Simple AI for 2D platformer

Introduction

In this package you can find a Plugin and a TestScene folder.

In the TestScene folder there is a little simple game where I show how the plugin works. So it's easier to understand how it works. (btw. it's pretty easy to use the plugin.)

In the Plugin folder you'll find 3 scripts (EnemyController, EnemyShooting and FlyingFollower) and a prefab called JumpBox.

TestScene

I used the plugin to make the enemies patrol, follow the player and shoot, and to make that little Flying Follower that follows the Player. I created an enemy and then just drag and drop the EnemyController script and the EnemyShooting script. Then I just set up the speed and the other things what you can see on the inspector window. For the flying follower I drag and drop the FlyingFollower script and then I made the changes in the script what is necessary to make it work.

2D AI PLATFORMER

Plugin

Little Set Up:

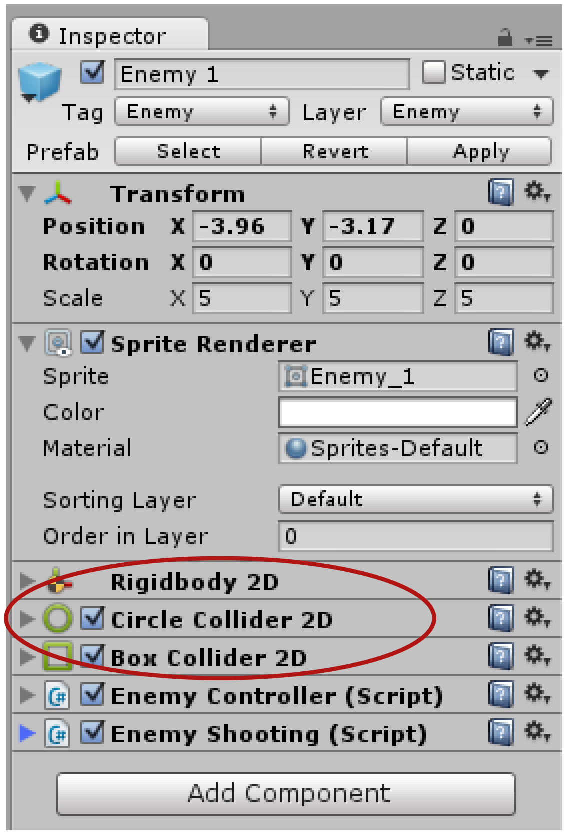
First of all, you have to do a little set up if you would like to use the test scene.

1. Edit -> Project Settings -> Physics2D: here I used Gravity y = -20.
2. Create a new layer called "Enemy" and then set it to the enemy prefab.
3. Edit -> Project Settings -> Physics2D: Edit the Layer Collision Matrix: Enemy and Enemy can't collide with each other so untick it.
4. If you would like to make the enemy/enemies follow the player than create a Player tag, and add it to the player.

EnemyController Script

What you will need to make this script work:

Attach the following things to your enemy sprite:


Inspector

✓ Enemy 1 — Static: ☐

Tag: Enemy — Layer: Enemy

Prefab: Select | Revert | Apply

Transform

Position	X	-3.96	Y	-3.17	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	5	Y	5	Z	5

✓ Sprite Renderer

Sprite	Enemy_1
Color	
Material	Sprites-Default
Sorting Layer	Default
Order in Layer	0

Rigidbody 2D

✓ Circle Collider 2D

✓ Box Collider 2D

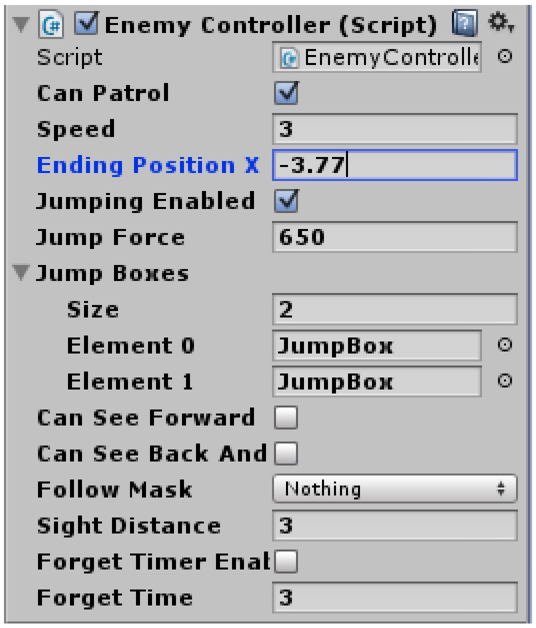
✓ Enemy Controller (Script)

✓ Enemy Shooting (Script)

Add Component

The **Rigidbody2D** and a **collider** is necessary!

You don't necessarily need 2 colliders. 1 is enough. I use 2 because of the smoother movement

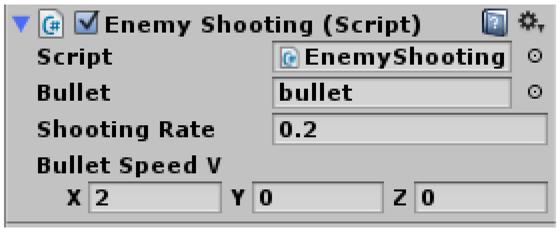


Enemy Controller (Script)	✓
Script	EnemyControlle
Can Patrol	✓
Speed	3
Ending Position X	-3.77
Jumping Enabled	✓
Jump Force	650
Jump Boxes	
Size	2
Element 0	JumpBox
Element 1	JumpBox
Can See Forward	☐
Can See Back And	☐
Follow Mask	Nothing
Sight Distance	3
Forget Timer Enal	☐
Forget Time	3

- **Can Patrol**: tick it if the enemy is not idle so it will patrol on the scene between to given coordinates.
- **Speed**: The speed of the enemy (how fast it patrols)
- **Ending Position X**: This is where it will goes. This is the destination of the enemy
- **Jumping Enabled**: If it's ticked it will jump with the given jumpBoxes
- **Jump Force**: This is the force what we use for the enemy jump
- **Jump Boxes**: You have to put those jumpBoxes here what you would like to effect the enemy, so it will jump
- **Can See Forward**: If it's ticked, the enemy can detect what is in front of it (you can set the distance)
- **Can See Back And Front**: If it's ticked, the enemy can detect what is in front of it and can detect if the target gameObject is reaching from back (you can set the distance)
- **Follow Mask**: It will detect the GameObjects what is on that layer. (it can be more than one)
- **Sight Distance**: How far it can detect (how far it can see). If you play the scene you can see a red line on the editor window so you can set it precisely
- **Forget Timer Enabled**: you can set a forget timer. It the target is out of the sight that it will wait for a given time to detect the target again. If it can't detect anything than it will continue patrolling but if it detects something than it will follow it
- **Forget Time**: This is the time it will wait (only if Forget Timer Enabled is ticked)

EnemyShooting Script:

With this script you can make your enemies shoot bullets. Just attach it to the enemy.



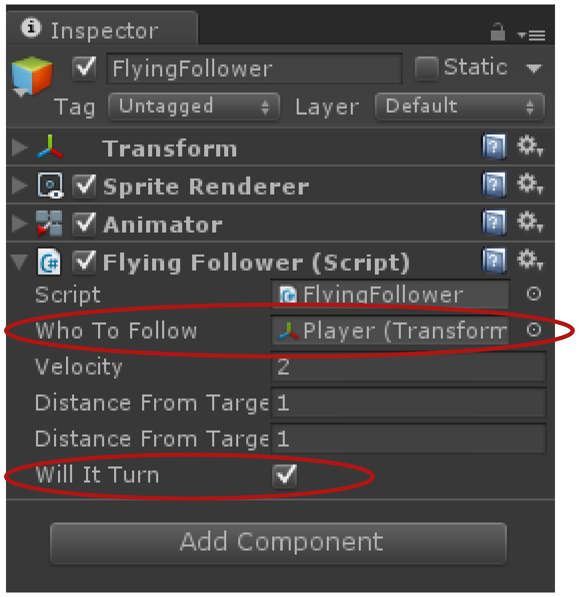
Enemy Shooting (Script)

Script	EnemyShooting
Bullet	bullet
Shooting Rate	0.2
Bullet Speed V	X 2 Y 0 Z 0

- **Bullet**: This should be a prefab, what is the bullet it will shoot (just drag and drop here your bullet prefab)
- **Shooting Rate**: How fast will it shoot
- **Bullet Speed V**: This is the shooting direction (if you set the x than it will shoot horizontally)

FlyingFollower script

Okay…so first you need something what you can attach this script. Then just use the drag and drop method to add the script to it.



Inspector

FlyingFollower Static

Tag Untagged Layer Default

Transform

Sprite Renderer

Animator

Flying Follower (Script)

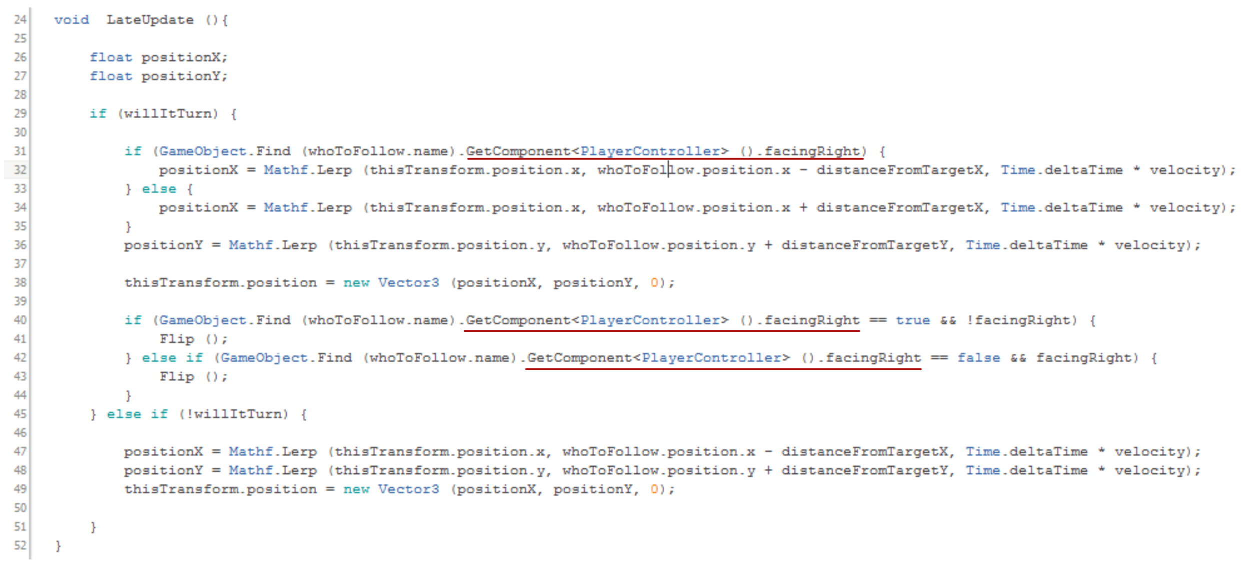
Script	FlyingFollower
Who To Follow	Player (Transform
Velocity	2
Distance From Targe	1
Distance From Targe	1
Will It Turn	✓

To make this script work you don't need necessary a RigidBody2D component buti f you attach it then check **Is Kinematic.**

Drag and drop the player to **the Who To Follow** field.

If the flying follower's sprite is not turning then just uncheck the **Will It Turn box.**

In case it will turn (left to right or right to left) then tick it and then **you have a little thing you have to do in the code.**



```
void  LateUpdate (){

    float positionX;
    float positionY;

    if (willItTurn) {

        if (GameObject.Find (whoToFollow.name).GetComponent<PlayerController> ().facingRight) {
            positionX = Mathf.Lerp (thisTransform.position.x, whoToFollow.position.x - distanceFromTargetX, Time.deltaTime * velocity);
        } else {
            positionX = Mathf.Lerp (thisTransform.position.x, whoToFollow.position.x + distanceFromTargetX, Time.deltaTime * velocity);
        }
        positionY = Mathf.Lerp (thisTransform.position.y, whoToFollow.position.y + distanceFromTargetY, Time.deltaTime * velocity);

        thisTransform.position = new Vector3 (positionX, positionY, 0);

        if (GameObject.Find (whoToFollow.name).GetComponent<PlayerController> ().facingRight == true && !facingRight) {
            Flip ();
        } else if (GameObject.Find (whoToFollow.name).GetComponent<PlayerController> ().facingRight == false && facingRight) {
            Flip ();
        }
    } else if (!willItTurn) {

        positionX = Mathf.Lerp (thisTransform.position.x, whoToFollow.position.x - distanceFromTargetX, Time.deltaTime * velocity);
        positionY = Mathf.Lerp (thisTransform.position.y, whoToFollow.position.y + distanceFromTargetY, Time.deltaTime * velocity);
        thisTransform.position = new Vector3 (positionX, positionY, 0);

    }
}
```

The script uses another script's variable and it's called **facingRight** in my script**.** This variable is necessary to make the object follow the other properly and to make it turn.

You just have to replace the *PlayerController* and the *facingRight* fields with the ones you use in your script.

1. The *PlayerController*: Replace it with the script's name what controls the target object and contains a bool variable what tells if the player is "looking" to the right.
2. The *facingRight*: Replace it with a public bool variable's name what is in the scrip (what you found in the 1st step).

And that's it for the FlyingFollower's use.

Have fun with this plugin and stay tuned for the updates.

support email: vecseigabor.x@gmail.com

blog: https://gaborvecsei.wordpress.com/