**ПРОГРАММИРОВАНИЕ**

**АВТОНОМНЫХ ОБЪЕКТОВ**

**(БОТов)**

**БОТ - автономная копия объекта, созданная на основе префаба, которая управляется программно без участия пользователя.**

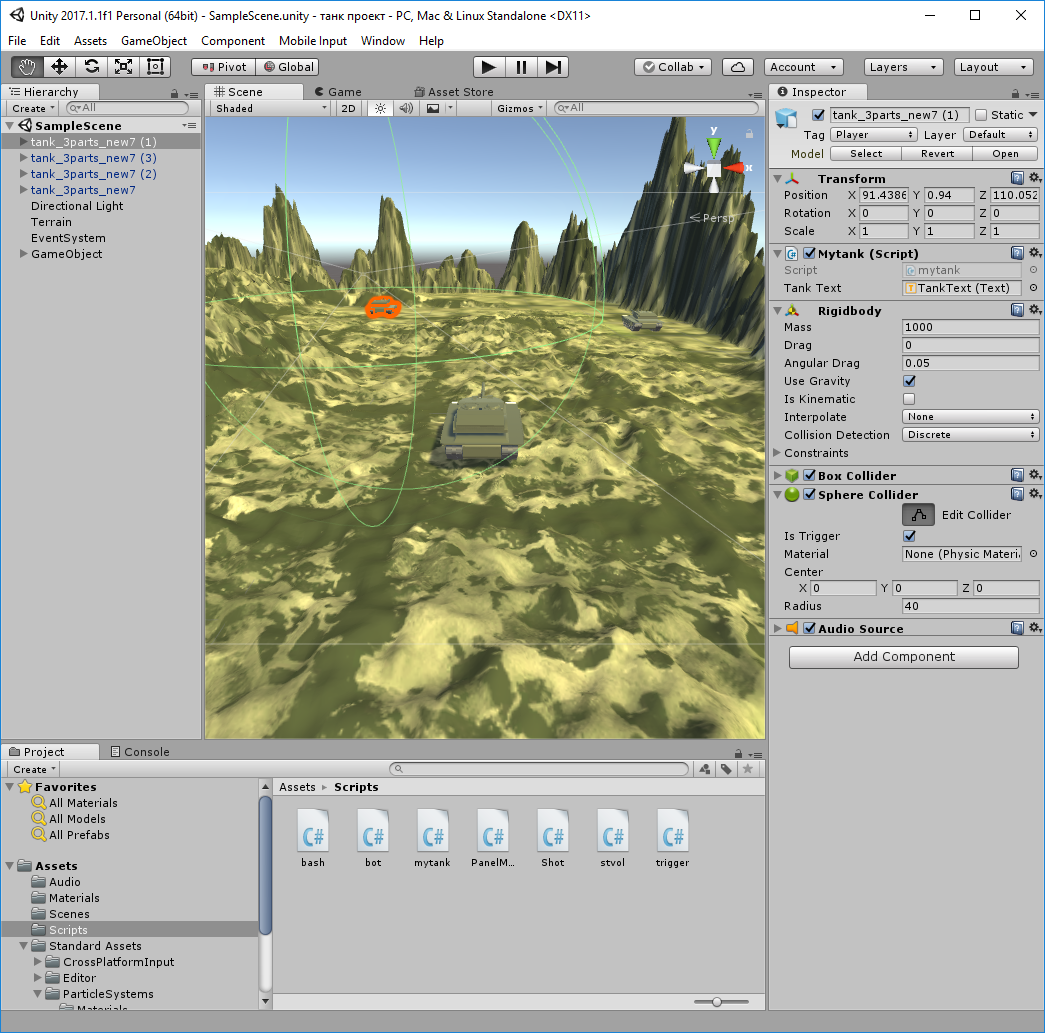
**АЛГОРИТМ СОЗДАНИЯ БОТА НА СЦЕНЕ** (шаги: 1, 2, 3, 4, 5)

1. **СОЗДАТЬ ПРЕФАБ** из уже имеющегося на сцене объекта танка-игрока

*Это самый простой вариант, хотя можно разработать для создания бота и совсем новый танк, действуя аналогично предыдущему по созданию основного танка, однако уже без программы управления ботом самим игроком.*

При этом все экземпляры префаба, размещаемые на сцене (**боты** – танкипротивника), будут обладать функциональностью, определяемую исходным танком-игроком**.**

1. **ДОБАВИТЬ НА ИСХОДНЫЙ ТАНК-ИГРОКА СФЕРИЧЕСКИЙ ТРИГГЕРНЫЙ КОЛЛАЙДЕР** с радиусом, соответствующим расстоянию, на котором будет обнаруживаться танк-игрока другими танками-противника (ботами) при попадании их в этот коллайдер при движении по сцене танка-игрока, при этом начнется «оживление» танка-бота с выполнением действий, предусмотренных созданным для него скриптом.



1. **СОЗДАТЬ НОВЫЙ СКРИПТ ДЛЯ ПРЕФАБА** – источника экземпляров танков-ботов, *при этом для префаба нужно* ***удалить******старыЙ скрипт******(!!!)*** *по управлению танком и его стволом*.

***АЛГОРИТМ ВЫПОЛНЕНИЯ ДЕЙСТВИЙ СКРИПТА ТАНКА-БОТА:***

1. *Обнаружить попадание в коллайдер танка-игрока танка-бота*
2. *При попадании в коллайдер повернуть башню танка-бота в сторону танка-игрока*
3. *Определить, нацелен ли ствол танка-бота на танк-игрока*
4. *Произвести выстрел танка-бота в танк-игрока с заданной с задержкой с помощью программы-короутины*
5. *Определить минимальную дистанцию, при которой начинается движение танка-бота в сторону танка-игрока*
6. *Начать движение танка-бота в сторону танка-игрока*
7. *Уничтожить танк-бот при попадании в него нескольких снарядов*

*ПРИ ДОБАВЛЕНИИ СОЗДАННОГО ДЛЯ ПРЕФАБА БОТА-ТАНКА СКРИПТА, НАПРИМЕР, С ИМЕНЕМ* ***«BOT»*** *РЕКОМЕНДУЕТСЯ ДЛЯ ИЗБЕЖАНИЯ ВОЗМОЖНЫХ «ГЛЮКОВ» СОБЛЮДАТЬ СЛЕДУЮЩУЮ ОЧЕРЕДНОСТЬ ДЕЙСТВИЙ:*

* 1. разместить экземпляр префаба танка-бота на сцене;
  2. «перетянуть» в **Assets** скрипт **Bot** на префаб танка;
  3. добавить на экземпляр префаба тег **«goal» -** для обозначения танка-бота как цели для снаряда танка-игрока;**;**
  4. связать объектные переменные для башни, ствола и снаряда в скрипте **Bot** танка-бота с соответствующими 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 соответствие) и нажмите кнопку **Apply;**
  5. удалить у танка-бота на сцене камеру (если она была привязана к исходному танку=игроку);
  6. удалить 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»**.

**СКРИПТ ДЛЯ ПРЕФАБА ТАНКА**

ОПРЕДЕЛИТЬ ПЕРЕМЕННЫЕ

**public class bot : MonoBehaviour {  
    float movespeed = 0.25f; //** скорость передвижения танка-бота **float rotspeedtank = 0.1f; //** скорость поворота танка-бота **float rotspeedbash = 0.5f; //** скорость поворота башни танка-бота **float speedcore = 3f; //** скорость снаряда танка-бота **public Transform bash; //** для управления башней **public Transform stvol; //** для управления стволом **public GameObject core; //** для ссылки на префаб снаряда **bool canshoot = true; //** для определения, может ли танк-бот произвести выстрел **int life = 3; //** для определения максимального количества попаданий в танк-бот

**...**

*ПРИ ЭТОМ НЕОБХОДИМО БУДЕТ ПРОИНИЦИАЛИЗИРОВАТЬ ОБЪЕКТНЫЕ ПЕРЕМЕННЫЕ* **public***ЧЕРЕЗ* **Inspector** *(УКАЗАТЬ НА СООТВЕТСТВУЮЩИЕ ОБЪЕКТЫ НА СЦЕНЕ, КОТОРЫЕ СВЯЗАНЫ С ЭТИМИ ПЕРЕМЕННЫМИ) ПОСЛЕ ДОБАВЛЕНИЯ СКРИПТА НА ТАНК-БОТ.*

**ДОБАВИТЬ В СКРИПТ БОТА МЕТОД *OnTriggerStay(Collider other)***

В методе определяется, какие действия должен выполнять танк-бот при нахождении в триггере танка-игрока (*в последующем коде* *главному танку-игроку присвоен тэг* ***Player)*.**

**private void OnTriggerStay(Collider other)  
    {  
   if (other.tag == "Player")**   
**{**

// если бот попал в триггер танка-игрока выполняем действия **А, Б, В** и **Г**

**А.** ОПРЕДЕЛИТЬКВАТЕРНИОН ДЛЯ ПОВОРОТА БАШНИ БОТА В СТОРОНУ ИГРОКА

// определяем вектор направления между игроком и ботом

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

// по найденному вектору определяем кватернион для поворотабашни бота

**Quaternion newrot = Quaternion.LookRotation(relativePos);**

*Здесь метод* ***LookRotation()*** *вычисляет кватернион, проворачивающий объект в направлении, указанном вектором в аргументе метода.*

**Б.** ПЛАВНО ПОВЕРНУТЬ БАШНЮ ТАНКА-БОТА В СТОРОНУ ТАНКА-ИГРОКА

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

*Метод* ***Slerp()*** *задает плавный поворот объекта от начального положения вектора поворота в его конечное положение (сферическая линейная интерполяция) с заданной скоростью.*

**В.** ИСПОЛЬЗОВАТЬ МЕТОД «БРОСАНИЯ ЛУЧЕЙ» для определения момента, повернута ли башня танка-бота на танк-игрока

//переменная **hit** для определения объекта, в который попал «луч» от танка-бота

**RaycastHit hit;**

//если выпущен луч из башни танка-бота в направлении относительно нее – вперед

**If(Physics.Raycast(bash.position, bash.TransformDirection(Vector3.forward), out hit))**

**{**

//если луч попал в коллайдер игрока и можно выстрелить

**If((hit.transform.tag==”Player”) && canshoot)**

//запускаем программу-короутину **botshoot()** для выстрела танка-бота

**StartCoroutine(botshoot());**

**}**

*Метод* ***Physics.Raycast*** *создает «луч» в заданном направлении.*

**Г**. ЗАДАТЬ ПЛАВНЫЕ ДВИЖЕНИЕ И ПОВОРОТ ТАНКА-БОТА В НАПРАВЛЕНИИ К ТАНКУ-ИГРОКУ

// вычисляем дистанцию бота до игрока

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

// если дистанция меньше заданной (20)

**If(distance<20)**

**{**

//плавно поворачиваем танк-бот в сторону танка-игрока с заданной скоростью

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

// плавно двигаем танк-бота в сторону игрока

**transform.position = new Vector3.Lerp(transform.position,other.transform.position,Time.deltaTime\*movespeed),**

**}**

*Метод* ***Lerp()*** *задает плавный переход в 3D-пространстве от одной позиции к другой с заданной скоростью (линейная интерполяция – linear interpolation).*

*Сопрограммы* ***– короутины (coroutines)*** *в Unity выполняются параллельно основной программе в течение некоторого времени. Этим они отличаются от большинства функций, заставляющих программу ждать окончания своей работы. Короутины обычно используются для задержки выполнения какого-либо действия во время выполнения основной программы.*

*Короутины для своего выполнения используют встроенную функцию* ***IEnumerator.*** *Главным компонентом в теле короутины является ключевое слово* ***yield****, временно прерывающее ее работу, возвращающее управление основной программе и в следующем кадре возобновляющее сопрограмму с прерванной точки с заданным интервалом ожидания в секундах функцией* ***WaitForSeconds().***

КОРОУТИНА ДЛЯ ЗАПУСКА ВЫСТРЕЛА ТАНКА-БОТА ЧЕРЕЗ ПРОМЕЖУТОК ВРЕМЕНИ:

**IEnumerator botshoot()**

**{**

//указываем, что танк-бот стрелять пока не может

**canshoot = false;**

//определяем координату для положения снаряда танка-бота

**Vector3 forwardofstvol = stvol.transform.position + stvol.TransformDirection(Vector3.forward\*4f);**

//создаем снаряд из префаба снаряда в требуемой координате относительно ствола

**GameObject newcore = Instantiate(core, forwardofstvol, stvol.rotation);**

//ждем 3 секунды (время «перезарядки»)

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

//указываем, что танк может сделать выстрел

**canshoot = true;**

**}**

КОД ДЛЯ УНИЧТОЖЕНИЯ ТАНКА-БОТА

Добавить в скрипт префаба метод **OnCollisionEnter ()**, внутри которого определить по тэгу для снаряда его коллизию со снарядом:

**private void OnCollisionEnter(Collision col)  
    {  
        if (col.gameObject.tag == "core")  
        {  
            life - -;**//если количество «жизней» меньше 1

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