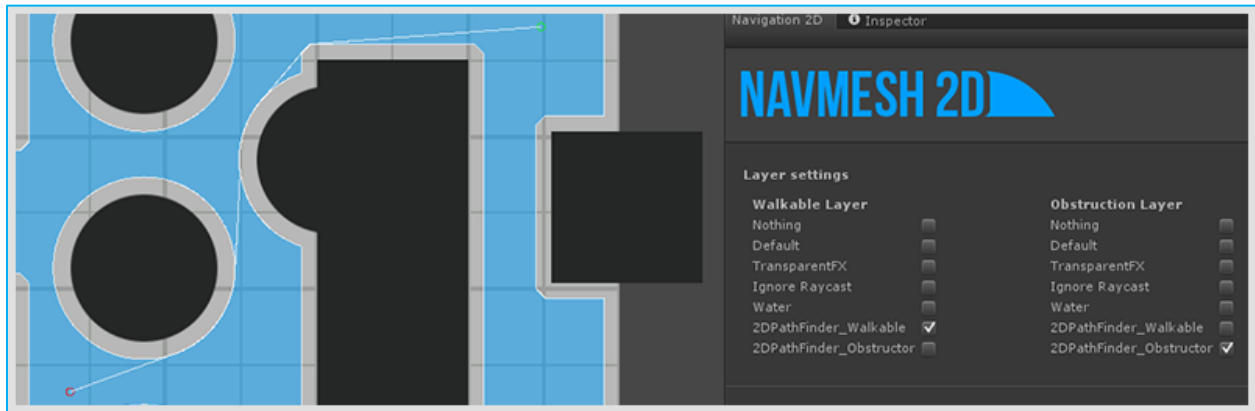


# Pigeon Coop Toolkit - NavMesh2D

---



NavMesh2D is a tool to generate and navigate navmeshes for 2D projects. It is designed to work very similarly to the built in Navigation tool which unfortunately doesn't yet work in 2D. With it, you can quickly add path finding to your game objects!

It is quick and easy to set up your navmesh with NavMesh2D. You can generate your navmesh in 3 steps - specify which layer contains your floors, specify which one contains your walls and hit bake.

## Contents

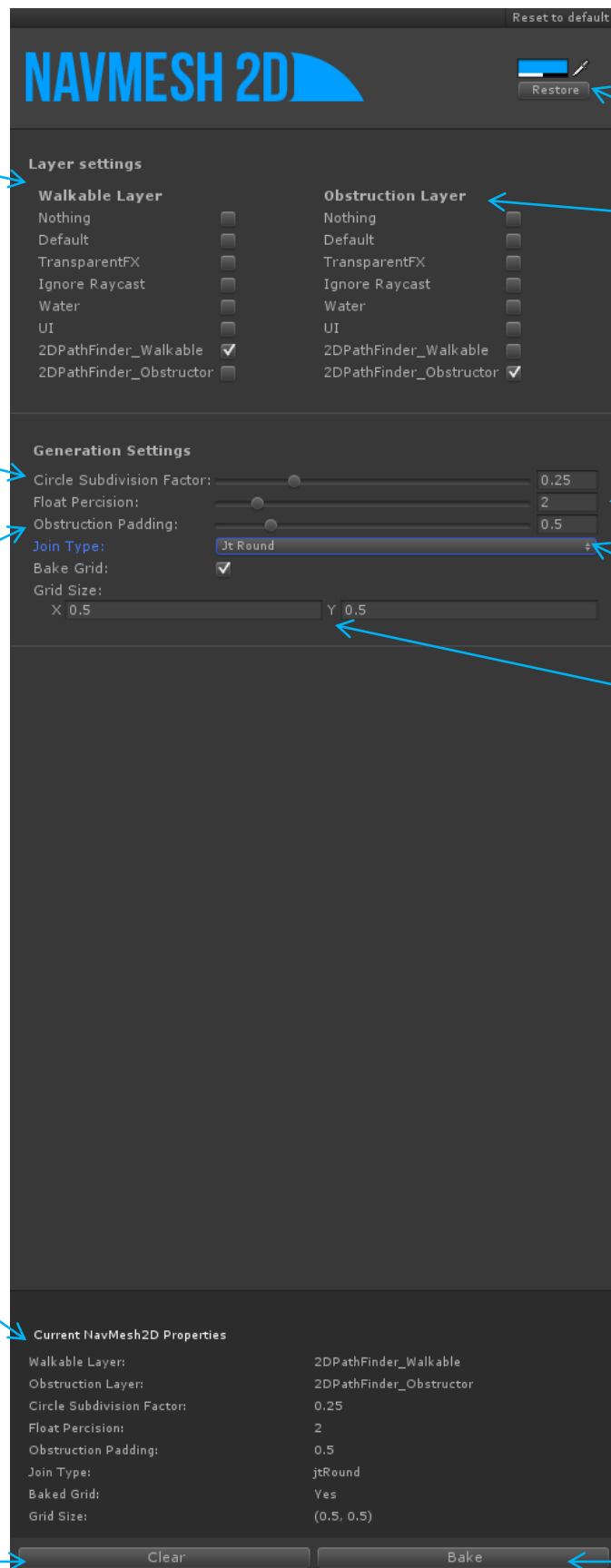
About the Editor.....	2
Useful Resources.....	2
Padding .....	4
Join Types.....	4
Grid.....	5
Walk me through generating a mesh! .....	6
Code Reference.....	10

## About the Editor

At this point, we'll assume that you've already purchased and imported the package into your project (Thank you!). So first things first, we need to open up the editor itself. You can access the editor through Windows > Pigeon Coop Toolkit > NavMesh 2D Generator. Woah! So many buttons and sliders, what do they all do? The next page has all the answers.

## Useful Resources

We've recorded a couple of videos and uploaded them to Youtube for you. [You can view them here](#), there is a 15 minute tutorial as well as a couple of videos explaining certain features.



Colliders on the walkable layer will add to the NavMesh

Resets all the properties below to the default values

Changes the color of the NavMesh in scene

Colliders on the obstruction layer will subtract from the NavMesh

Multiplier for subdividing circle colliders, higher values result in less subdivision

Number of decimal places for floating point operations. 2 is a good value, however if your colliders are really tiny you may need to increase this value to add more precision.

Size of the padding on the edges of your NavMesh.

Join type to use on the corners of your NavMesh (see below)

Size of the grid, if Bake Grid is enabled

The values that were used to generate the NavMesh in this scene (if any)

Completely clear all NavMesh information from this scene (if any)

Bake a NavMesh for this scene using the above settings

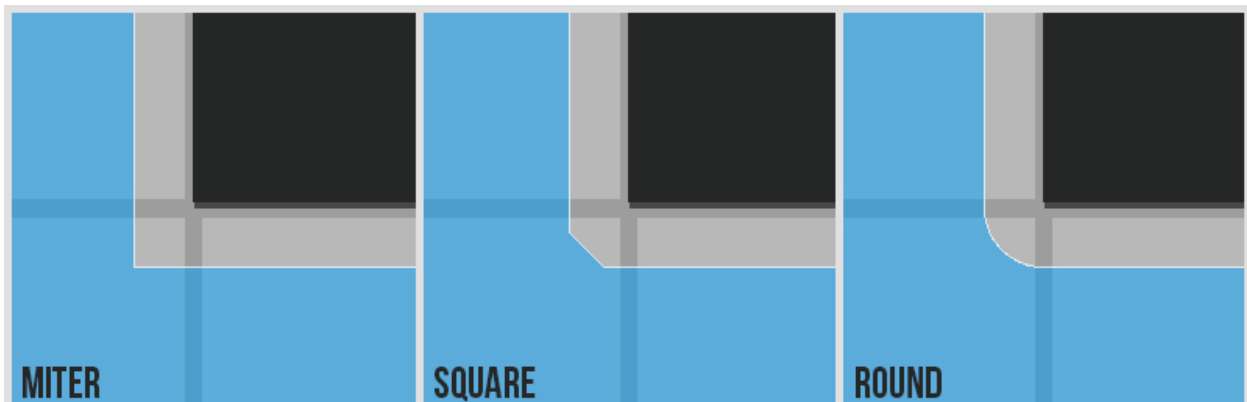
## Padding

You can create a buffer around the edges of your navmesh by changing the padding. This is a value specified in Unity units. You should make the padding at least as big as or bigger than the radius of the units that will navigate the mesh. If the radius of your units collider is larger than the padding, the unit will try the path through tight corridors that it cannot fit into and it will scrape along the wall when going around corners – not Ideal!



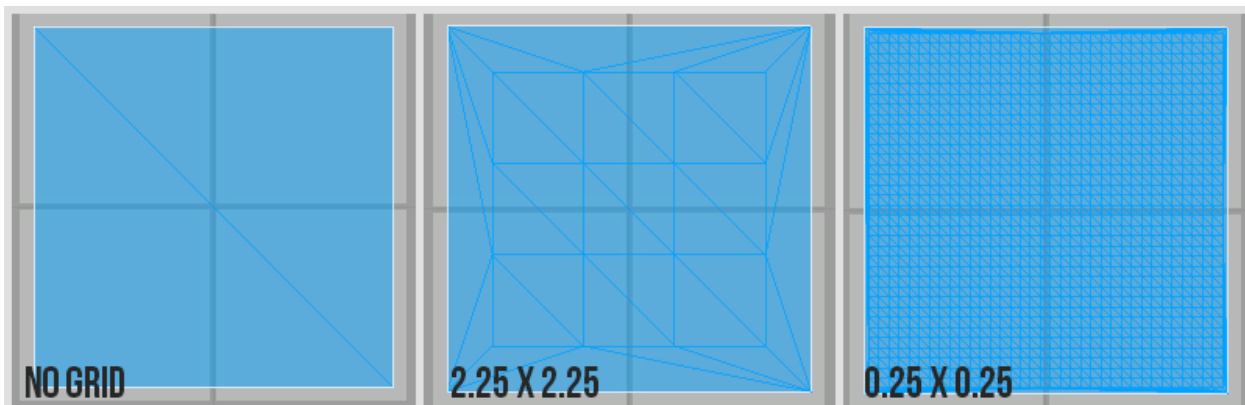
## Join Types

The join type is the type of join used on the corners of the NavMesh. Miter is your standard join type and should fit all your needs. However, navmeshes generated with Miter corners can cause a tiny bit of corner cutting for units that are navigating the navmesh. It is too tiny to really cause issues but if you want to avoid it you can use the Square join type which will square off the edge and give the path finding algorithm more points to work with so it doesn't have to cut corners. Lastly, we have the round corners. This gives pathing units a nice curved turn around edges at the cost of increasing navmesh complexity. You can usually get results that are just as good with Square edges, so do some experimenting before settling on round edges! The path finding algorithm will take a performance hit on these join types.



## Grid

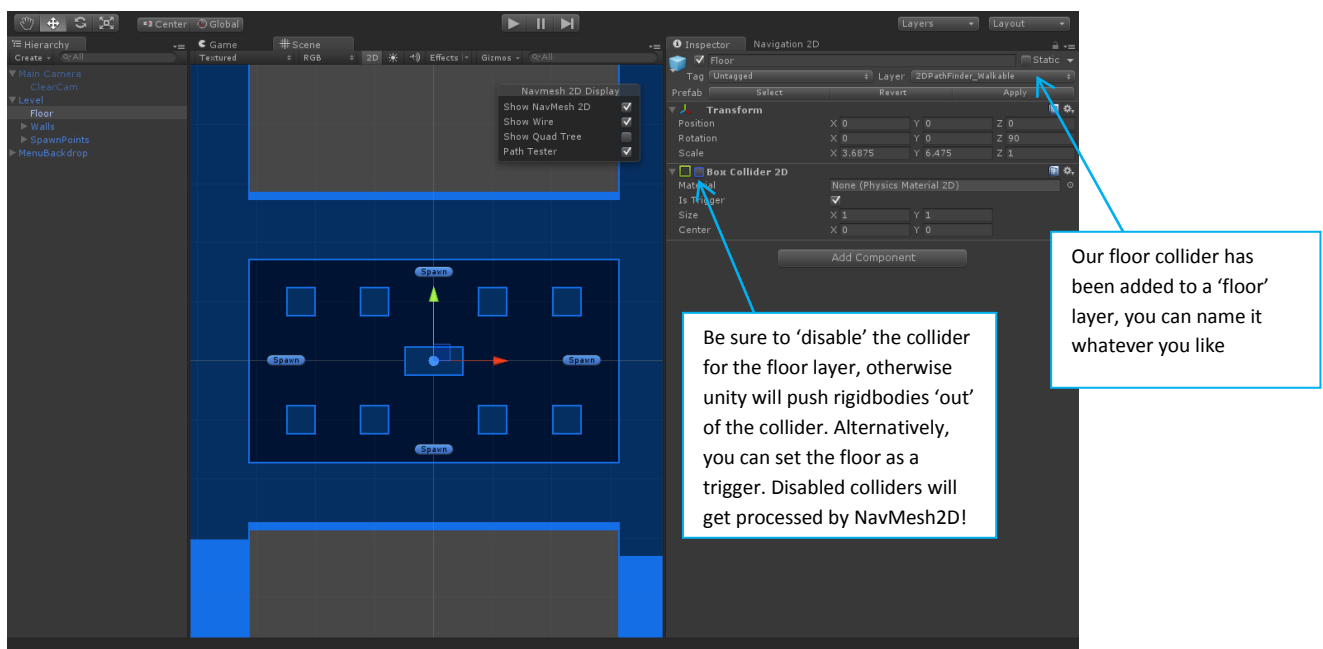
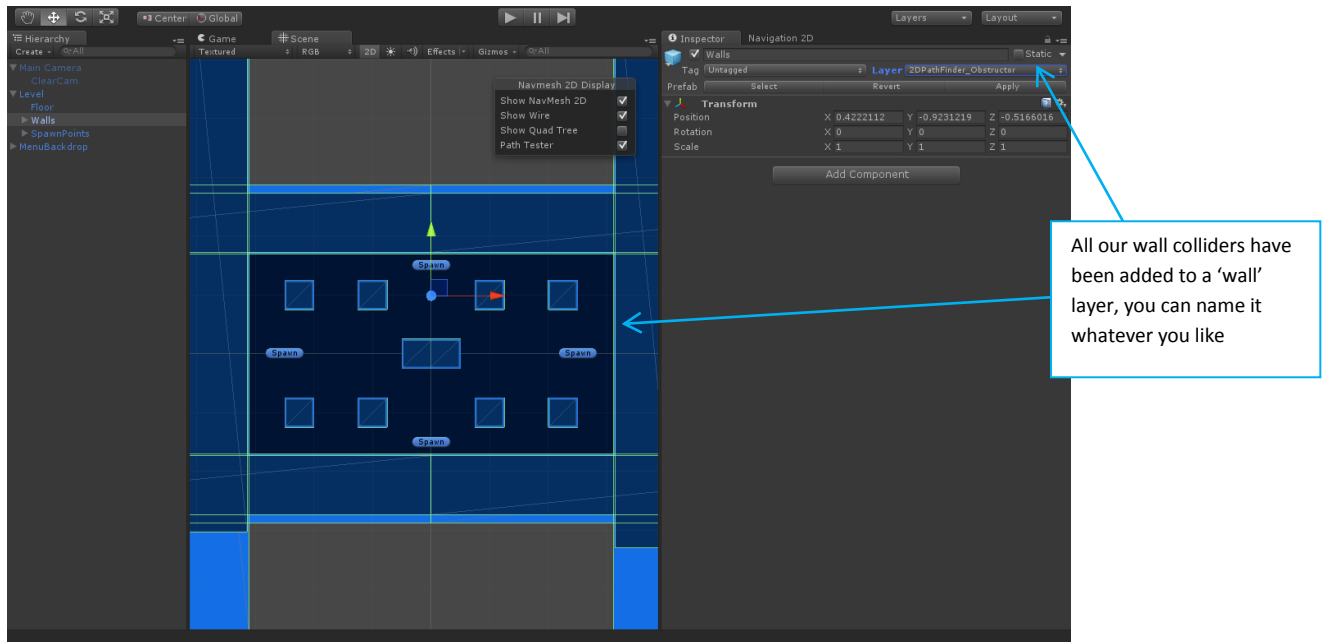
Grids are extremely helping for generating good paths your navmesh. You should click on “Show Wire” before playing around with the grid, so you can see the results of the generation. With no grid, the path finding algorithm (A star) does not have much information to determine the best path. It'll still give you a path, but your path may contain errors particularly around corners! This is not so much of a problem in levels with lots of small, tight corridors. However, in open areas the path finding will struggle to provide you with a good path. You need to bake a grid into your navmesh to assist the path finding. The density of the grid is up to you to decide. There is no ‘best value’. If you have lots of open areas and very little obstructions in the way, you may want to opt for a denser grid. If you have lots of small corridors and some small rooms, you might want to use a larger grid (or none at all). Because the grid increases the complexity of the NavMesh, the path finding will take a performance hit on really dense grids. It is up to you to find a nice balance since there is no ‘wrong’ answer. Whatever setting you choose (from no grid to super dense grids) the path finder will always return you a result.



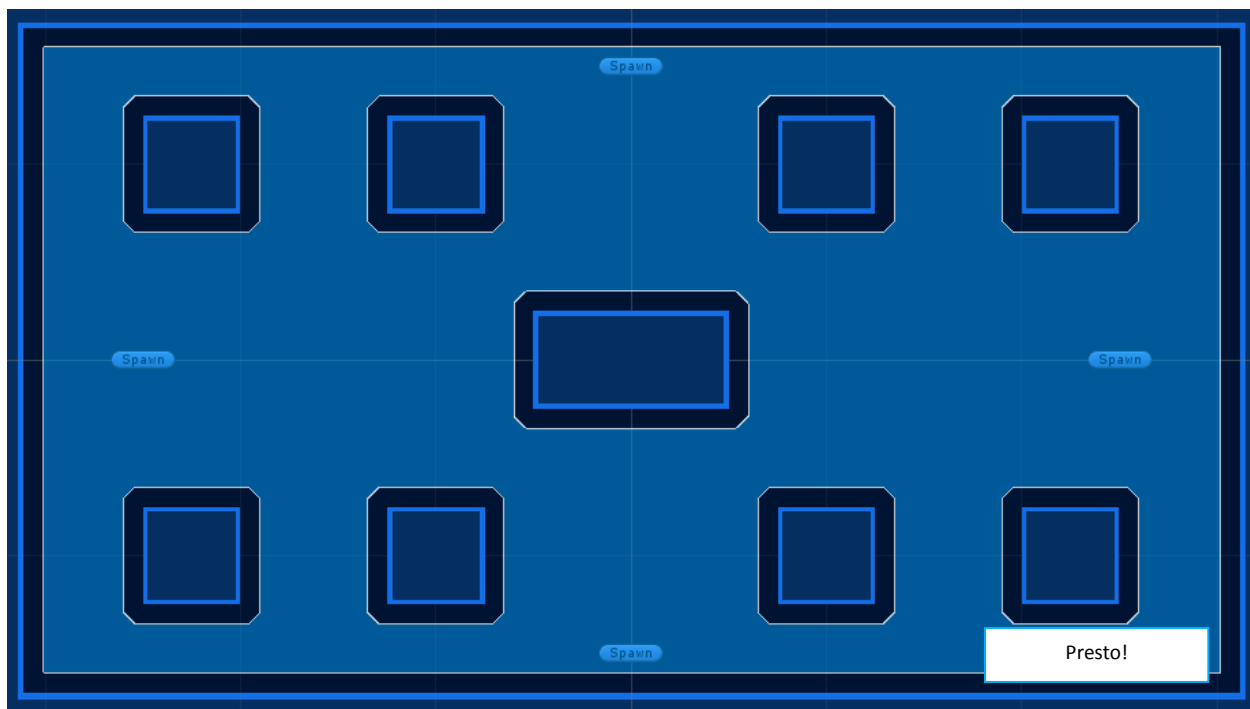
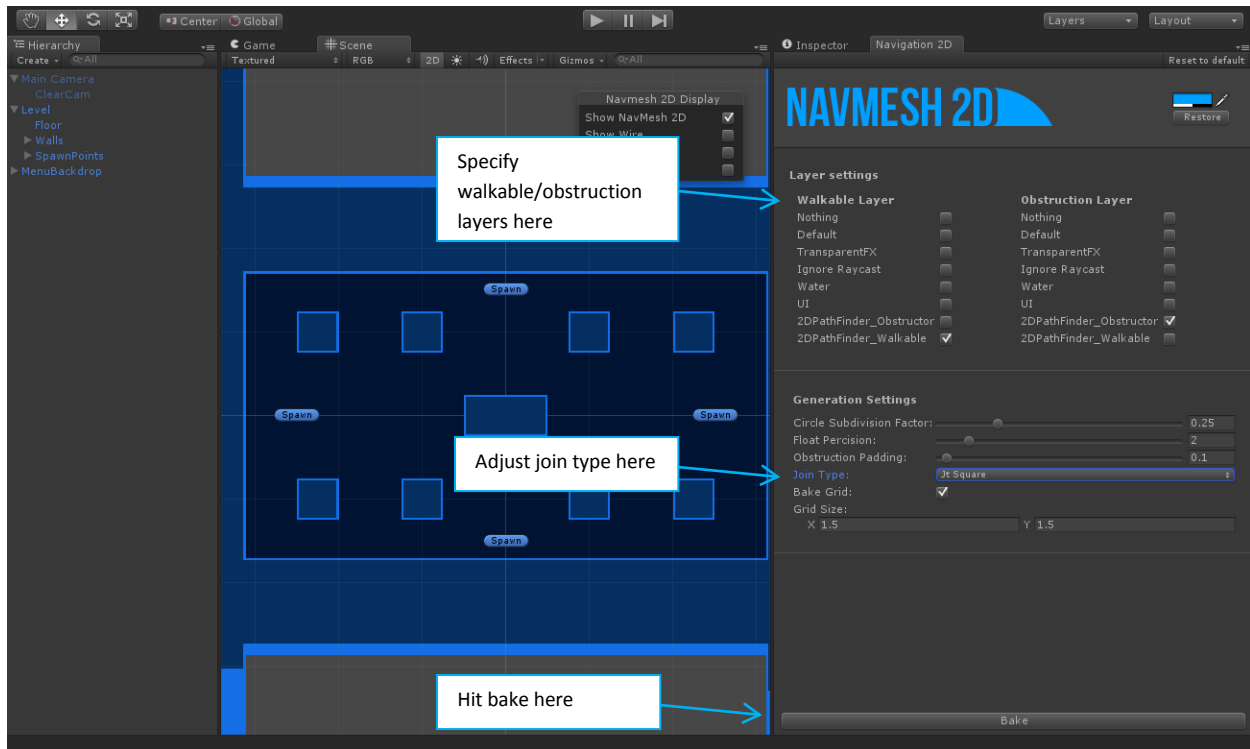
## Walk me through generating a mesh!

Ok, you've got it. We created this tool for our game Velocity Shift, and so we'll walk you through how we've gone about adding a navmesh to that project. So what do we need to do to get started?

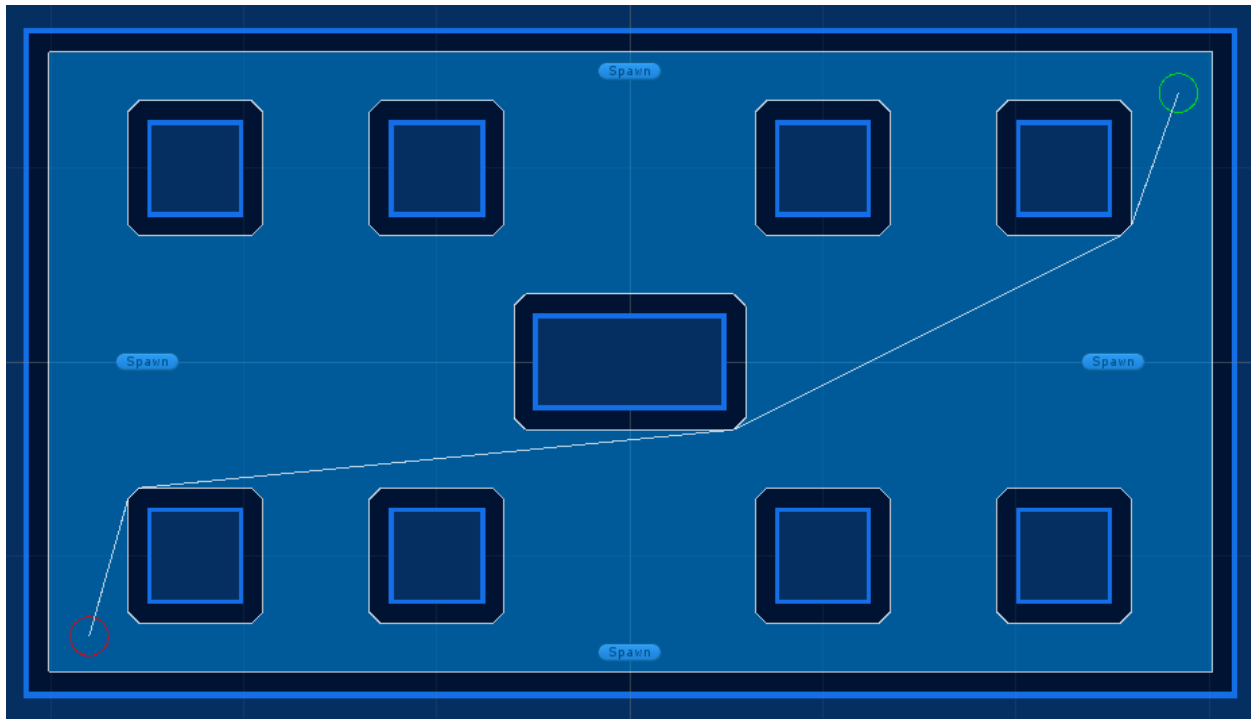
There is not too much preparation needed to ready your scene for NavMesh2D to generate a navmesh. You need to put all of the Collider2D's in your scene you consider to be a 'wall' into its own layer (You can specify whichever layer you like). You need to do the same with all the layers you consider floors. Now you're ready!



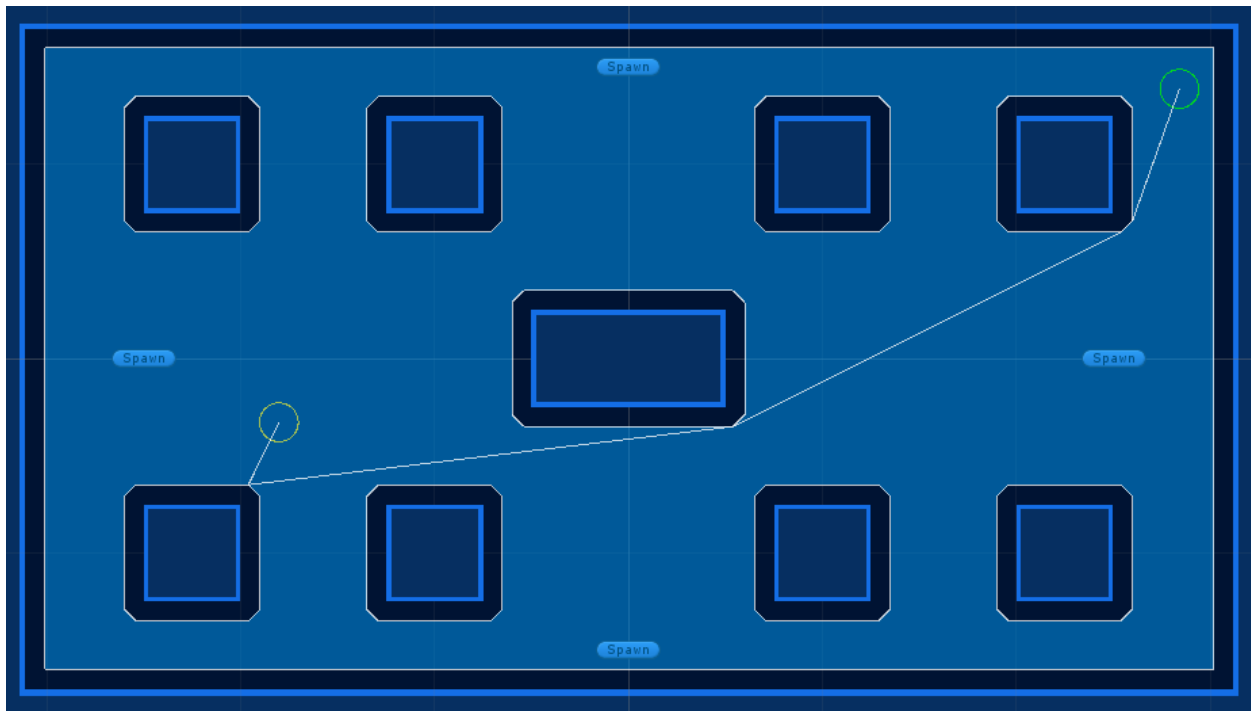
Now, open up the NavMesh2D editor. You can find it under *Windows>Pigeon Coop Toolkit>NavMesh 2D Generator*. In the editor window, tick the layers you've chosen as your floors and walls (Walkable/Obstruction). I am happy with all the default settings except for the Join Type. Let's change it to "Square" and hit "Bake".



In the utility window floating on the scene view, click on “Path Tester”. We can use this tool to test our path and decide whether or not we need to make alterations (Can the grid be less dense? Is the padding too big? Etc). Just drag around the green and red points anywhere on the NavMesh.

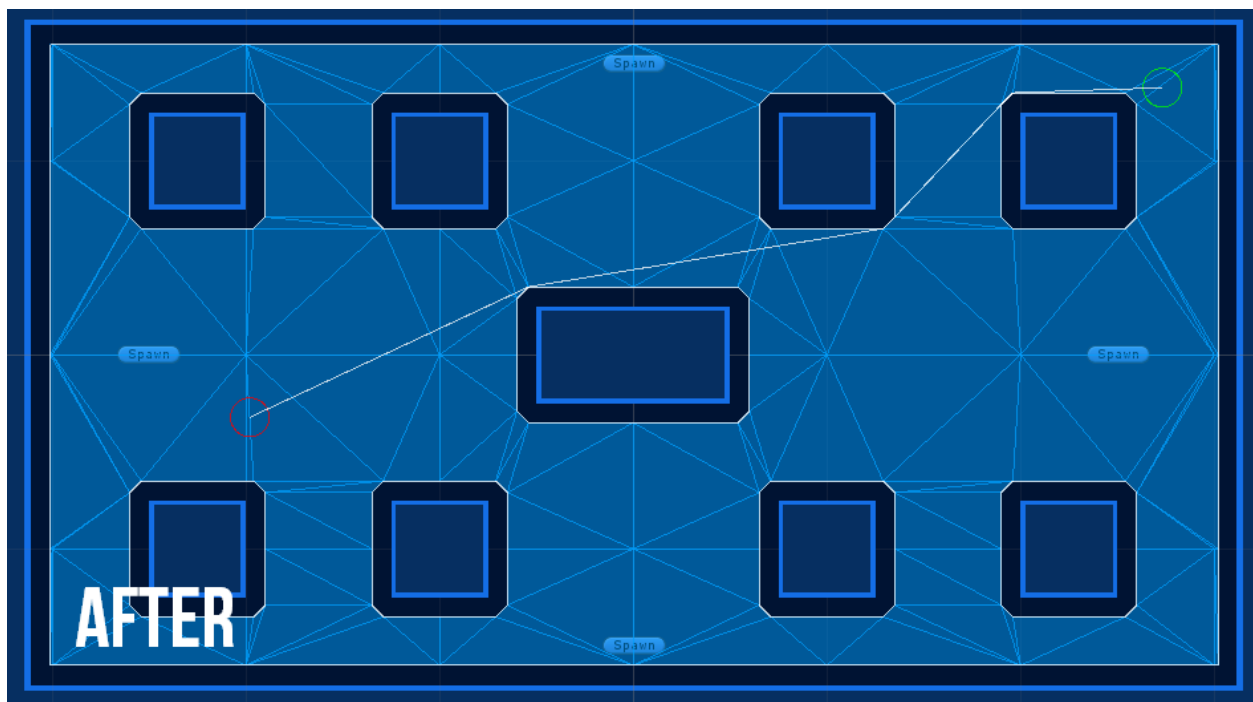
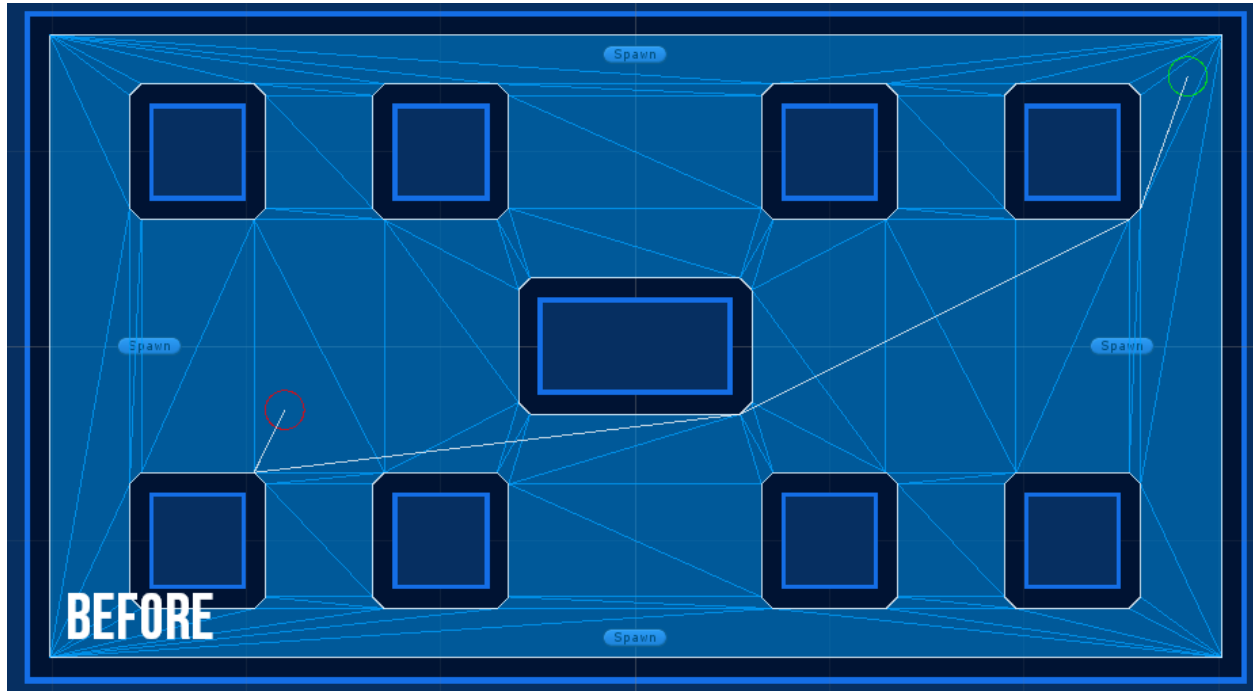


Here is an example of a problem that can occur when your navmesh is not dense enough.





This is a sign that your navmesh isn't detailed enough for the path finder to make accurate decisions. We can click on "Show Wire" to see the complexity of the NavMesh. The navmesh is pretty good for this level, but there are those 2 open areas where problems are occurring. We're going to bake in a 1x1 grid into the navmesh. The results are below – much better!



## Code Reference

`static List<Vector2> NavMesh2D.GetSmoothedPath(Vector2 start, Vector2 end)`

Returns a list of Vector2 points representing a smoothed path from start to end. The list is ordered from start to end.

`static List<Vector2> NavMesh2D.GetPath(Vector2 start, Vector2 end)`

Returns a list of Vector2 points representing an unsmoothed path from start to end. The list is ordered from start to end.

`static bool NavMesh2D.SceneHasNavmesh()`

Returns true if the scene contains a NavMesh.

`static NavMesh2DBehaviour NavMesh2D.GetNavMeshObject()`

Returns the actual NavMesh2D object that lives in the scene. You will probably never use this.