**Тема 17. Создание ботов и обработка их взаимодействия с объектами сцены**

1. Создать префаб для будущих танков-ботов противника из уже имеющегося в проекте танка, созданного на предыдущих занятиях (не убирая со сцены самого танка).
2. Добавить к танку на сцене (не к префабу) достаточно большой (радиусом ~30) сферический триггерный коллайдер, для обнаружения в дальнейшем при попадании в него других танков-ботов танка-игрока (не забудьте включить режим **Is Trigger** !).

*При создании кода используйте материалы* ***Лекции 16*** *с приведенными в них пояснениями*

1. Создать предварительно в папке **Assets** новый скрипт с именем, например, **«Bot»** для использования в дальнейшем танком-ботом со следующим программным кодом:
   * объявить переменные типа **float** переменные для задания движений танка-бота – скорость передвижения, скорость поворота и скорость поворота башни;
   * публичные объектные переменные типа **Transform** для управления его башнейи стволом и переменную типа **GameObject** для ссылки на префаб снаряда;
   * булеву переменную **bool** для определения, может ли в данный момент танк произвести выстрелс начальным значением **true**.
2. Добавить в скрипт бота метод **OnTriggerStay(Collider other)**, т.к. выполнять какие-либо действия он должен при нахождении в большом триггере игрока. Внутри этого метода определить условие, что танк-бот находится именно в триггере игрока:

**if (other.tag=="Player")**

1. При выполнении условия, что танк-бот находится в триггере игрока, необходимо вычислить:
   * дистанцию от бота до игрока, используя метод **Distance()**:

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

* + определить 3D-вектор направления от бота к игроку:

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

* + определить по вектору направления кватернион 3D-поворота башни и ствола (здесь второй кватернион доворачивает вокруг вертикальной оси на угол в 90 градусов для согласования осей в **Unity** и **3ds MAX**):

**Quaternion newRot= Quaternion.LookRotation(relativePos)\*Quaternion.AngleAxis(-90,Vector3.up );**

1. При попадании бота в сферический коллайдер танка-игрока медленно повернуть башню бота в направлении танка-игрока с заданной функцией **Slerp()** скоростью (здесь: **bash** – переменная, связанная с башней бота, **rotspeedbsh** – скорость поворота башни):

**bash.rotation = Quaternion.Slerp(bash.rotation, newrot, Time.deltaTime \*rotspeedbash);**

1. Определить переменную типа **Raycast**, определяющую попадание «луча» в объект.
2. Задать условие проверки попадания выпущенного из башни бота «луча» в объект на сцене (здесь: **Physics.Raycast()** – метод бросания «луча» от башни бота в направлении 3D-вектора **Vector3.right** (вперед)**; hit** – определенная ранеепеременная типа **Raycast)**:

**if (Physics.Raycast(bash.transform.position, bash.transform.TransformDirection(Vector3.right), out hit)) ;**

1. При выполнении условия попадания «луча» в объект задать еще условие попадания «луча» в заданный объект – танк-игрока, а также проверки условия на разрешение выстрела булевой переменной **canshoot:**

**if((hit.transform.tag=="Player")&&canshoot)**

1. При выполнении всех этих условий запустить короутину **botshoot()**, которая запускает выстрелы ботом в танк-игрок каждые 3 секунды:

**IEnumerator botshoot()  
{canshoot = false;  
Vector3 forwardofstvol= stvol.transform.position + stvol.transform.TransformDirection(Vector3.up \* 4f);  
GameObject newcore = Instantiate(core, forwardofstvol, stvol.rotation);  
yield return new WaitForSeconds(3f);  
canshoot = true;}**

1. При дистанции меньше 20 единиц (это расстояние задается специально меньше радиуса сферического коллайдера танка-игрока, поэтому бот должен будет вначале поворачивать башню и стрелять, а уже затем будучи внутри сферического коллайдера сам поворачиваться и одновременно двигаться навстречу танку-игроку), бот должен двигаться и поворачиваться в направлении к игроку (здесь: **rotSeedTank** – скорость поворота танка-бота, определяемая функцией **Slerp()**; **moveSpeed** – скорость движения танка-бота, определяемая функцией **Lerp()** ):

**if (distance < 20)**

**{transform.rotation = Quaternion.Slerp(transform.rotation, newRot, Time.deltaTime \* rotSeedTank);  
 transform.position = Vector3.Lerp(transform.position, other.transform.position, Time.deltaTime \* moveSpeed); }**

1. Проведите следующую процедуру добавления скрипта **«Bot»** на префаб бота-танка (будьте внимательны и строго соблюдайте для избежания «глюков» очередность перечисленных ниже действий):
   * разместите экземпляр префаба танка-бота на сцене;
   * «перетяните» в **Assets** скрипт **Bot** на префаб танка;
   * добавьте на экземпляр префаба тег **«goal»;**
   * свяжите объектные переменные для башни, ствола и снаряда в скрипте **Bot** танка-бота с соответсвующими объектами на сцене (выбирайте башню и ствол из списка правильно, поскольку названия этих объектов у танка-игрока и танка-бота совпадают и нужно подобрать правильное соответсвие) и нажмите кнопку **Apply;**
   * удалите у танка-бота на сцене камеру;
   * удалите у танка-бота на сцене все старые скрипты для управления исходным танком (оставшиеся от танка-игрока) и его стволом (для запуска снаряда танком-игроком), оставив на танке-боте только новый скрипт **«Bot»**.
2. Запустите приложение и проверьте реакцию бота на приближение к нему танка-игрока.
3. Добавьте возможность уничтожения вражеского танка, добавив в скрипт Bot переменную типа float для задания количества «жизней» у танка-бота при попадании в него снарядами танка-игрока, например**, life=3**, а также создайте в нём метод **OnCollisionEnter (Collision col),** который удаляет танк-бот со сцены после трех попаданий в него снарядов игрока (здесь **core** – тег снаряда):

**if (gameObject.tag=="core")  
            { if (--life < 1) Destroy(gameObject);  }**

1. Разместите на сцене в разных ее краях (на расстоянии большем размера сферического коллайдера танка-игрока, чтобы обстрел танка-игрока не начался сразу) еще несколько танков-ботов, повторив для него все этапы подготовки, перечисленные в п. 10.
2. Запустите приложение и разыграйте «танковое сражение» до уничтожения всех танков, кроме танка-игрока, который будет по заданным программным кодом условиям игры «бессмертным».
3. (\*, факультативно). Создать в **3ds MAX**, импортировать в проект и использовать в качестве префаба для ботов танк другой конструкции с другими текстурами, используя для него тот же уже разработанный скрипт «**Bot**» и действуя аналогично описанному выше, но уже без ограничений, связанных с использованием исходного танка-игрока для создания префаба танка-бота, перечисленные в п. 12.