**Тема 10. УПРАВЛЕНИЕ 3D-МОДЕЛЬЮ ТАНКА В СРЕДЕ UNITY (4 часа)**

1. В созданном ранее проекте Unity3D с внедренной моделью танка *\*.3DS* в окружающую среду *Terrain* задать в *Инспекторе компонент* для компоненты *Rigidbody* танка задать ему массу *(Mass)* около 1000.
2. Раскрыть иерархию вложенных в танк объектов в окне *Hierarchy* проекта и проверить правильность вложений: в танк должен быть вложен корпус, а затем башня, а в башню вложен ствол. Кроме того, при программном управлении движением 3D-объектов необходимо учесть разную направленность осей координат в Unity (вертикальная ось – Y, а в горизонтальной плоскости – оси X-Z) и 3ds MAX (вертикальная ось – Z, а в горизонтальной плоскости – оси X-Y), что, как правило, приводит к необходимости использовать требуемое направление (Vector3.forward, Vector3.right).
3. Создать файл сценария, выполнив в инспекторе ресурсов *Assets* команду **Create/C# Script** и дать файлу соответствующее имя *(например, mytank)*.
4. Добавить скрипт как компонент в модель танка (то есть на верхний объект иерархии танка, в нашем случае это его корпус) – перетянуть файл из окна *Assets* на объект танка в окне *Hierarchy* (или нажать кнопку *Add Component* и найти в списке созданный начальный скрипт). В инспекторе ресурсов *Assets*, кликнув на значке скрипта два раза, открыть файл в *Visual Studio*.
5. Задать в области объявлений класса **MonoBehaviour:** объектные переменные башни **bashnya** и ствола **stvol** через **Transform**, а также переменные для скорости **float TankMoveSpeed** **= 0.1f;** и поворота танка **float RotateSpeed** **= 1f**;. Также необходимо добавить **float**переменные для максимального и минимального угла поворота ствола с соответствующими значениями.
6. Добавить в функцию инициализации **void Start()** значения для вложенных объектов башни **bashnya = gameObject.transform.Find("bashnya");** и ствола **stvol = bashnya.transform.Find("stvol");**
7. Добавить в функцию выполнения **void Update()** движения танка вперед-назадоператоры проверки условия ввода **float x = Input.GetAxis("Horizontal"); // кнопки A D** и **float z = Input.GetAxis("Vertical"); // кнопки W S**, с обработкой условия **if (z != 0)** через вычисление смещения **transform.position += transform.TransformDirection(Vector3.right\*TankMoveSpeed\*z);**
8. Добавить в функцию выполнения **void Update()** вращение танка вокруг вертикальной оси влево-вправо операторы проверки условия нажатия соответствующих клавиш с обработкой условия через метод *transform.Rotate.*
9. Добавить в функцию выполнения вращения башни вокруг вертикальной оси влево-вправо за мышью через **Input.GetAxis("Mouse X");** с обработкой условия через метод вращения *transform. Rotate*с применением скорости **bashnya.Rotate(0f, h\*RotateSpeed, 0f);.**
10. Добавить в функцию выполнения вращение ствола по оси Z вверх-вниз. Сначала необходимо задать поворот на требуемый угол в градусах **stvol.transform.Rotate(0f, 0f, v\*RotateSpeed);.** Затем утановить ограничения угла в в заданных пределах поворота ствола через метод локального вращения с ограничением **float new\_angle = Mathf.Clamp(stvol.transform.localEulerAngles.z, min, max);** и его применением **stvol.localEulerAngles = new Vector3(0f, 0f, new\_angle);.**
11. Перейти к режиму **Game** и включить переход к полному экрану игры – нажать кнопку **Maximize on Play**, затем запустить игру – нажать кнопку **Play** над сценой и проверить работу созданного сценария управления танком. Для реалистичного передвижения необходимо отрегулировать скорость и массу танка.
12. В окне иерархии **Hierarchy** добавить новый объект – снаряд **core** в форме сферы **Sphere** соответствующих размеров для ствола танка. Сделать дочерним снаряд стволу танка, наложив его перетаскиванием мышью в окне иерархии **Hierarchy** на ствол.
13. Добавить к ядру **core** «физику» командой **Add Component/Physics/Rigidbody**. При необходимости добавить ядру массу в инспекторе в свойствах **Rigidbody**, а также поставить галочку **isKinematic**, чтобы снаряд мог воздействовать на объекты на сцене, а на него воздействие могло оказываться только из кода. Убрать галочку **Use Gravity**.
14. Открыть компонент **Script** танка и добавить в его программный код переменные: типа **Transform** для управления перемещением снаряда *core*, булевскую для выстрела (**bool shoot**).
15. В функцию **Start()** добавить строку для инициализации значения для переменной снаряда.
16. В функцию **Update()** добавить обработку условий нажатий определенной клавиши на клавиатуре (например, пробел) или на клик мыши для выполнения выстрела. В данном условии сделать снаряд дочерним элементом ствола **core.transform.parent = stvol.transform;**, т.к. после выстрела снаряд будет отвязан от ствола, чтобы во время полета он не повторял поворот ствола. Также нужно повернуть снаряд так, как повернут ствол **core.transform.localRotation = stvol. transform.localRotation;**. Затем поместить снаряд в ствол со смещением вперед **core.transform.position = stvol. transform.position + stvol. transform.TransformDirection(Vector3.up\*4f)**, где 4f – длина ствола (необходимо выставить нужную величину в своответвии с вашей моделью танка). В последнюю очередь укажите через **shoot=true**, что можно произвести выстрел.
17. В функцию **Update()** добавить обработку полета снаряда **if (shoot)** c отвязкой снаряда от ствола **core.transform.parent = null;** и его перемещением **core.transform.position += core.transform.TransformDirection(Vector3.left);**.
18. Запустить игру и проконтролировать соответствие полета снарядов из разных точек сцены, различных поворотах танка и его башни.
19. Добавить на сцену 3 различных цели-объекта (например, куб, сфера и цилиндр), наложить на них материалы и текстуры, задать им соответствующие размеры и расположить в разных местах сцены на ее поверхности. Добавить объектам-целям «физику» командой **Add Component/Physics/Rigidbody**. Добавить тэг **goal** к объектам-целям, по которым будут производиться выстрелы.
20. В созданном ранее проекте Unity3D с управлением движением танка и выстрелами снарядами добавить обработку попадания в цель, которая происходит при добавлении хотя бы на один объект компонента **Physics/Rigidbody** и на оба объекта коллайдера.
21. Добавим новый скрипт, например, с именем **trigger** на снаряд. В программном коде нового скрипта вместо пустых функций **Start()** и **Update()** определить новую функцию обработки попадания одного объекта в коллайдер другого, например, **void** **OnCollisionEnter(Collision col)**, где переменная **col** содержит ссылку на коллайдер другого объекта, с которым произошло столкновение.
22. Задать в функции **OnCollisionEnter** проверяемое условие столкновения, например, **col.gameObject.tag == "goal"**, где **goal** – тэг объектов, над которыми нужно производить действия (все объекты-цели).
23. Задать значение какого-либо свойства для цели, которое должно поменяться после столкновения, например, цели становятся окрашенными в разные оттенки красного: **Color color = new Color(Random.Range(0f,1f), 0,0)**, где **Random.Range** возвращает случайное значение в заданном диапазоне от 0 до 1 с плавающей точкой.
24. При попадании присваиваем объектам с тегом **goal** заданный цвет **col.gameObject.GetComponent<Renderer>().material.color = color;** и скрываем снаряд **GetComponent<Renderer>().enabled = false;.** При этом в скрипте **mytank.cs** в условии для выполнения выстрела добавите отображение снаряда **core.GetComponent<Renderer>().enabled = true;**
25. Добавьте возврат цвета к белому через 3 секунды. Для этого необходимо создать корутину: **IEnumerator wait3seconds (GameObject obj)**. В теле корутины написать **yield return new WaitForSeconds(3f)**, где 3 – это время ожидания в секундах. За этой строчкой добавить **obj.gameObject.GetComponent<Renderer>().material.color = new Color(1,1,1);** – окрашиваем переданный в метод объект в белый цвет. Осталось добавить вызов корутины из из условия столкновения в методе *OnCollisionEnter*: **StartCoroutine(wait3seconds(col.gameObject));**
26. На источники звука (цели, снаряд, танк) поместить компонент **AudioSourse**. В инспекторе на всех компонентах отключить галочку **Play On Awake**, чтобы звук не проигрывался сразу при загрузке сцены. И поставить галочку **Loop** на танке, чтобы звук проигрывался циклически. Далее нужно добавить аудио клипы на соответствующие источники (кроме целей, на них клипы будут добавлены программно). Слушатель по умолчанию уже находится на камере.
27. Теперь нужно лишь проиграть данные аудио клипы на соответствующих источниках при определенных событиях в игре. События: танк поехал, танк остановился, выстрел, смена цвета цели при попадании, смена цвета цели спустя 3 секунды после попадания.
28. Для этого для танка: объявить переменные для источника **AudioSource source\_tank;** и переменную, которая хранит состояние запущен ли звук танка (изначально не запущен) **bool isPlaying = false;** в метод **void Start ()** добавить инициализацию **source\_tank = GetComponent<AudioSource>();**
29. Затем добавьте условие если танк двигается и звук не включён **if ((x!=0 || z!=0) && !isPlaying),** в котором включите проигрывание источника на танке **source\_tank.Play();** и отметьте, что звук включен **isPlaying = true;**
30. Создайте ещё условие на проверку остановлен ли танк и играет ли звук **if (x == 0 && z == 0 && isPlaying),** в котором выключитепроигрывание звука на танке **source\_tank.Stop();** и отметьте, что звук выключен **isPlaying = false;**
31. Для добавления звука выстрела снаряда необходимо вусловии выстрела добавить запуск звука выстрела на ядре, взяв компонент источника, например, без использования переменной **core.gameObject.GetComponent<AudioSource>().PlayOneShot(core.gameObject.GetComponent<AudioSource>().clip);**
32. Для добавления звука смены цвета (в скрипте **trigger.cs**) нужно указать, что это поле можно заполнять из инспектора с помощью атрибута **[SerializeField]** и на следующей строке объявить переменную для звука попадания в цель **AudioClip clipIN;**. Аналогично проделать для звука смены цвета спустя 3 секунды (переменная **clipOUT**).
33. В функции **OnCollisionEnter** в условии столкновения добавить получение ссылки на источник и подставить звуковое сопровождение при столкновении с целью **col.gameObject.GetComponent<AudioSource>().PlayOneShot(clipIN);**
34. Также в корутину, когда цвет через 3 секунды возвращается к белому добавить произведение запуска звука для объекта, ссылку на который передали в корутину, и подставляем звук возврата цвета **obj.gameObject.GetComponent<AudioSource>().PlayOneShot(clipOUT);**
35. Создайте панель управления состоящую из 2-ух слайдеров, задающих скорости танка и снаряда соответственно (связать значения слайдеров с переменными скоростей танка и снаряда), с подписями: «скорость танка:» и «скорость снаряда:», а также кнопки «скрыть» и «показать», которые соответственно скрывают и разворачивают панель управления на экране.
36. Добавить в программный код танка функции, обрабатывающие соответствующим образом щелчки по кнопкам «скрыть» и «показать».