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

public class NewBehaviourScript : MonoBehaviour, IPointerClickHandler

{

float red, green, blue;

public void OnPointerClick(PointerEventData pointerEvent)

{

red = Random.Range(0f, 1.0f);

green = Random.Range(0f, 1.0f);

blue = Random.Range(0f, 1.0f);

Color col1 = new Color(red, green, blue);

gameObject.GetComponent<Renderer>().material.color = col1;

}

}

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

private void OnCollisionEnter(Collision collision)

{

collision.gameObject.GetComponent<Renderer>().material.color = new Color(Random.value, Random.value, Random.value);

collision.gameObject.GetComponent<Renderer>().material.mainTexture = texture;

}

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

void Start()

{

objectRenderer = GetComponent<Renderer>();

}

void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Player")) // Замените "Player" на тег вашего объекта

{

objectRenderer.material.color = Color.red; // Измените цвет на красный

}

}

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

public GameObject fan; --обЪект который будет двигаться после попадания другого в триггер

public void OnTriggerStay(Collider other)

{

if (other.name == "robot") –если имя второго объекта робот. Можно без этой проверки тогда триггер будет реагировать на все объекты попавшие в него

{

fan.transform.position += new Vector3(0, 0, -fanSpeed \* Time.deltaTime); -изменять положение по оси Z с заданной скоростью

fan.transform.Rotate(0, 0, rotateSpeed \* Time.deltaTime); -вращение, не обязательно

}

}

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

**void Update () {**

**float z = Input.GetAxis("Vertical");** *// клавиши W и S* **transform.Translate(z,0,0);** *// движение вперед-назад*

**float x = Input.GetAxis("Horizontal");** *// клавиши A и D*  
**transform.Rotate(0f, x, 0f);** *//**повороты танка*

**}**

6. Программа управления вращением башни танка

Transform bash;

private void Start()

{

bash = gameObject.transform.Find("Tower");

}

Private void Update()  
{  
 float mouseX = Input.GetAxis("Mouse X") \* towerRotateSpeed;

bash.Rotate(0, 0, mouseX);  
}

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

public float minAngle = -30f; // Минимальный угол поворота

public float maxAngle = 30f; // Максимальный угол поворота

void Update()

{

float v = Input.GetAxis("Mouse Y");

float rotationAmount = v \* rotationSpeed \* Time.deltaTime;

// Получаем текущий угол

float currentAngle = transform.eulerAngles.z;

// Приводим угол к диапазону -180 до 180

if (currentAngle > 180)

currentAngle -= 360;

// Рассчитываем новый угол

float newAngle = currentAngle + rotationAmount;

// Ограничиваем угол поворота

newAngle = Mathf.Clamp(newAngle, minAngle, maxAngle);

// Применяем вращение

transform.rotation = Quaternion.Euler(0, 0, newAngle);

}

8. Программа реализации выстрела танка с использованием префаба снаряда

public GameObject core;

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Space))

{

Vector3 spawnPosition = transform.position -

transform.right \* 4.5f

//- transform.up \* 6.1f

//+ transform.forward \* 1.0f

; –выставляем позицию спавна снаряда на конец дула

GameObject newcore = Instantiate(core, spawnPosition, transform.rotation); -создание из префаба который мы поместили в переменную core

}

}

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

public GameObject exp1; -помещаем взрыу эффект

void Update()

{

transform.position -= transform.TransformDirection(Vector3.right \* coreSpeed);

} –полет снаряда

private void OnCollisionEnter(Collision col)

{

if (col.gameObject.tag == "goal")

{

GetComponent<Renderer>().enabled = false; --делаем снаряд невидимым

Instantiate(exp1, gameObject.transform); --запускаем эффект взрыва

}

}

10. Программа озвучивания событий движения танка.

AudioSource zvtank;

Private void Update()  
{

if ((Input.GetAxis("Horizontal") != 0 || Input.GetAxis("Vertical") != 0) && !isPlaying)

{ zvtank.Play(); isPlaying = true; }

if (Input.GetAxis("Horizontal") == 0 && Input.GetAxis("Vertical") == 0 && isPlaying)

{ zvtank.Stop(); isPlaying = false; }

}

11. Программа озвучивания события выстрела из ствола танка.

void Update()

{

if (Input.GetKeyDown(KeyCode.Space))

{

Vector3 spawnPosition = transform.position

- transform.right \* 4.5f

//- transform.up \* 6.1f

//+ transform.forward \* 1.0f

;

GameObject newcore = Instantiate(core, spawnPosition, transform.rotation);

gameObject.GetComponent<AudioSource>().PlayOneShot(gameObject.GetComponent<AudioSource>().clip);

}

}

То же самое что и 8, просто тут мы запускаем звук заложенный в объект (дуло)

12. Программа озвучивания события попадания снаряда в цель.

private void OnCollisionEnter(Collision col)

{

if (col.gameObject.tag == "goal")

{

GetComponent<Renderer>().enabled = false;

Instantiate(exp1, gameObject.transform);

col.gameObject.GetComponent<AudioSource>().PlayOneShot(col.gameObject.GetComponent<AudioSource>().clip);

}

}

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

using UnityEngine;

public class TankTurretRotation : MonoBehaviour

{

public Transform playerTank; // Ссылка на танк-игрока

public float rotationSpeed = 5f; // Скорость поворота

private bool isPlayerInRange = false; // Флаг, указывающий, находится ли игрок в триггере

void Update()

{

if (isPlayerInRange)

{

// Поворачиваем башню к танку-игроку

Vector3 direction = (playerTank.position - transform.position).normalized;

Quaternion lookRotation = Quaternion.LookRotation(direction);

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

}

}

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("PlayerTank"))

{

isPlayerInRange = true; // Игрок вошел в триггер

}

}

private void OnTriggerExit(Collider other)

{

if (other.CompareTag("PlayerTank"))

{

isPlayerInRange = false; // Игрок вышел из триггера

}

}

}

14. 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 танк-игрока

public float moveSpeed = 5f;

public float rotationSpeed = 5f;

public float detectionDistance = 10f;

void Update()

{

// Проверяем расстояние до игрока

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

if (distance < detectionDistance)

{

// Поворачиваем танк к игроку

Vector3 direction = (playerTank.position - transform.position).normalized;

Quaternion lookRotation = Quaternion.LookRotation(direction);

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

// Двигаемся к игроку

transform.position += transform.forward \* moveSpeed \* Time.deltaTime;

}

}

}

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

using UnityEngine;

public class TankBotShooting : MonoBehaviour

{

public Transform playerTank; // Ссылка на танк-игрока

public GameObject bulletPrefab; // Префаб снаряда

public Transform firePoint; // Точка, откуда будет производиться выстрел

public float shootDistance = 15f;

public float fireRate = 1f;

private float nextFireTime = 0f;

void Update()

{

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

if (distance < shootDistance && Time.time >= nextFireTime)

{

Shoot();

nextFireTime = Time.time + 1f / fireRate;

}

}

void Shoot()

{

Instantiate(bulletPrefab, firePoint.position, firePoint.rotation);

}

}

16. Программа короутины для запуска снаряда танка-бота с задержкой по времени,

using UnityEngine;

public class TankBotCoroutineShooting : MonoBehaviour

{

public Transform playerTank; // Ссылка на танк-игрока

public GameObject bulletPrefab; // Префаб снаряда

public Transform firePoint; // Точка, откуда будет производиться выстрел

public float shootDistance = 15f;

public float fireRate = 1f;

private void Start()

{

StartCoroutine(ShootCoroutine());

}

private IEnumerator ShootCoroutine()

{

while (true)

{

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

if (distance < shootDistance)

{

Shoot();

yield return new WaitForSeconds(1f / fireRate);

}

yield return null; // Ждем следующий кадр

}

}

void Shoot()

{

Instantiate(bulletPrefab, firePoint.position, firePoint.rotation);

}

}