**Тема 6. Выстрелы танка снарядами, обработка попадания их в цель и озвучивание сцены**

1. В созданном ранее проекте **Unity3D** с управлением движения танка **Тема 12** создайте новую сцену на основе существующей, дав ей новое имя, например, «**выстрелы**» и добавьте на сцену новый объект – снаряд с именем, например, **core** на основе примитива типа **Sphere** или **Capsule** соответствующих размеров для ствола танка.
2. Добавьте к снаряду **core** «физику» командой **Add Component/Physics/Rigidbody**. При необходимости добавьте ядру массу в инспекторе в свойствах **Rigidbody**, а также поставьте галочку **is Kinematic**, чтобы *снаряд мог воздействовать на объекты на сцене, а на него воздействие могло оказываться только из кода*. Наложите на снаряд соответствующую текстуру или цвет.
3. Создать **prefab** снаряда, действуя аналогично **Теме 9** – перетянуть его мышью со сцены в папку **Assets** и удалить его со сцены. *В дальнейшем для префаба снаряда добавим еще скрипт его полета и взаимодействия с целями на сцене при его попадании*.
4. Добавить для **ствола** танка новый скрипт, в котором, прежде всего, объявить публичную объектную переменную типа **GameObject** для ссылки на префаб cнаряда и в скрипте ствола в **Инспекторе** выбрать объект в списке **Assets** для связи его с объявленной переменной, нажав на «кружочек» справа от имени объектной переменной.
5. В метод **Update()** добавьте условие для проверки нажатия определенной клавиши на клавиатуре, которая не задействована для управления танком (например, пробел Space) для выполнения выстрела.
6. В условии нажатия клавиши необходимо записать код для двух действий:

* определить вектор **Vector3**  точки, в которой нужно разместить экземпляр снаряда относительно точки привязки ствола откуда он должен вылетать – нужно добавить к этой точке смещение вперед на длину ствола в соответствии с вашей моделью танка (иначе он будет вылетать из точки привязки объекта ствола танка – там, где он присоединен к башне);
* сгенерировать экземпляр снаряда из префаба в этой точке методом **Instantiate().**

1. Добавьте новый скрипт на **префаб** снаряда в **Assets** для управления поведением снаряда, выполняющий следующие действия:

* определите переменную типа **float** для задания скорости снаряда;
* в методе **Update()** добавьте постоянное перемещение снаряда вперед с заданной скоростью.

1. Запустите игру и проконтролируйте соответствие полета снарядов из разных точек сцены, различных поворотах танка и его башни, их попадания в объекты-цели с автоматической обработкой столкновений коллайдеров снаряда и целей с их отталкиванием просто как твердых тел.
2. После выстрелов можно заметить, что снаряды постоянно находятся на сцене и не удаляются, даже если отлетели уже на большое расстояние и по-прежнему остаются в памяти, что плохо для производительности компьютера (особенно, если вы будете стрелять «очередями»). Чтобы исправить данный факт, в код снаряда в методе **Start()** добавьте метод **Destroy()**,который будет уничтожать снаряд через заданное количество секунд.
3. Для дополнительной обработки (кроме их простого «расталкивания») столкновений снаряда с объектами на сценедобавить в скрипт префаба снаряда код на основе функции **OnCollisionEnter(),** при этом задать для всех объектов-целей один и тот же маркер-тег, например, **goal**, для чего в **Инспекторе** под именем каждого объекта-цели в строке **Tag** вести выбранное имя тега. *При этом все объекты будут обнаруживаться по имени тега, а не по их именам name, которые у всех объектов обязаны быть разными. Это позволит все такие объекты обрабатывать одним и тем же программным кодом.*
4. Добавьте в код скрипта префаба в функцию **OnCollisionEnter()** скрытие снаряда после столкновения с объектом с использованием для компоненты визуализации объекта **GetComponent<Renderer>** свойства **enabled** со значением **false**.
5. Добавьте в код скрипта префаба снаряда публичную объектную переменную, например, **exp1** для обращения к имеющемся в папке **Assets** проекта в разделе **Standard Assets/ParticleSystems/Prefabs** различным префабам с эффектами, например, к эффекту взрыва **Explosion**. После задания имени для публичной объектной переменной свяжите ее в **Инспекторе** в скрипте префаба снаряда с этим объектом в списке **Assets**, аналогично тому, как это делалось в п. 4 инструкции.
6. Добав ьте также в код скрипта префаба снаряда в функцию **OnCollisionEnter()** в обработку условия попадания снаряда в цель генерацию эффекта взрыва **exp1** в точке столкновения с использованием метода **Instantiate()**.
7. Для добавления звуковых эффектов на сцену нужно, прежде всего, найти и перенести в папку **Assets** звуковые файлы, которые соответствуют звукам движения танка, выстрела, попадания в цель, а также фоновому звуку проигрывания сцены
8. В инспекторе на все объекты-источники звука: танк, ствол, цели и камеру добавить компонент **Audio/AudioSourse**, в котором связать с объектной переменной **AudioClip** соответствующий звуковой файл (нажать «кружочек» справа и выбрать в **Assets** звуковой файл). При этом:

* на всех компонентах объектов-источников звука отключить галочку **Play On Awake,** чтобы звук не проигрывался сразу при загрузке сцены, кроме компоненты **AudioSourse** укамеры для проигрывания фонового звука;
* поставить галочку **Loop** в компонентах **AudioSourse** танка и камеры, чтобы звук проигрывался циклически, т. к. танк может долго непрерывно перемещаться по сцене;
* установить для каждого звука соответствующую громкость в параметре **Volume**;
* на всех компонентах объектов-источников звука установить режим **3D-**звука для всех звуков, кроме фонового в параметре **Spatial Blend**;
* изменить при необходимости минимальную и максимальную дистанцию для прослушивания каждого из звуков в режиме 3D Sound Settings.

1. Для проигрывания фонового звука не требуется создание программного кода – он будет проигрываться автоматически сразу после запуске сцены. При этом прослушивателем всех звуков как и источником фонового звука будет камера **Main Camera**, в которую компонент **AudioListener** добавлен по умолчанию.
2. Запустите приложение и проверьте, что фоновый звук проигрывается.
3. Для проигрывания других звуков необходимо добавить соответствующий программный код в скрипты: для звука движения танка в **скрипт танка**, для звука выстрела в **скрипт ствола**, для звука разрыва в **скрипт префаба снаряда**.
4. Для проигрывания звука движения танка необходимо в его скрипт внести следующие добавления:

* объявить объектную переменную для источника - **AudioSource zvtank;**
* объявить булевскую переменную, которая хранит состояние запущен ли звук танка (изначально не запущен) - **bool isPlaying = false;**
* в метод **Start ()** добавить инициализацию переменной **zvtank**

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

* в метод **Update ()** добавить проверку через условный оператор **if** - если танк двигается и звук не включён, то включить проигрывание источника звука на танке и отметить через булевскую переменную, что звук включен:

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

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

* добавить также в метод **Update ()** условие на проверку остановлен ли танк и играет ли звук, при выполнении которого выключитьпроигрывание звука на танкеи отметить, что звук выключен:

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

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

1. Запустить приложение и проверить проигрывание и остановку звука при движении и остановке танка.
2. Для запуска звука выстрела необходимо добавить в скрипт ствола в условие проверки нажатия клавиши клавиатуры для запуска снаряда строку для запуска звука, связанного со стволом:

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

1. Для запуска звука разрыва снаряда при попадании его в цель необходимо добавить в скрипт префаба снаряда в методе **OnCollisionEnter(Collision col)** в условие проверки столкновения снаряда с целью аналогичную строку, но с учетом объекта столкновения для запуска звука, связанного уже с целью:

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