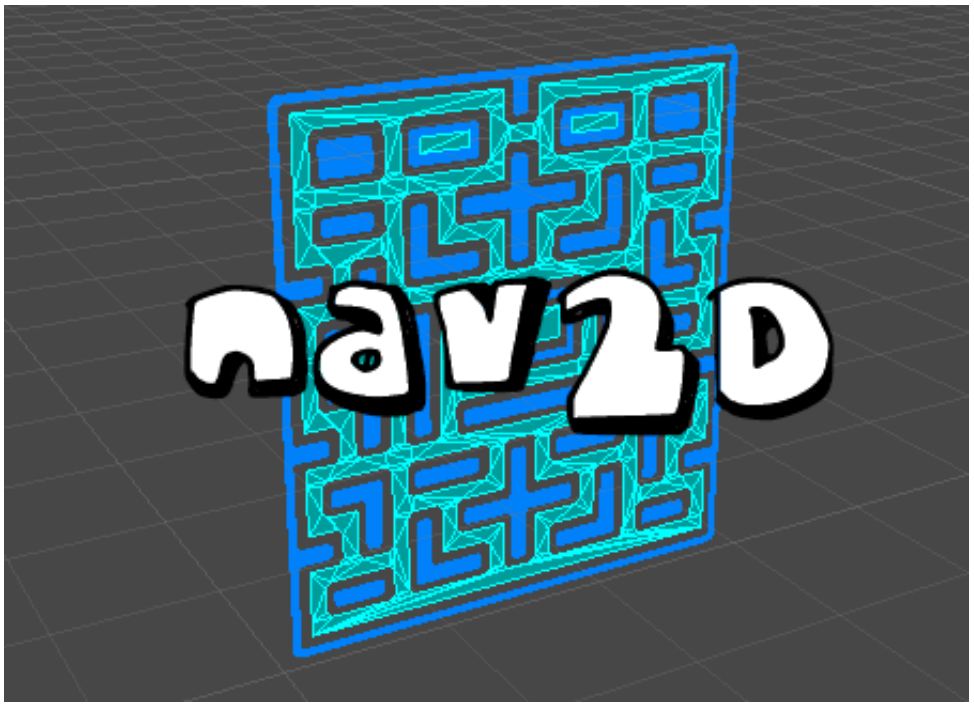
Navigation2D - Unity 2D Pathfinding



We are proud to present a new Editor Extension: **Navigation2D**.

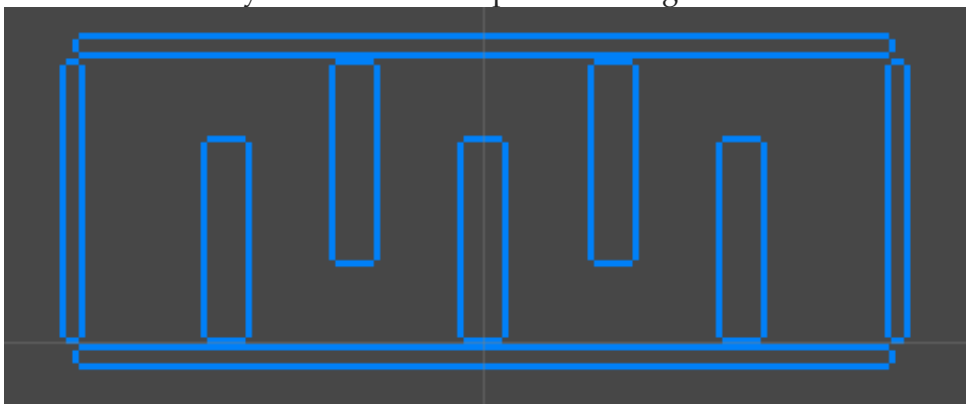
Unity Webplayer Demo: click here

How it works

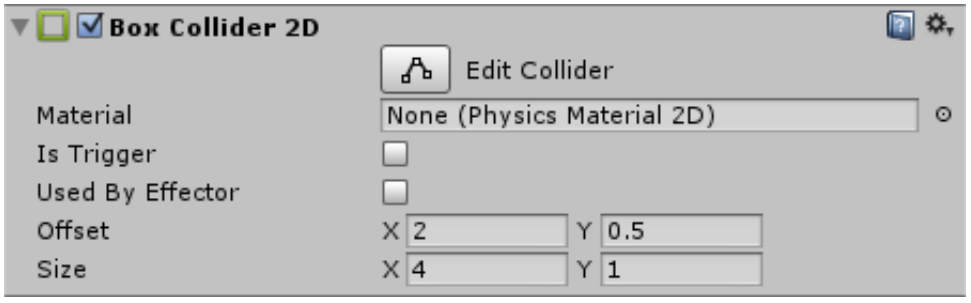
Navigation2D uses Unity's builtin Navigation system to make 2D Pathfinding possible without any axis rotations.

Usage Guide

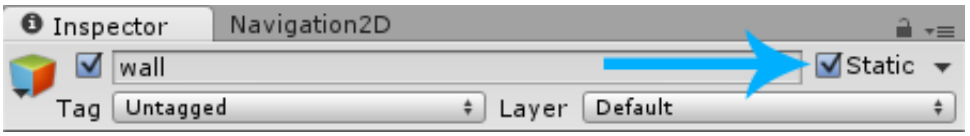
At first we add any amount of wall Sprites to our game:



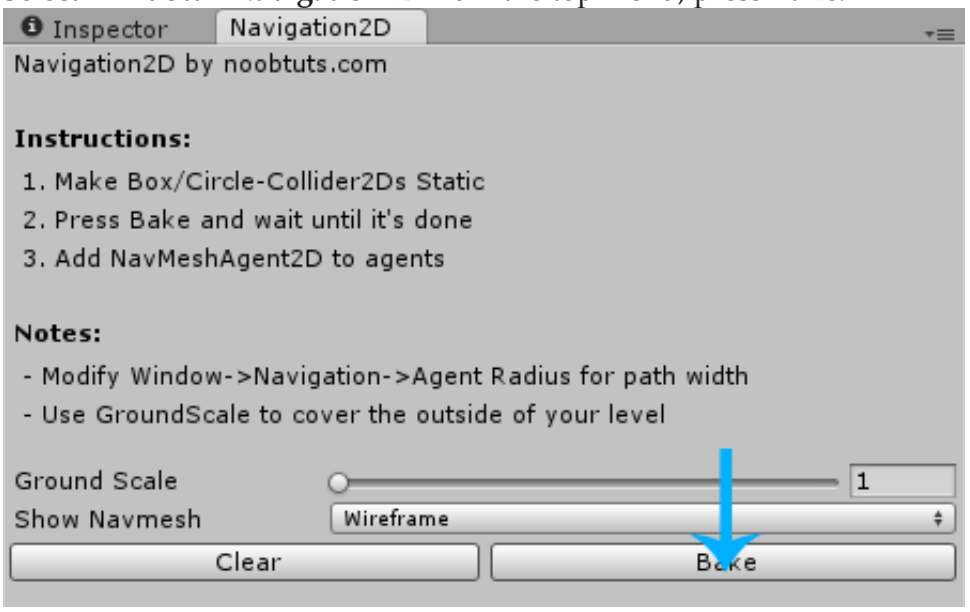
Afterwards we select all of them and add a **BoxCollider2D** or a**CircleCollider2D** to each one:



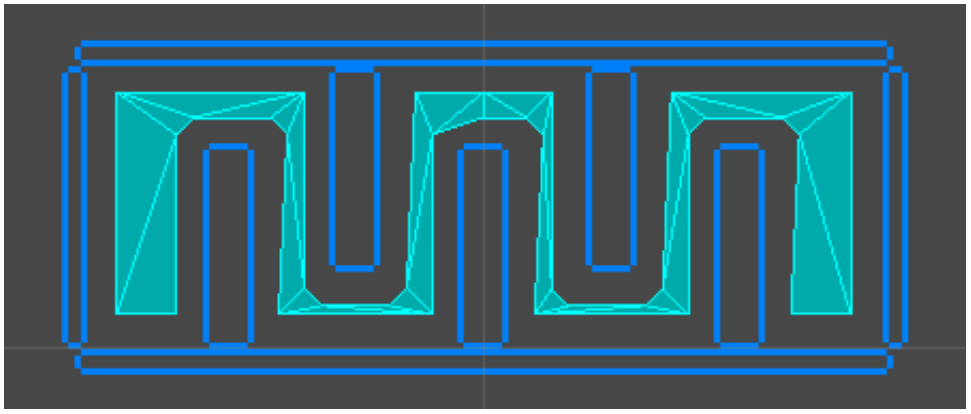
Now we make them all **Static**:



Select **Window->Navigation2D** from the top menu, press **Bake**:

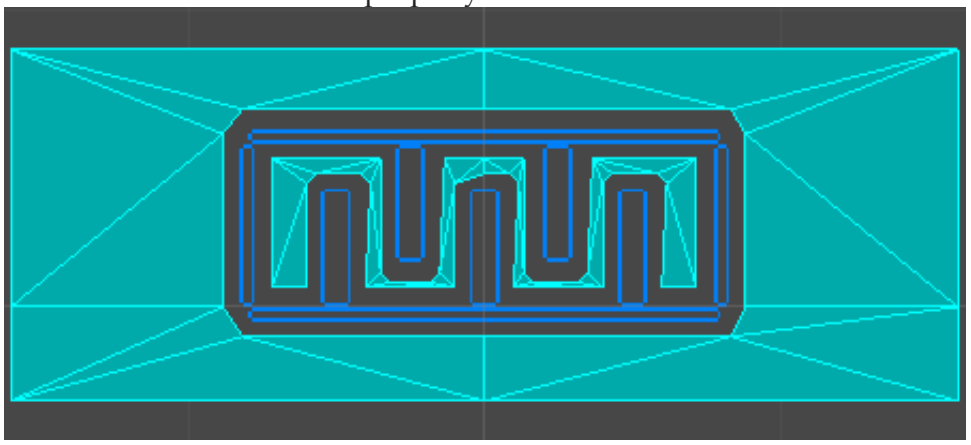


Afterwards we can see the **2D NavMesh** in the **Scene View**:

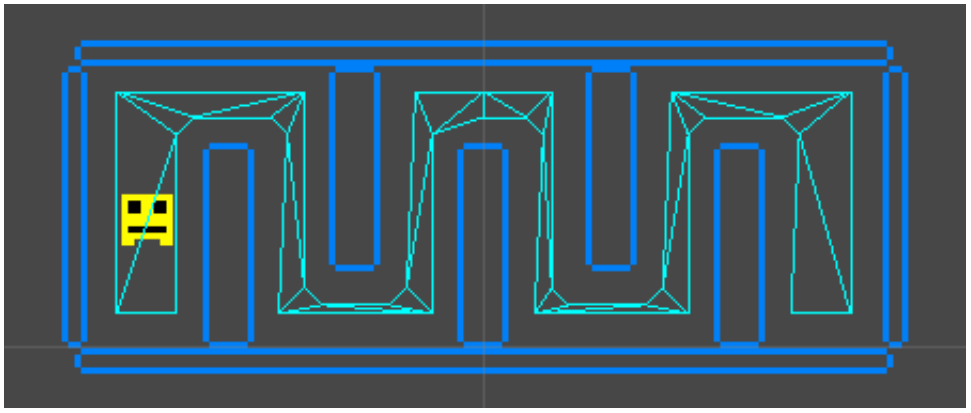


*Note: we can modify the width of the mesh by using the **Agent Radius**property in Unity's built-in **Navigation** settings.*

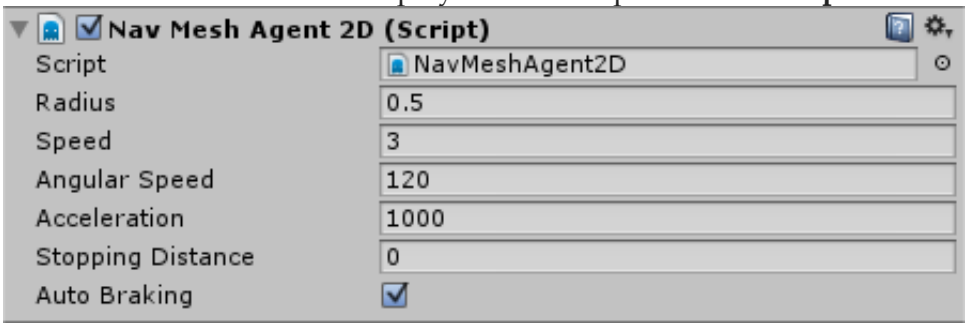
Increase the **Ground Scale** property if the NavMesh should also be outside of your level:



Add a player:



Afterwards we can select our player and then press **Add Component->Scripts->NavMeshAgent2D**:



Note: we can modify the Agent settings just like in Unity's built-in NavMeshAgent.

Finally we select **Add Component->New Script**, name it **Move** and select **CSharp** as the language. Our new Script will simply make use of the NavMeshAgent2D's destination property:

```
using UnityEngine;
using System.Collections;

public class Move : MonoBehaviour {

    void Update() {
        if (Input.GetMouseButton(0)) {
            Vector3 w = Camera.main.ScreenToWorldPoint(Input.mousePosition);
            GetComponent<NavMeshAgent2D>().destination = w;
        }
    }
}
```

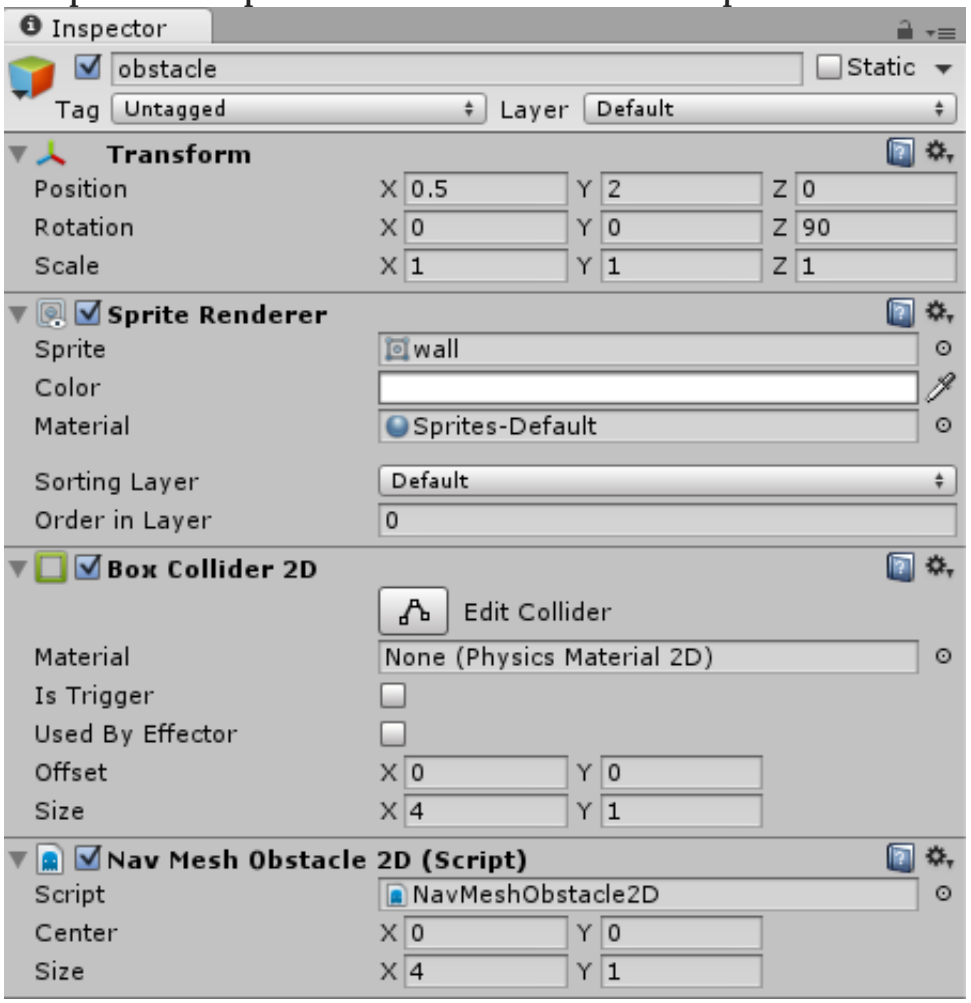
If we press **Play** then we can now click into the level to make the yellow guy move:

Advanced Features

Navigation2D supports a few more advanced features that we will cover here.

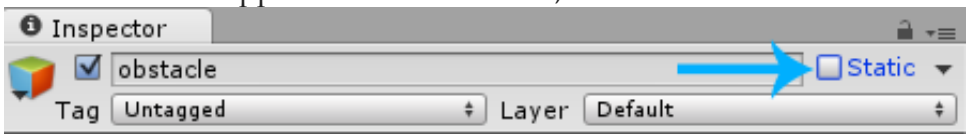
NavMeshObstacle2D

We can use NavMeshObstacle2D for obstacles that should block the navigation path, just like we would with Unity's built-in Navigation system. At first we select any of our wall sprites and then we select **Add Component->Script->NavMeshObstacle2D** in the **Inspector**:



Make sure to adjust the **Center** and **Size** properties until they fit our Sprite. If you already have a BoxCollider2D around your sprite, then simply use the same values that the collider uses.

Our obstacle is supposed to move around, so let's also make sure that it is **not** static:

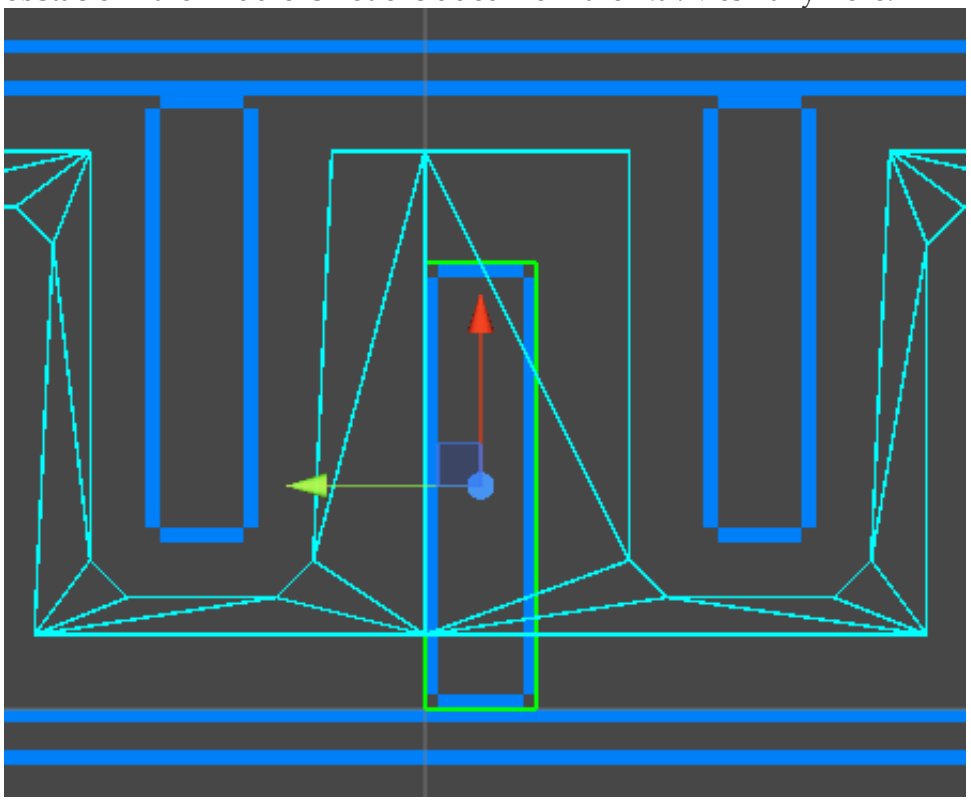


Let's also select **Add Component->New Script**. We will name it**MoveUpDown**, select **CSharp** as the language, open it and then add some simply code that makes our obstacle move upwards and then downwards all the time:

```
using UnityEngine;
using System.Collections;

public class MoveUpDown : MonoBehaviour {

    // Velocity
    public Vector2 velocity = Vector2.up;

    // Direction Change Interval
    public float interval = 1.5f;

    // Use this for initialization
    void Start () {
```

```
        // Change Direction every now and then
        InvokeRepeating("ChangeDir", interval, interval);
    }

    // Update is called once per frame
    void Update () {
        transform.position += (Vector3)velocity * Time.deltaTime;
    }

    void ChangeDir() {
        velocity = -velocity;
    }
}
```

Finally we go to **Window->Navigation 2D** again to rebake our NavMesh. Now we can see how the obstacle in the middle is not excluded from the NavMesh anymore:



Note: in other words, the area at the obstacle is now walkable by default, it only becomes unwalkable if our NavMeshObstacle2D moves there.

If we press **Play** then we can now see how the yellow guy navigates around our obstacle, no matter where it is: