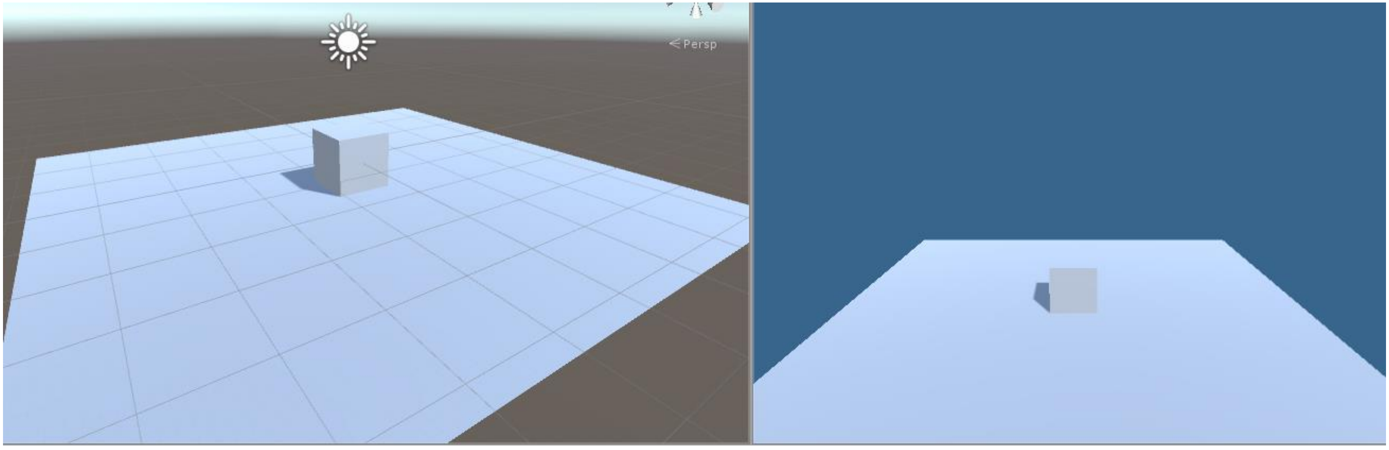
Unity & PlayMaker: Move Towards, Instantiate & Collision.

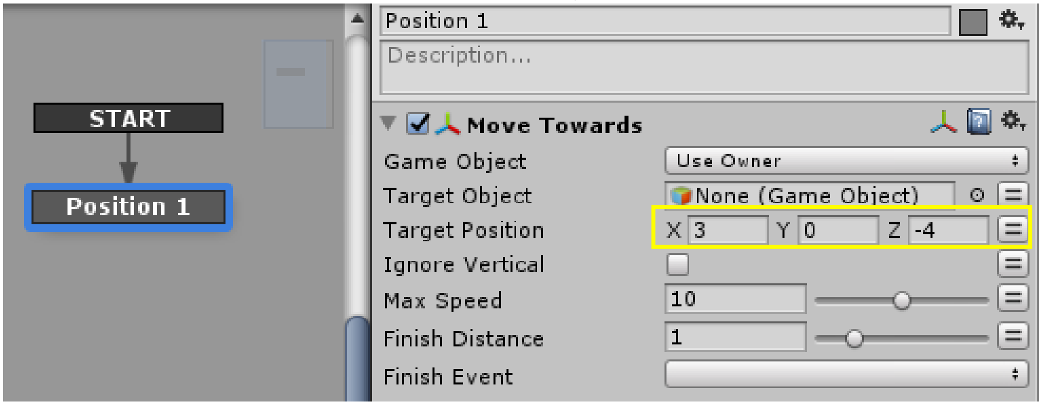
Basic AI behaviour (Move towards action)

Add a plane with a cube, so that the scene looks like this.



We will create a basic patrolling behaviour.

Click on the cube and a state called “**Position 1**”, add “**Move Towards**” action.



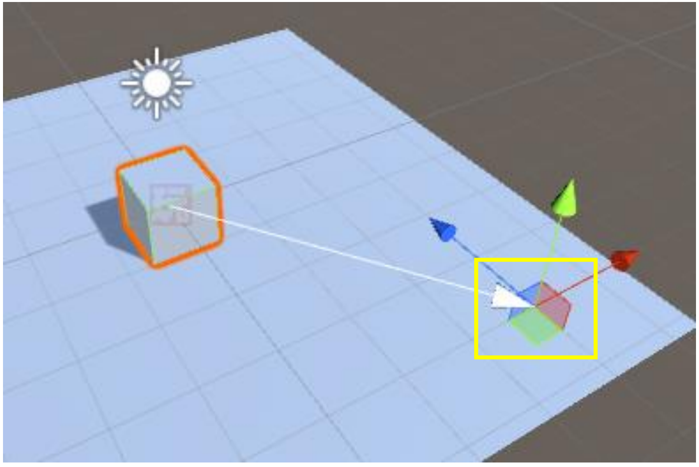
Position 1

Description...

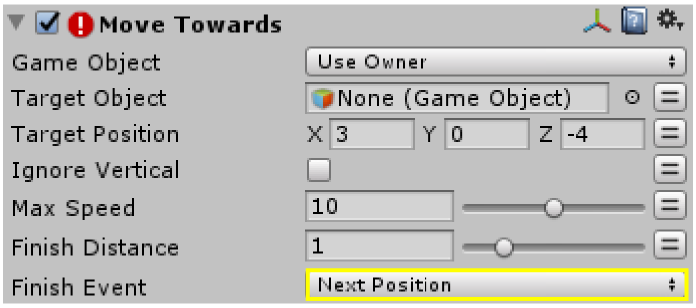
☑ Move Towards

Game Object	Use Owner
Target Object	None (Game Object)
Target Position	X 3 Y 0 Z -4
Ignore Vertical	□
Max Speed	10
Finish Distance	1
Finish Event	

Change “Target Position” to **(3,0, -4)** or you can move around the little position gizmo in the scene.



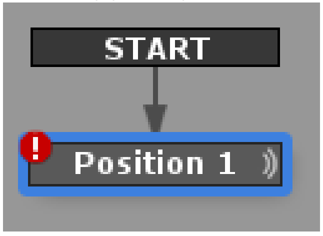
Add “Finish Event” called “Next Position”.



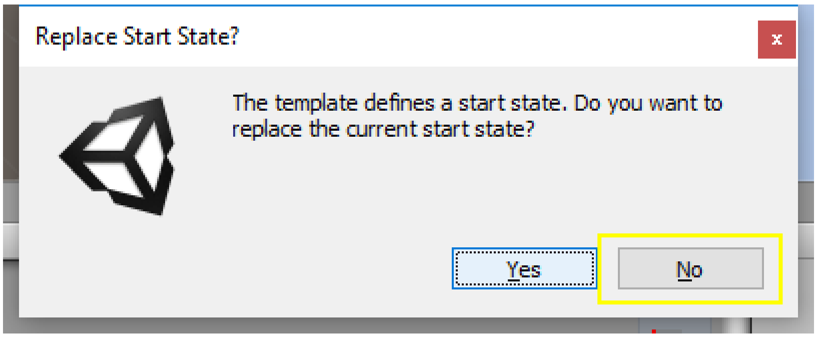
☑ Move Towards

Game Object	Use Owner
Target Object	None (Game Object)
Target Position	X 3 Y 0 Z -4
Ignore Vertical	□
Max Speed	10
Finish Distance	1
Finish Event	Next Position

Now copy and paste this state. Select it and press [**Control + C**] and [**Control + V**].



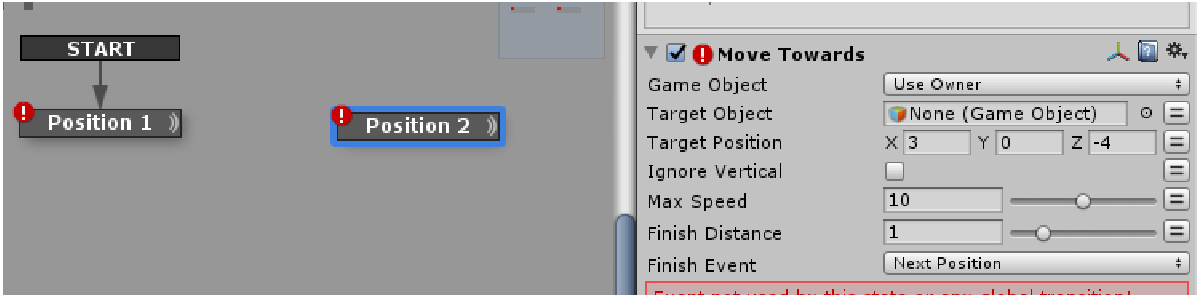
Select “**No**”.



Replace Start State?

The template defines a start state. Do you want to replace the current start state?

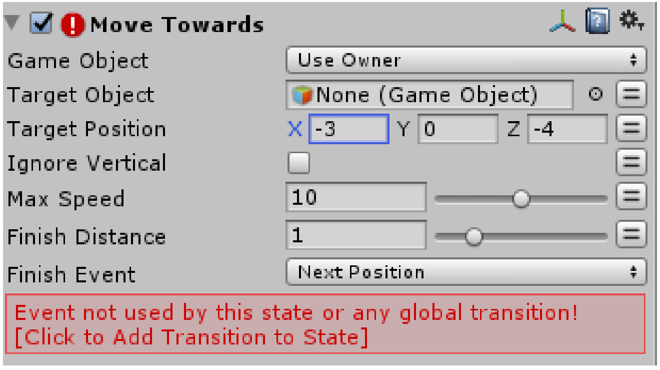
Now you should see another state, with the same action.



☑ Move Towards

Game Object	Use Owner
Target Object	None (Game Object)
Target Position	X 3 Y 0 Z -4
Ignore Vertical	□
Max Speed	10
Finish Distance	1
Finish Event	Next Position

Change its target position to **(-3,0, -4)**

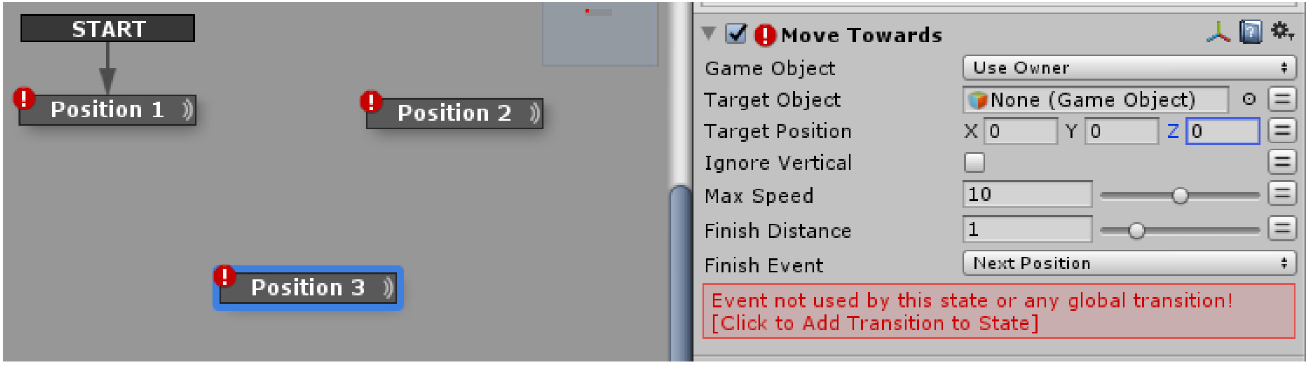


☑ Move Towards

Game Object	Use Owner
Target Object	None (Game Object)
Target Position	X -3 Y 0 Z -4
Ignore Vertical	□
Max Speed	10
Finish Distance	1
Finish Event	Next Position

Event not used by this state or any global transition!
[Click to Add Transition to State]

Paste another state, and change its target position to **(0,0,0)**

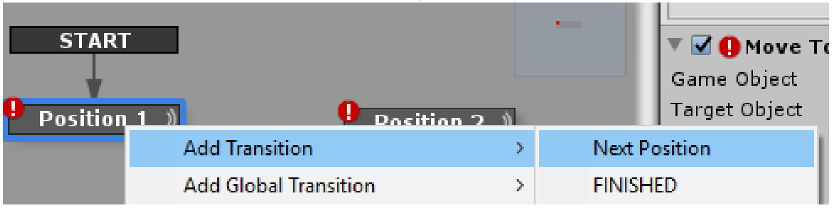


☑ Move Towards

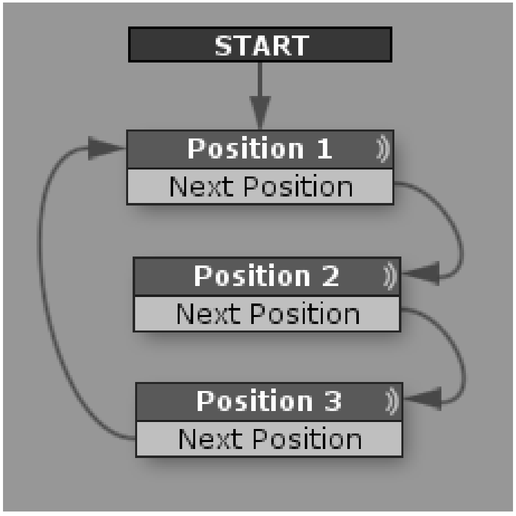
Game Object	Use Owner
Target Object	None (Game Object)
Target Position	X 0 Y 0 Z 0
Ignore Vertical	□
Max Speed	10
Finish Distance	1
Finish Event	Next Position

Event not used by this state or any global transition!
[Click to Add Transition to State]

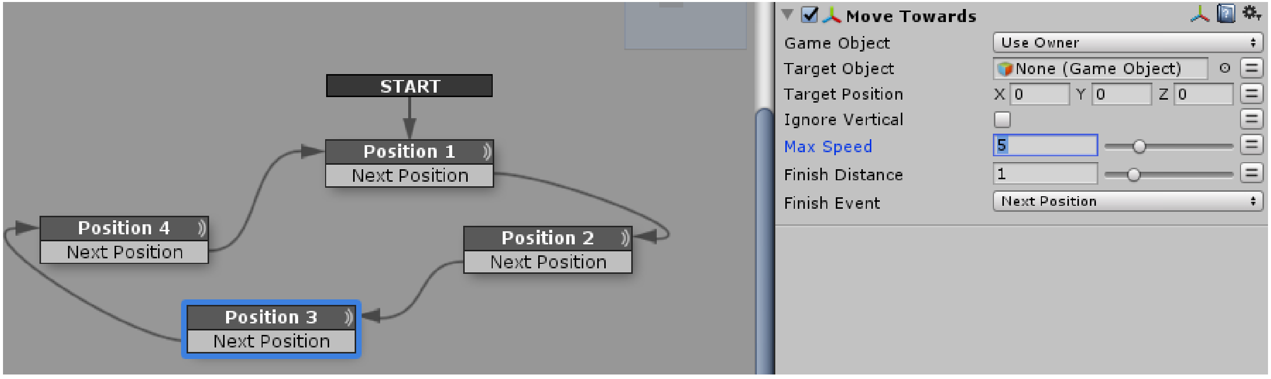
Now connect all of the states, using “**Next Position**” transition.



The complete FSM should look like this.



Run the game, and you should see the cube alternating between these positions.
You can play around and add more position states, change the speed of the movement etc.



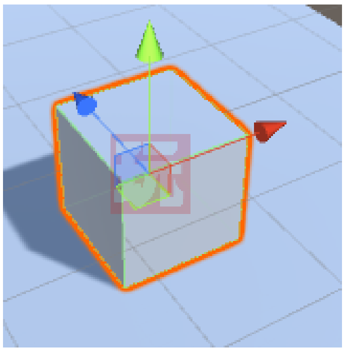
Move Towards

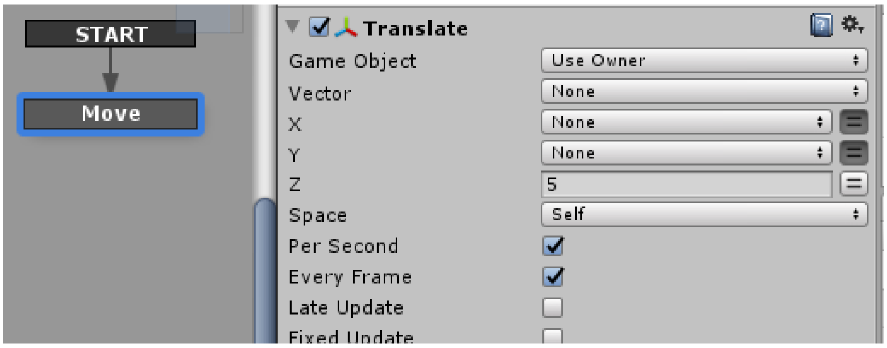
Game Object	Use Owner
Target Object	None (Game Object)
Target Position	X 0 Y 0 Z 0
Ignore Vertical	□
Max Speed	5
Finish Distance	1
Finish Event	Next Position

Instantiating Prefab

Here we will create a spawner (object creating other objects), and a prefab (prefabricated object).

First, let’s create a basic bullet object, add a cube and rename it “**Bullet**” then give it a translate state, that moves it on Z axis.

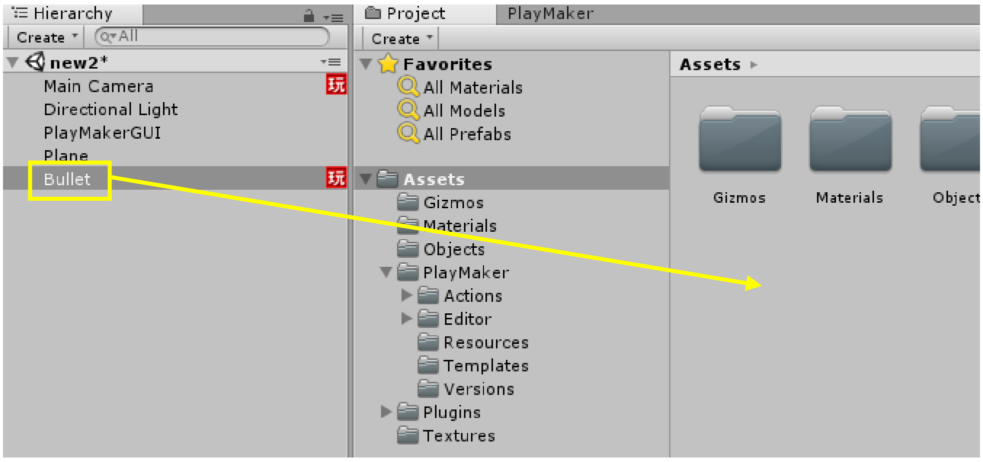


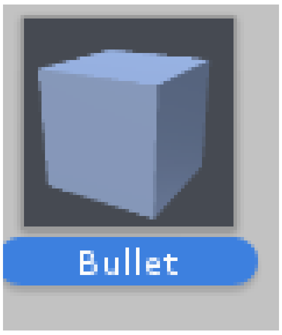


Translate

Game Object	Use Owner
Vector	None
X	None
Y	None
Z	5
Space	Self
Per Second	☑
Every Frame	☑
Late Update	□
Fixed Update	□

Drag and drop the “**Bullet**” object into the Project window.

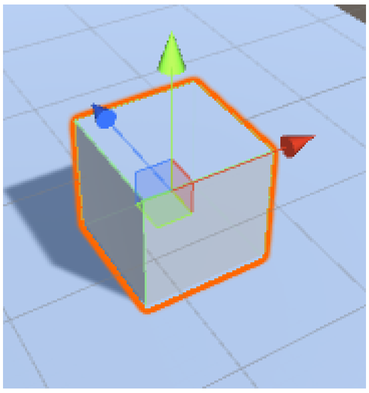




This will make it into a prefab.

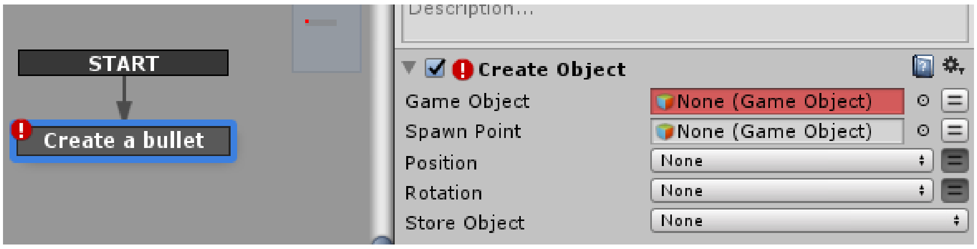
Now you can delete the “**Bullet**” from the Scene view.

Create a new cube, rename it to “Bullet Spawner”



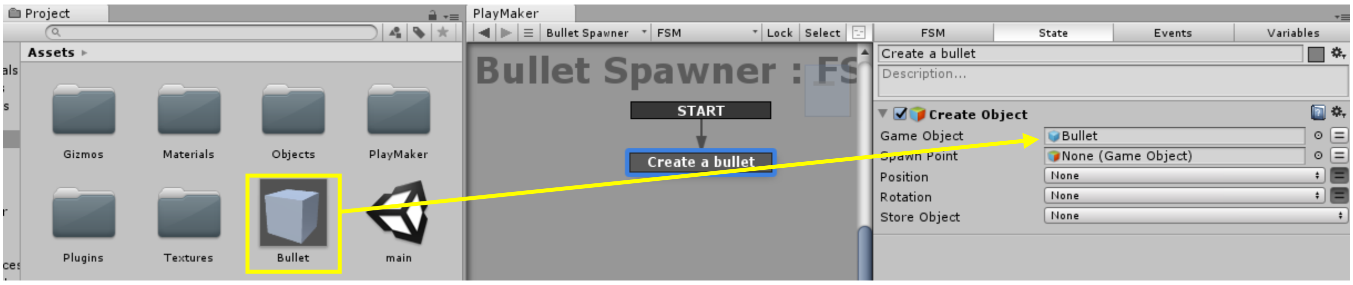
We will use this object to create Bullets.

Add a state called “**Create a bullet**”, add “Create Object” Action.



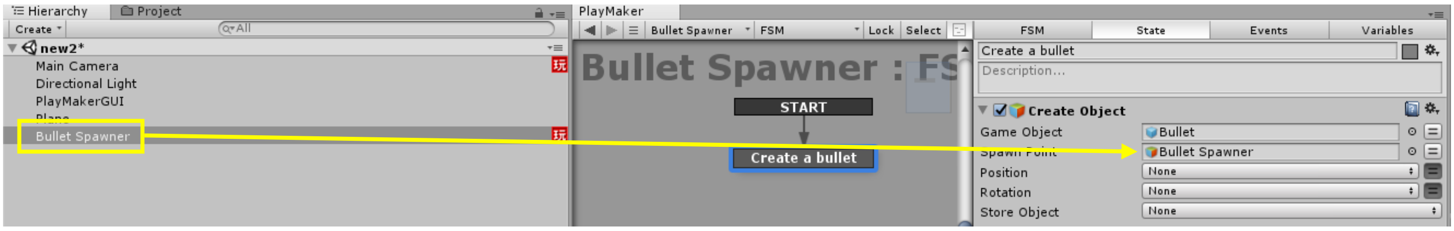
Game Object: None (Game Object)
Spawn Point: None (Game Object)
Position: None
Rotation: None
Store Object: None

Add “**Bullet**” prefab from the **Project window** into the **Game Object** property.



Game Object: Bullet
Spawn Point: None (Game Object)
Position: None
Rotation: None
Store Object: None

Add “Bullet Spawner” from the Hierarchy to the “**Spawn Point**”.



Create a bullet

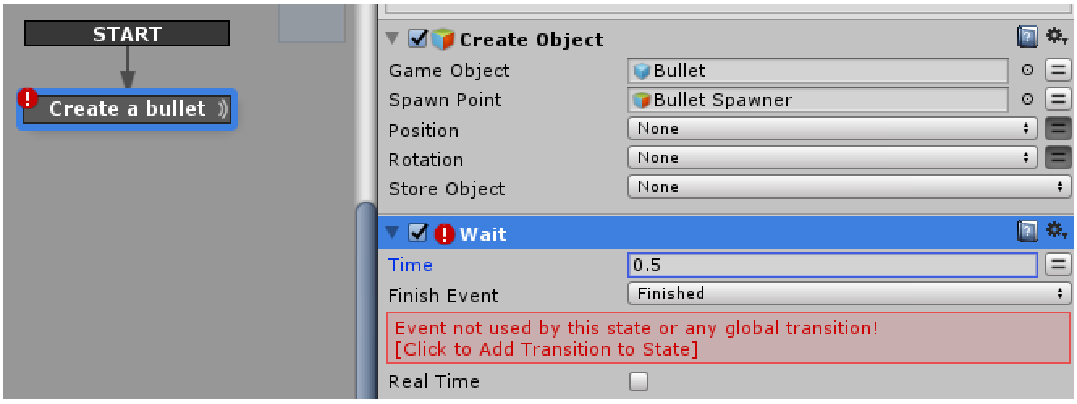
Description...

Create Object

Game Object	Bullet
Spawn Point	Bullet Spawner
Position	None
Rotation	None
Store Object	None

If you run the game now, the spawner will create one bullet from its position.

Add a “**Wait**” action, set Time to 0.5 and set Finish Event to “**Finished**”.



☑ Create Object

Game Object	Bullet
Spawn Point	Bullet Spawner
Position	None
Rotation	None
Store Object	None

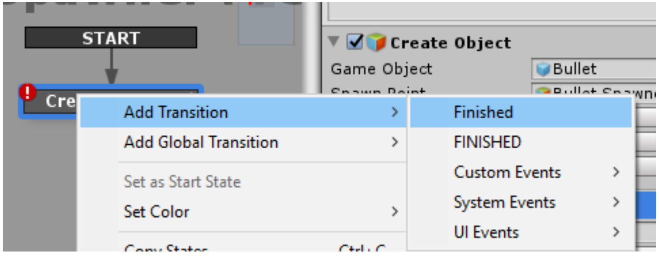
☑ Wait

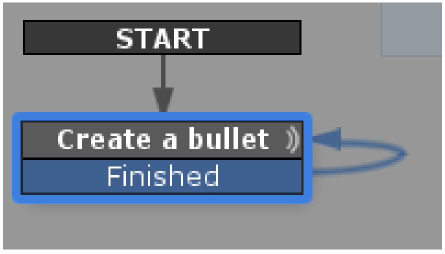
Time	0.5
Finish Event	Finished

Event not used by this state or any global transition!
[Click to Add Transition to State]

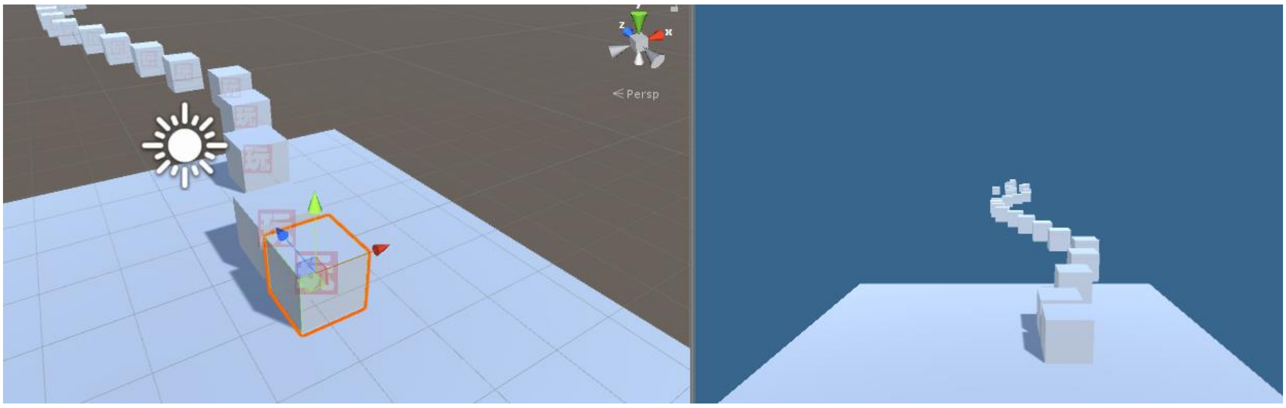
Real Time □

Add this transition to the state, connect it to the current state.
Effectively, this will create a bullet, wait for half a second, and repeat the process.





Run the game, you will now spawn a lot of bullets.



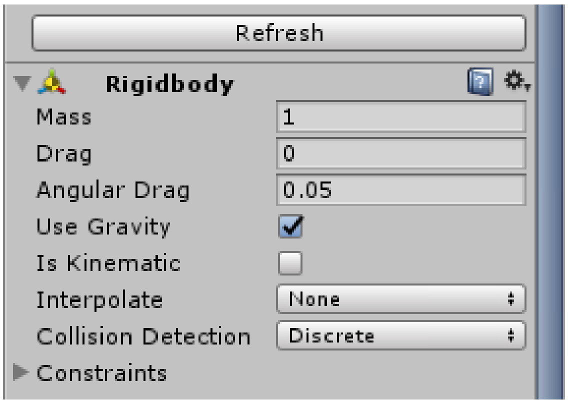
You can move or rotate the spawner to shoot bullets into different directions.

Collision detection

To detect collision between two objects, one of the objects needs to have a “**Rigidbody**” component.

Add “**Rigidbody**” component to the Bullet in the **Project window**.

Select the bullet, and in the Inspector click “add component” and add Rigidbody.



Rigidbody

Mass 1

Drag 0

Angular Drag 0.05

Use Gravity ☑

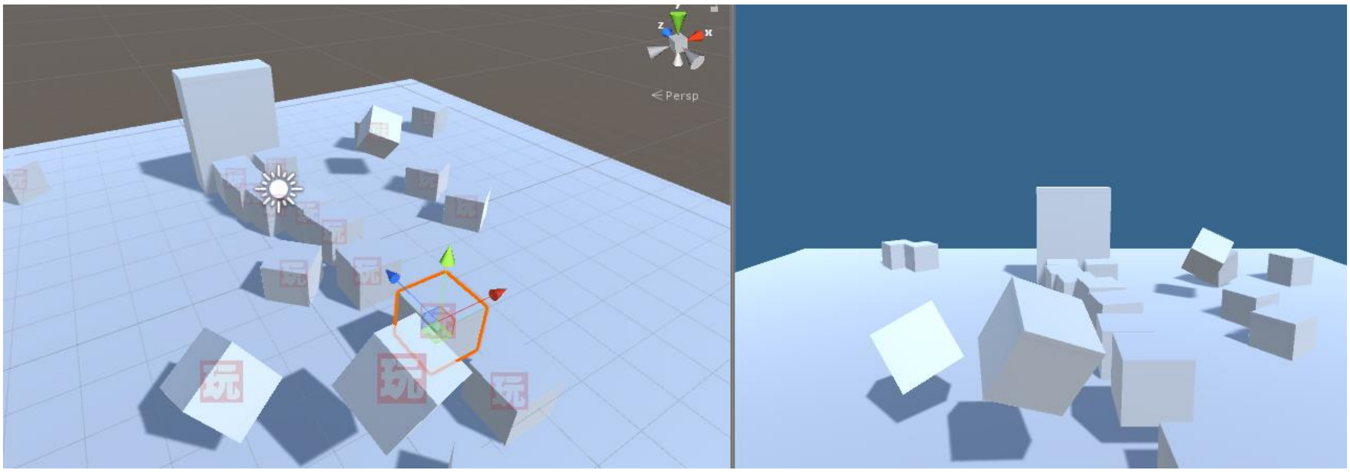
Is Kinematic □

Interpolate None

Collision Detection Discrete

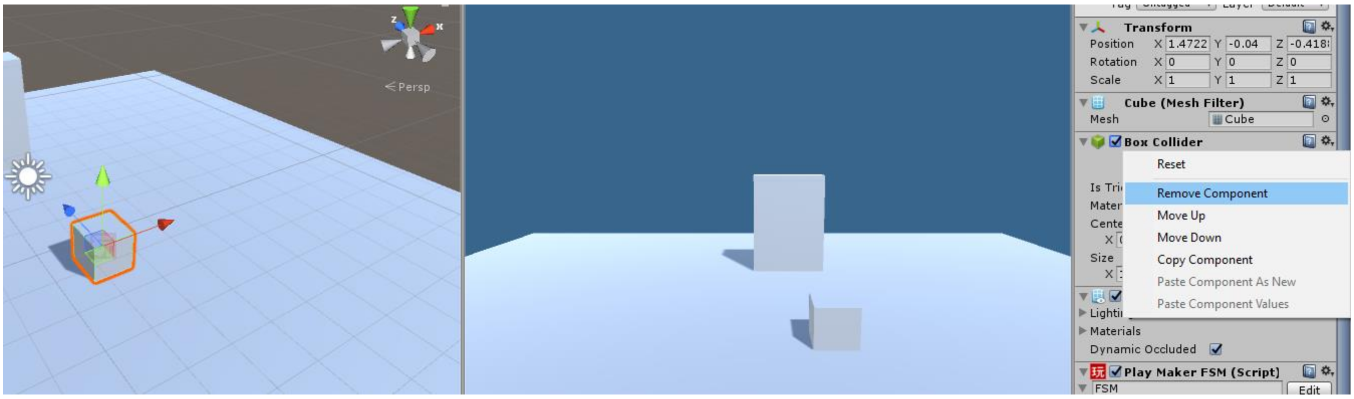
Constraints

Run the game, you should see that all the bullets have physical properties now.



You can see, the bullets are colliding with the spawner and move in unpredictable ways.

To fix that remove box collider from the spawner.



Transform

Position X 1.4722 Y -0.04 Z -0.418

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Cube (Mesh Filter)

Mesh Cube

☑ Box Collider

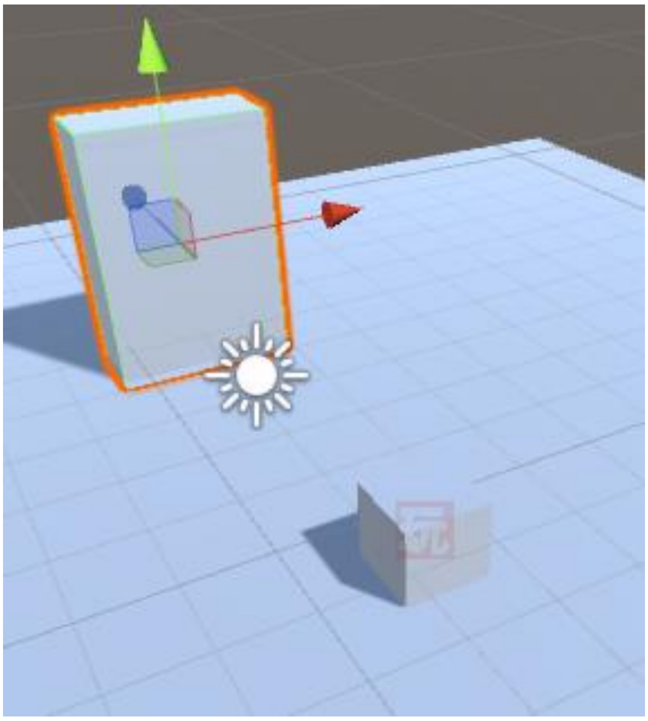
Dynamic Occluded ☑

☑ Play Maker FSM (Script)

FSM

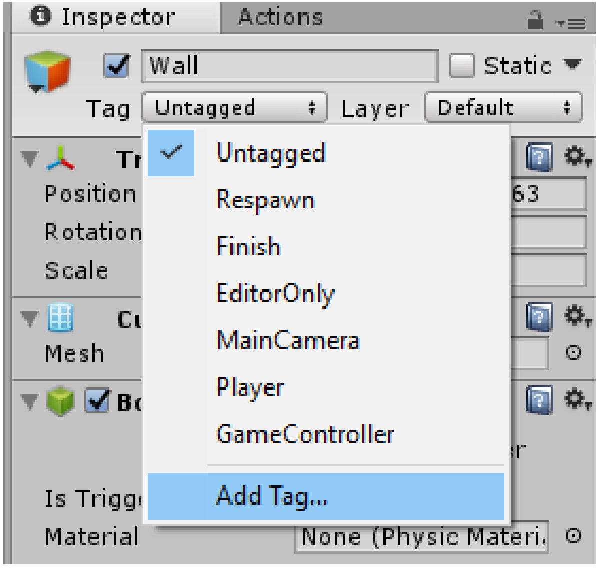
If you run the game now, the spawner should shoot bullets more consistently.

Now add an object name it "Wall", shape it like a wall.

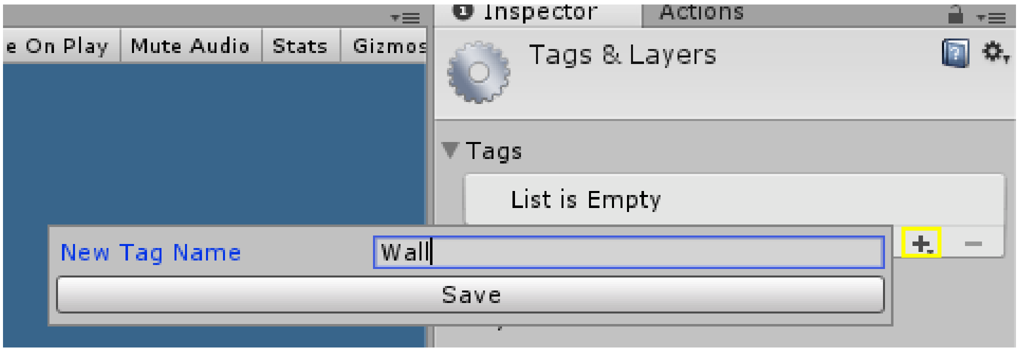


Add a Tag.

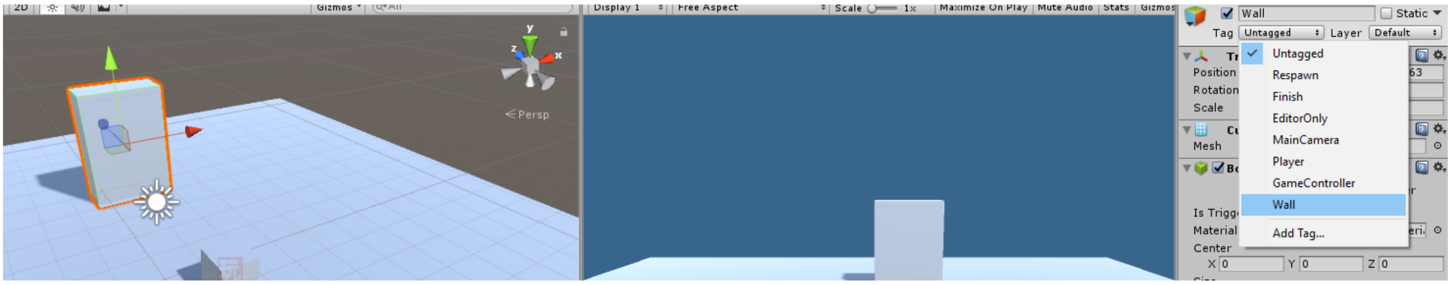
Tags are used to distinguish objects by type, in this case we will be checking if the bullet has collided with a "Wall".



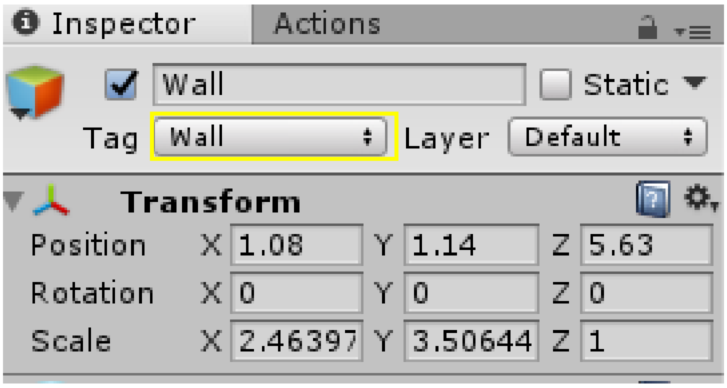
Click on the plus and add a new tag name.



Now select the wall in the scene and give it the new tag.



Make sure the new tag is selected.



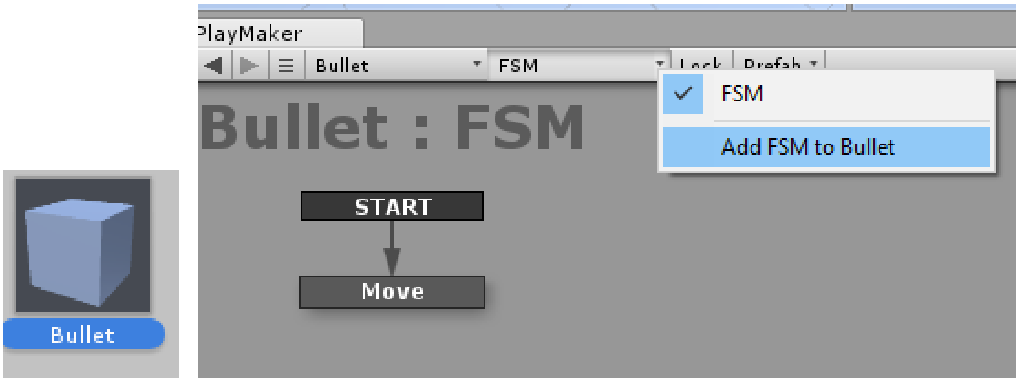
☑ Wall ☐ Static

Tag Wall Layer Default

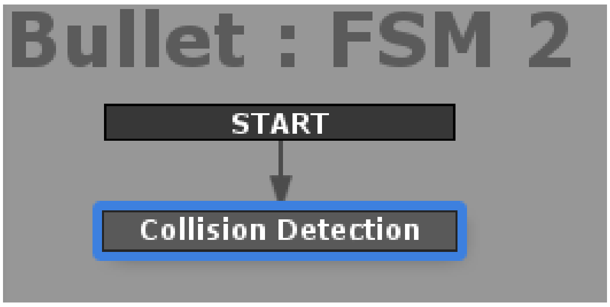
	X	Y	Z
Position	1.08	1.14	5.63
Rotation	0	0	0
Scale	2.46397	3.50644	1

Go to the Project window select the bullet prefab, add a new FSM.

Objects can have more than one FSM, in this case one will be used for movement, and the other one for collision.

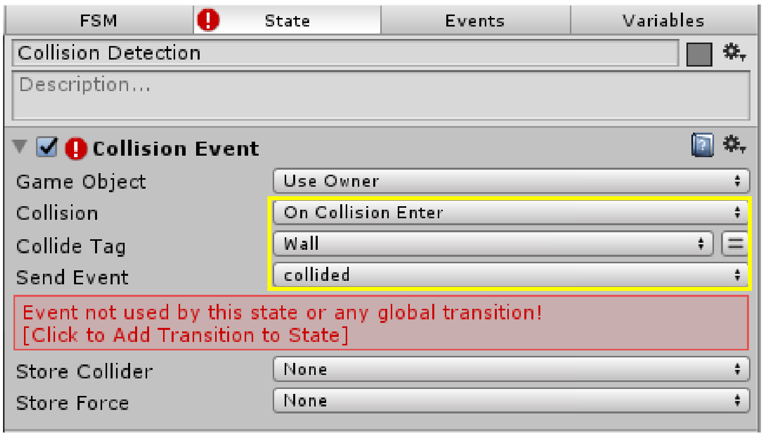


Add a new state called “Collision Detection”.



Now add a “Collision Event” action.

Add the properties shown below.



Collision Detection

Description...

☑ Collision Event

Game Object: Use Owner

Collision: On Collision Enter

Collide Tag: Wall

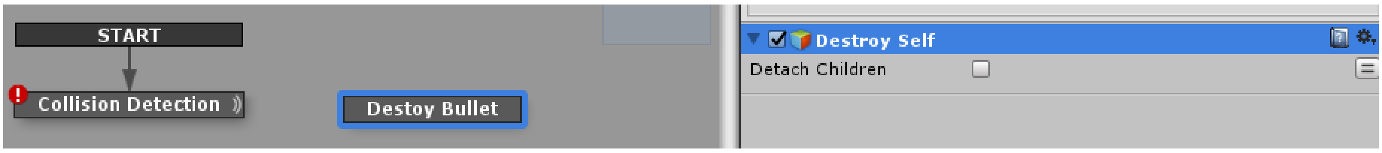
Send Event: collided

Event not used by this state or any global transition!
[Click to Add Transition to State]

Store Collider: None

Store Force: None

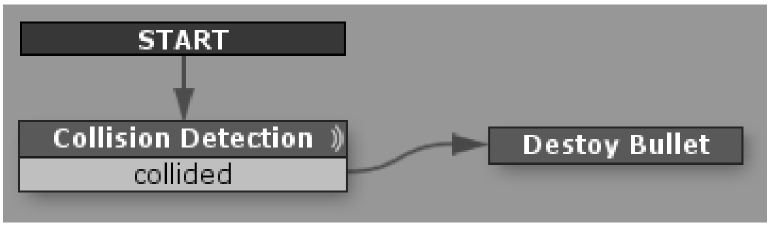
Then add a new state, which will remove the bullet. Using “**Destroy Self**” action.



☑ Destroy Self

Detach Children ☐

Link it together.



Run the game, the bullets will be deleted when they will hit the objects with the tag “**Wall**”.