

**}**

**}**

1. Программа генерации объекта на сцене из префаба Prefab в случайной позиции на плоскости нажатием определенной клавиши клавиатуры.

**public GameObject objectPrefab; // Шаблон объекта**

**void Update()**

**{**

**if (Input.GetKeyDown(KeyCode.G)) {**

**// Создаем экземпляр объекта из шаблона в случайной позиции**

**Vector3 randomPosition = new Vector3(Random.Range(minX, maxX), 0f, Random.Range(minZ, maxZ));**

**GameObject obj = Instantiate(objectPrefab, randomPosition, Quaternion.identity);**

**}**

**}**

1. Программа для смены цвета 3D-объекта при щелчке по нему мышью.

**void Start()**

**{**

**private Renderer objectRenderer = GetComponent<Renderer>(); // Получаем компонент Renderer при запуске (да, не надо в жизни объявлять переменные в start и update…)**

**}**

**void OnMouseDown()**

**{**

**objectRenderer.material.color = Random.ColorHSV(); // Генерируем случайный цвет**

**}**

1. Программа вращения 3D-объекта клавишами клавиатуры.

**void Update()**

**{**

**// Вращение вокруг оси X при нажатии клавиши "W"**

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

**{**

**transform.Rotate(Vector3.right, 90f \* Time.deltaTime);**

**}**

**}**

1. Программа вращения 3D-объекта с помощью мыши.

**void Update()**

**{**

**float mouseX = Input.GetAxis("Mouse X"); // Получаем изменение положения мыши по осям X и Y**

**float mouseY = Input.GetAxis("Mouse Y");**

**transform.Rotate(Vector3.up, mouseX \* 90f, Space.World);**

**transform.Rotate(Vector3.left, mouseY \* 90f, Space.Self);**

**}**

1. Программа для обработки столкновения двух 3D-объектов с изменением их цвета.

**private Renderer objectRenderer; // Ссылка на компонент Renderer объекта**

**void Start()**

**{**

**objectRenderer = GetComponent<Renderer>(); // Получаем компонент Renderer при запуске**

**}**

**void OnCollisionEnter(Collision collision)**

**{**

**objectRenderer.material.color = Color.red; // Меняем цвет объекта при столкновении**

**}**

1. Программа для обработки входа и выхода 3D-объекта в триггер с изменением цвета объекта.

**void Start()**

**{**

**private Renderer objectRenderer = GetComponent<Renderer>(); // Получаем компонент Renderer**

**}**

**void OnTriggerEnter(Collider other)**

**{**

**if (other.CompareTag("Player")) // Проверяем тег объекта, который вошел в триггер**

**{**

**objectRenderer.material.color = Color.green; // Меняем цвет объекта при входе в триггер**

**}**

**}**

**void OnTriggerExit(Collider other)**

**{**

**if (other.CompareTag("Player")) // Проверяем тег объекта, который вышел из триггера**

**{**

**objectRenderer.material.color = Color.red; // Меняем цвет объекта при выходе из триггера**

**}**

**}**

1. Программа для запуска вращения одного 3D-объекта при попадании другого 3D-объекта в триггер.

**public GameObject objectToRotate; // Объект, который будет вращаться**

**void OnTriggerEnter(Collider other)**

**{**

**if (other.CompareTag("Player")) // Проверяем тег объекта, который попал в триггер {**

**// Запускаем вращение объекта ToRotate вокруг оси Y**

**objectToRotate.transform.Rotate(Vector3.up, 90f \* Time.deltaTime);**

**}**

**}**

1. Программа для озвучивания движения объекта по сцене.

**public AudioSource audioSource; // Ссылка на компонент Audio Source**

**public AudioClip movementSound; // Звуковой клип для движения**

**void Start()**

**{**

**audioSource = GetComponent<AudioSource>(); // Получаем компонент Audio Source при запуске**

**}**

**void Update()**

**{**

**// Пример движения объекта по сцене (может быть заменен на ваше собственное движение)**

**transform.Translate(Vector3.forward \* Time.deltaTime);**

**// Проверяем, если объект движется, то проигрываем звуковой клип**

**if (transform.position != Vector3.zero) {**

**if (!audioSource.isPlaying) {**

**audioSource.clip = movementSound;**

**audioSource.Play();**

**}**

**}**

**else**

**{**

**audioSource.Stop(); // Останавливаем воспроизведение звука, если объект остановлен**

**}**

**}**

1. Программа для озвучивания столкновения 3D-объектов при попадании одного из объектов в другой.

**public AudioSource audioSource; // Ссылка на компонент Audio Source**

**public AudioClip collisionSound; // Звуковой клип для столкновения**

**void Start()**

**{**

**audioSource = GetComponent<AudioSource>(); // Получаем компонент Audio Source при запуске**

**}**

**void OnCollisionEnter(Collision collision)**

**{**

**// Проверяем, если столкновение происходит с другим объектом**

**if (collision.gameObject.CompareTag("OtherObject"))**

**{**

**// Проигрываем звуковой клип столкновения**

**audioSource.PlayOneShot(collisionSound);**

**}**

**}**