Waypoints patroling tutorial

This shows how to make an AI patrol between way points

1. Create a new scene

Start by creating a new scene called `AIPatroling`.

Add a plane called `Floor`, a 3D Cube named `Enemy` and a 3D very small Cube named `WayPoint`.

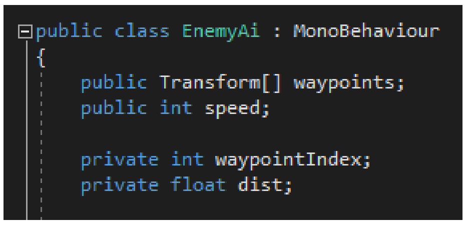
Duplicate the small 3D Cube, this wil give us three `WayPoints`.

Make three new materials and asing a different colour for each one. Add a different color to the `Floor`, `Enemy` and `Waypoints`.

2. Create the script and fill the function Start

Create a new folder called Scripts and create a new script called `EnemyAI`.

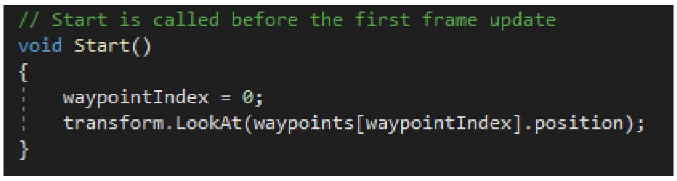
We are going to make our enemy patrol between specific waypoints, in any order we want, and a speed public variable so we can choose how fast we want the enemy to patrol around. To make this we will need a `public Transform` which means wherever the waypoints are, a `public int` for the speed so we can change it in inside Unity. We will also make a `private int waypointsIndex` which will show us the array in the inspector and a `float dist` to check the distance between our AI and the current waypoint.



```
public class EnemyAi : MonoBehaviour
{
    public Transform[] waypoints;
    public int speed;

    private int waypointIndex;
    private float dist;
```

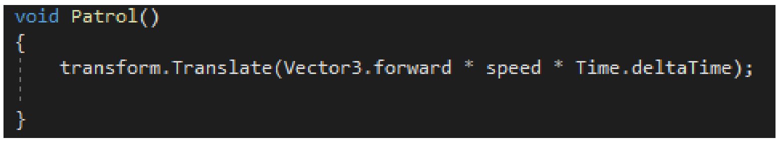
Inside of Start, we will need to set our waypointIndex to 0 so our AI starts walking to the first waypoint in the array. A `transform.LookAt` will make our AI to face the object it is heading to.



```
// Start is called before the first frame update
void Start()
{
    waypointIndex = 0;
    transform.LookAt(waypoints[waypointIndex].position);
}
```

3. Add extra functions

Once start is done, we will ignore Update and create a function called `void Patrol` to set the AI.

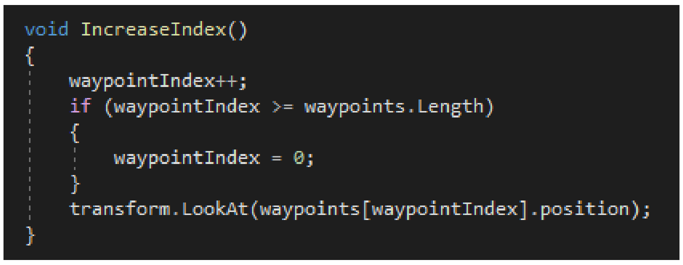


```
void Patrol()
{
    transform.Translate(Vector3.forward * speed * Time.deltaTime);

}
```

The next function called `void IncreaseIndex` will do the job that its name says. We will add our `waypointIndex` and a conditional clause `if` to make sure that when the `Array` is finished, it goes back to 0 and the AI moves to the first waypoint in the `Array`.

Again add a lookAt function to make the AI face where its heading.

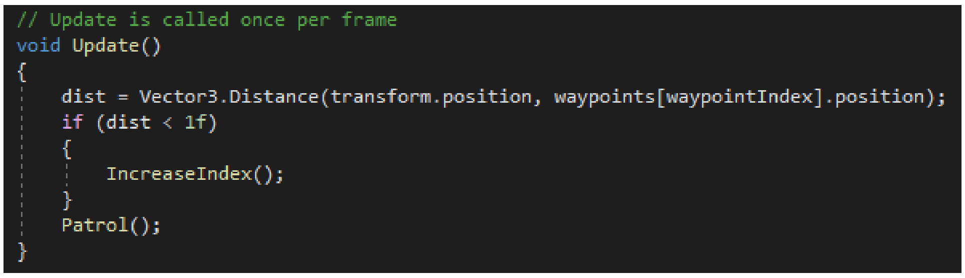


```
void IncreaseIndex()
{
    waypointIndex++;
    if (waypointIndex >= waypoints.Length)
    {
        waypointIndex = 0;
    }
    transform.LookAt(waypoints[waypointIndex].position);
}
```

4. Fill the function Update

We will add a `dist` which will check the distance between the AI and the waypoints, adding a `if` clause will let us decide how close we want our AI to get to the `waypoint` before increasing the `Index`, making our AI head to the next `waypoint` in the `Array`.

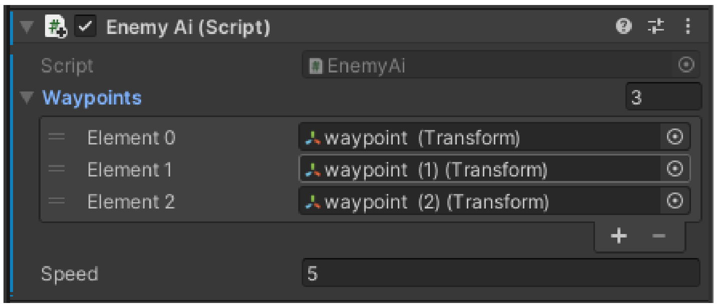
We will add a `Patrol()` to make sure the AI is patroling in every moment.



```
// Update is called once per frame
void Update()
{
    dist = Vector3.Distance(transform.position, waypoints[waypointIndex].position);
    if (dist < 1f)
    {
        IncreaseIndex();
    }
    Patrol();
}
```

5. In the inspector

Add the script to our `Enemy`. In the inspector choose a number size for the waypoints, we currently have 3 so we will need to write that down. In every `waypoint` element drag one of the small 3D cubes until filling every space and set a speed that suits your AI.



Enemy Ai (Script)

Script	EnemyAi
Waypoints	3
Element 0	waypoint (Transform)
Element 1	waypoint (1) (Transform)
Element 2	waypoint (2) (Transform)
Speed	5

6. Testing

Test by running the game. Check if the speed and AI are working properly and set your speed to any number that suits your game, you can also add as many waypoints as you would like.