**КОНТРОЛЬНЫЕ ВОПРОСЫ по теме UNITY**

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

**using UnityEngine;**

**// Этот скрипт позволяет изменить цвет 3D-объекта при щелчке мышью.**

**public class ChangeColorOnClick : MonoBehaviour**

**{**

**// Вызывается, когда происходит клик по объекту.**

**void OnMouseDown()**

**{**

**// Получаем компонент Renderer для изменения цвета объекта.**

**Renderer renderer = GetComponent<Renderer>();**

**// Устанавливаем цвет объекта в красный (можете выбрать другой цвет по вашему выбору).**

**renderer.material.color = Color.red;**

**}**

**}**

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

**Using UnityEngine;**

**// Этот скрипт обрабатывает столкновения двух 3D-объектов и изменяет их цвет и текстуру.**

**public class ChangeColorAndTextureOnCollision : MonoBehaviour**

**{**

**// Переменная для новой текстуры, которую вы должны задать в инспекторе Unity.**

**public Texture newTexture;**

**// Вызывается при столкновении объекта с другим объектом.**

**void OnCollisionEnter(Collision collision)**

**{**

**// Проверяем, имеет ли столкнувшийся объект тег "OtherObject".**

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

**{**

**// Получаем компонент Renderer для изменения цвета и текстуры объекта.**

**Renderer renderer = GetComponent<Renderer>();**

**// Устанавливаем цвет объекта в синий (вы можете выбрать другой цвет по вашему выбору).**

**renderer.material.color = Color.blue;**

**// Устанавливаем новую текстуру объекта (предполагается, что newTexture была предварительно назначена).**

**renderer.material.mainTexture = newTexture;**

**}**

**}**

**}**

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

**using UnityEngine;**

**// Этот скрипт обрабатывает вход и выход объекта в триггер и изменяет его цвет.**

**public class ChangeColorOnTrigger : MonoBehaviour**

**{**

**// Вызывается, когда объект входит в триггер.**

**void OnTriggerEnter(Collider other)**

**{**

**// Проверяем, имеет ли объект, входящий в триггер, тег "TriggerZone".**

**if (other.CompareTag("TriggerZone"))**

**{**

**// Получаем компонент Renderer для изменения цвета объекта.**

**Renderer renderer = GetComponent<Renderer>();**

**// Устанавливаем цвет объекта в зеленый (вы можете выбрать другой цвет по вашему выбору).**

**renderer.material.color = Color.green;**

**}**

**}**

**// Вызывается, когда объект выходит из триггера.**

**void OnTriggerExit(Collider other)**

**{**

**// Проверяем, имеет ли объект, выходящий из триггера, тег "TriggerZone".**

**if (other.CompareTag("TriggerZone"))**

**{**

**// Получаем компонент Renderer для изменения цвета объекта.**

**Renderer renderer = GetComponent<Renderer>();**

**// Устанавливаем цвет объекта в белый (или другой цвет по вашему выбору) при выходе из триггера.**

**renderer.material.color = Color.white;**

**}**

**}**

**}**

1. Программа для запуска движения 3D-объекта при попадании другого 3D-объекта в заданный триггер.

**using UnityEngine;**

**// Этот скрипт запускает движение объекта при попадании другого объекта в триггер.**

**public class LaunchOnTrigger : MonoBehaviour**

**{**

**// Скорость движения объекта.**

**public float speed = 5.0f;**

**// Вызывается, когда объект попадает в триггер.**

**void OnTriggerEnter(Collider other)**

**{**

**// Проверяем, имеет ли объект, попавший в триггер, тег "TriggerZone".**

**if (other.CompareTag("TriggerZone"))**

**{**

**// Выполняем движение объекта вперед с учетом скорости и времени.**

**// Vector3.forward представляет направление движения вперед.**

**// Умножаем на speed и Time.deltaTime, чтобы учесть скорость и фреймы.**

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

**}**

**}**

**}**

1. Программа управления движением танка по сцене в горизонтальной плоскости.

**using UnityEngine;**

**// Этот скрипт управляет движением танка по горизонтальной плоскости.**

**public class TankMovement : MonoBehaviour**

**{**

**// Скорость движения танка.**

**public float movementSpeed = 5.0f;**

**// Вызывается каждый кадр.**

**void Update()**

**{**

**// Получаем значение горизонтального ввода от клавиатуры (A и D или стрелки влево и вправо).**

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

**// Создаем вектор движения по горизонтали.**

**Vector3 movement = new Vector3(horizontalInput, 0, 0) \* movementSpeed \* Time.deltaTime;**

**// Применяем движение к трансформации танка.**

**transform.Translate(movement);**

**}**

**}**

1. Программа управления вращением башни танка.

**using UnityEngine;**

**// Этот скрипт управляет вращением башни танка.**

**public class TankTurretRotation : MonoBehaviour**

**{**

**// Скорость вращения башни.**

**public float rotationSpeed = 5.0f;**

**// Вызывается каждый кадр.**

**void Update()**

**{**

**// Получаем значение горизонтального ввода от клавиатуры (A и D или стрелки влево и вправо).**

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

**// Создаем вектор вращения вокруг оси Y (вертикальной оси) с учетом ввода и скорости.**

**Vector3 rotation = Vector3.up \* horizontalInput \* rotationSpeed \* Time.deltaTime;**

**// Применяем вращение к башне танка.**

**transform.Rotate(rotation);**

**}**

**}**

1. Программа управления вращением ствола ствола танка с ограничениями по углу поворота.

**using UnityEngine;**

**// Этот скрипт управляет вращением ствола танка с ограничениями по углу поворота.**

**public class TankTurretRotationWithLimits : MonoBehaviour**

**{**

**// Максимальный угол поворота ствола.**

**public float maxRotationAngle = 45.0f;**

**// Вызывается каждый кадр.**

**void Update()**

**{**

**// Получаем значение вертикального ввода от клавиатуры (W и S или стрелки вверх и вниз).**

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

**// Вычисляем новый угол поворота ствола с учетом ввода.**

**float newRotation = Mathf.Clamp(transform.localEulerAngles.x + verticalInput, -maxRotationAngle, maxRotationAngle);**

**// Устанавливаем новый угол поворота ствола.**

**transform.localEulerAngles = new Vector3(newRotation, 0, 0);**

**}**

**}**

1. Программа реализации выстрела танка с использованием префаба снаряда.

**using UnityEngine;**

**// Этот скрипт реализует выстрел танка с использованием префаба снаряда.**

**public class TankShooting : MonoBehaviour**

**{**

**// Префаб снаряда для выстрела.**

**public GameObject projectilePrefab;**

**// Позиция, откуда будут выпускаться снаряды.**

**public Transform firePoint;**

**// Вызывается каждый кадр.**

**void Update()**

**{**

**// Проверяем, была ли нажата клавиша пробела (или другая клавиша по вашему выбору).**

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

**{**

**// Создаем новый экземпляр снаряда из префаба на позиции firePoint с его текущей ориентацией.**

**Instantiate(projectilePrefab, firePoint.position, firePoint.rotation);**

**}**

**}**

**}**

1. Программа префаба снаряда для обработки полета и попадания снаряда в цель.

**using UnityEngine;**

**// Этот скрипт обрабатывает движение и попадание снаряда в цель.**

**public class Projectile : MonoBehaviour**

**{**

**// Скорость снаряда.**

**public float speed = 10.0f;**

**// Время жизни снаряда.**

**public float lifetime = 2.0f;**

**// Вызывается при создании снаряда.**

**private void Start()**

**{**

**// Уничтожаем снаряд через указанное время lifetime.**

**Destroy(gameObject, lifetime);**

**}**

**// Вызывается каждый кадр.**

**private void Update()**

**{**

**// Двигаем снаряд вперед вдоль своей ориентации с учетом скорости и времени.**

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

**}**

**// Вызывается при столкновении с другим коллайдером.**

**private void OnTriggerEnter(Collider other)**

**{**

**// Проверяем, имеет ли объект, с которым столкнулся снаряд, тег "Target".**

**if (other.CompareTag("Target"))**

**{**

**// Уничтожаем объект-мишень.**

**Destroy(other.gameObject);**

**// Уничтожаем сам снаряд.**

**Destroy(gameObject);**

**}**

**}**

**}**

1. Программа озвучивания событий движения танка.

**using UnityEngine;**

**// Этот скрипт озвучивает события движения танка.**

**public class TankAudio : MonoBehaviour**

**{**

**// Компонент AudioSource для воспроизведения звука.**

**private AudioSource audioSource;**

**// Вызывается при старте объекта.**

**private void Start()**

**{**

**// Получаем компонент AudioSource из объекта.**

**audioSource = GetComponent<AudioSource>();**

**}**

**// Вызывается каждый кадр.**

**private void Update()**

**{**

**// Проверяем, находится ли танк в движении (по позиции).**

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

**{**

**// Если звук не воспроизводится, воспроизводим его.**

**if (!audioSource.isPlaying)**

**{**

**audioSource.Play();**

**}**

**}**

**else**

**{**

**// Если танк не двигается (находится в нулевой позиции), останавливаем воспроизведение звука.**

**audioSource.Stop();**

**}**

**}**

**}**

1. Программа озвучивания события выстрела из ствола танка.

**using UnityEngine;**

**// Этот скрипт озвучивает событие выстрела из ствола танка.**

**public class TankShootingAudio : MonoBehaviour**

**{**

**// Компонент AudioSource для воспроизведения звука.**

**private AudioSource audioSource;**

**// Звуковой эффект выстрела.**

**public AudioClip shotSound;**

**// Вызывается при старте объекта.**

**private void Start()**

**{**

**// Получаем компонент AudioSource из объекта.**

**audioSource = GetComponent<AudioSource>();**

**}**

**// Вызывается каждый кадр.**

**private void Update()**

**{**

**// Проверяем, была ли нажата клавиша пробела (или другая клавиша по вашему выбору).**

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

**{**

**// Воспроизводим звук выстрела однократно с помощью PlayOneShot.**

**audioSource.PlayOneShot(shotSound);**

**}**

**}**

**}**

1. Программа озвучивания события попадания снаряда в цель.

**using UnityEngine;**

**// Этот скрипт озвучивает событие попадания снаряда в цель.**

**public class TargetHitAudio : MonoBehaviour**

**{**

**// Компонент AudioSource для воспроизведения звука.**

**public AudioSource audioSource;**

**// Звуковой эффект попадания снаряда в цель.**

**public AudioClip collisionSound;**

**// Вызывается при старте объекта.**

**void Start()**

**{**

**// Получаем компонент AudioSource из объекта.**

**audioSource = GetComponent<AudioSource>();**

**}**

**// Вызывается при столкновении снаряда с объектом.**

**void OnCollisionEnter(Collision collision)**

**{**

**// Проверяем, имеет ли объект, с которым столкнулся снаряд, тег "Bullet".**

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

**{**

**// Воспроизводим звук попадания однократно с помощью PlayOneShot.**

**audioSource.PlayOneShot(collisionSound);**

**}**

**}**

**}**

1. Программа поворота башни танка-бота в сторону танка-игрока при попадании в триггер танка-игрока.

**// В методе OnTriggerEnter танка-бота**

**if (other.CompareTag("PlayerTank"))**

**{**

**// Вычисляем направление к танку-игроку от танка-бота.**

**Vector3 directionToPlayer = other.transform.position - transform.position;**

**// Создаем кватернион, который ориентирован в направлении к танку-игроку.**

**Quaternion lookRotation = Quaternion.LookRotation(directionToPlayer);**

**// Плавно поворачиваем башню танка-бота в направлении танка-игрока.**

**// Quaternion.Slerp позволяет плавно поворачивать от текущей ориентации к новой (lookRotation).**

**// Time.deltaTime \* rotationSpeed определяет скорость поворота.**

**turretTransform.rotation = Quaternion.Slerp(turretTransform.rotation, lookRotation, Time.deltaTime \* rotationSpeed);**

**}**

1. Программа поворота и движения танка-бота в сторону танка-игрока при достижении заданной дистанции.

**using UnityEngine;**

**public class TankBotMovement : MonoBehaviour**

**{**

**public Transform playerTank; // Ссылка на трансформ танка-игрока.**

**public float detectionDistance = 10.0f; // Дистанция, на которой танк-бот обнаруживает танк-игрока.**

**public float movementSpeed = 5.0f; // Скорость движения танка-бота.**

**public float rotationSpeed = 2.0f; // Скорость поворота танка-бота.**

**private Rigidbody rb;**

**private void Start()**

**{**

**rb = GetComponent<Rigidbody>();**

**}**

**private void Update()**

**{**

**// Проверяем расстояние между танком-ботом и танком-игроком.**

**float distanceToPlayer = Vector3.Distance(transform.position, playerTank.position);**

**// Если расстояние меньше заданной дистанции, то начинаем движение к танку-игроку.**

**if (distanceToPlayer <= detectionDistance)**

**{**

**// Ориентируем танк-бота в сторону танка-игрока.**

**Vector3 directionToPlayer = playerTank.position - transform.position;**

**Quaternion lookRotation = Quaternion.LookRotation(directionToPlayer);**

**rb.rotation = Quaternion.Slerp(rb.rotation, lookRotation, rotationSpeed \* Time.deltaTime);**

**// Двигаем танк-бота вперед в направлении танка-игрока.**

**Vector3 moveDirection = transform.forward \* movementSpeed;**

**rb.velocity = moveDirection;**

**}**

**else**

**{**

**// Если расстояние больше заданной дистанции, останавливаем движение.**

**rb.velocity = Vector3.zero;**

**}**

**}**

**}**

1. Программа обработки попадания «луча» от башни танка-бота в танк-игрока с запуском выстрела.

**using UnityEngine;**

**public class TankBotShooting : MonoBehaviour**

**{**

**public Transform turretTransform; // Ссылка на трансформ башни танка-бота.**

**public Transform playerTank; // Ссылка на трансформ танка-игрока.**

**public GameObject projectilePrefab; // Префаб снаряда.**

**public Transform firePoint; // Позиция, откуда будут выпускаться снаряды.**

**public float fireRate = 2.0f; // Скорость стрельбы в выстрелах в секунду.**

**private float nextFireTime; // Время следующего выстрела.**

**private void Update()**

**{**

**// Проверяем, прошло ли достаточно времени для следующего выстрела.**

**if (Time.time >= nextFireTime)**

**{**

**// Создаем луч от башни танка-бота к танку-игроку.**

**Ray ray = new Ray(firePoint.position, playerTank.position - firePoint.position);**

**RaycastHit hit;**

**// Проверяем, попал ли луч в танк-игрока.**

**if (Physics.Raycast(ray, out hit))**

**{**

**if (hit.collider.CompareTag("PlayerTank"))**

**{**

**// Запускаем выстрел.**

**Shoot();**

**}**

**}**

**}**

**}**

**private void Shoot()**

**{**

**// Создаем снаряд из префаба на позиции firePoint с его текущей ориентацией.**

**Instantiate(projectilePrefab, firePoint.position, firePoint.rotation);**

**// Устанавливаем время следующего выстрела с учетом fireRate.**

**nextFireTime = Time.time + 1.0f / fireRate;**

**}**

**}**

1. Программный код короутины для запуска снаряда танка-бота с задержкой по времени

**using UnityEngine;**

**using System.Collections;**

**public class TankBot : MonoBehaviour**

**{**

**public Transform tower;**

**public Transform muzzle;**

**public GameObject bullet;**

**public float rotationSpeedMuzzle = 10.0f;**

**public float rotationSpeedTank = 5.0f;**

**public float moveSpeed = 5.0f;**

**public int life = 3;**

**private bool canShoot = true;**

**private void OnTriggerStay(Collider other)**

**{**

**if (other.tag == "player")**

**{**

**Vector3 relativePos = other.transform.position - transform.position;**

**Quaternion newrot = Quaternion.LookRotation(relativePos) \* Quaternion.AngleAxis(0, Vector3.down);**

**tower.rotation = Quaternion.Slerp(tower.rotation, newrot, Time.deltaTime \* rotationSpeedMuzzle);**

**RaycastHit hit;**

**if (Physics.Raycast(muzzle.position, muzzle.TransformDirection(new Vector3(0, 0, 1)), out hit))**

**{**

**if ((hit.transform.tag == "player") && canShoot)**

**StartCoroutine(BotShoot());**

**}**

**float distance = Vector3.Distance(other.transform.position, transform.position);**

**if (distance > 10 && distance < 20)**

**{**

**transform.rotation = Quaternion.Slerp(transform.rotation, newrot, Time.deltaTime \* rotationSpeedTank);**

**transform.position = Vector3.Lerp(transform.position, other.transform.position, Time.deltaTime \* moveSpeed);**

**}**

**}**

**}**

**IEnumerator BotShoot()**

**{**

**canShoot = false;**

**Vector3 MuzzleForward = muzzle.transform.position + muzzle.transform.TransformDirection(Vector3.zero);**

**Vector3 muzzleRot = muzzle.rotation.eulerAngles;**

**muzzleRot = new Vector3(muzzleRot.x + 90, muzzleRot.y, muzzleRot.z);**

**Instantiate(bullet, MuzzleForward, Quaternion.Euler(muzzleRot));**

**yield return new WaitForSeconds(3f);**

**canShoot = true;**

**}**

**private void OnCollisionEnter(Collision collision)**

**{**

**if (collision.gameObject.tag == "core")**

**{**

**life--;**

**if (life < 1) Destroy(gameObject);**

**}**

**}**

**}**