

# HOW TO

h8man edited this page Sep 23, 2022 · [23 revisions](#)

## NAV MESH PLUS

NavMesh Extensions System

by h8man

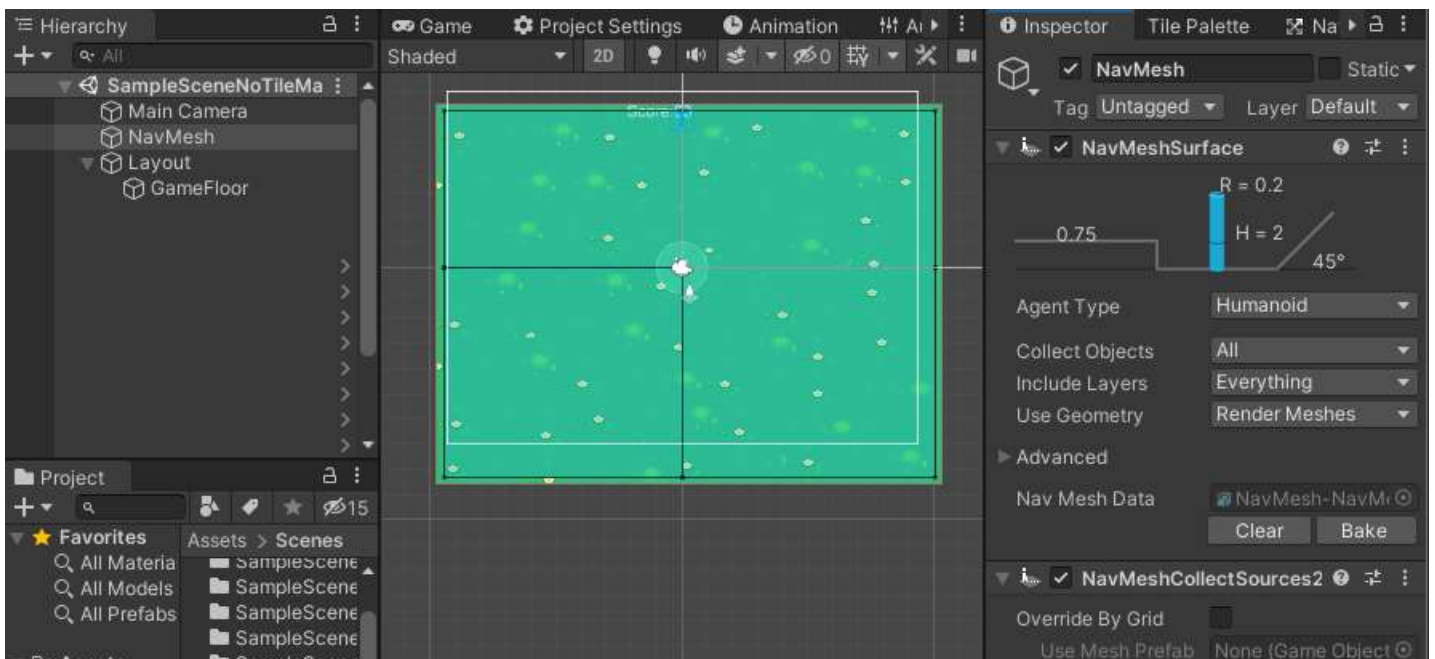
## INTRO

This short tutorial will focus on achieving the most basic functionality with NavMeshPlus - NavMesh Extensions System for 2D games, by using NavMeshCollectSources2d and NavMeshModifier extension components.

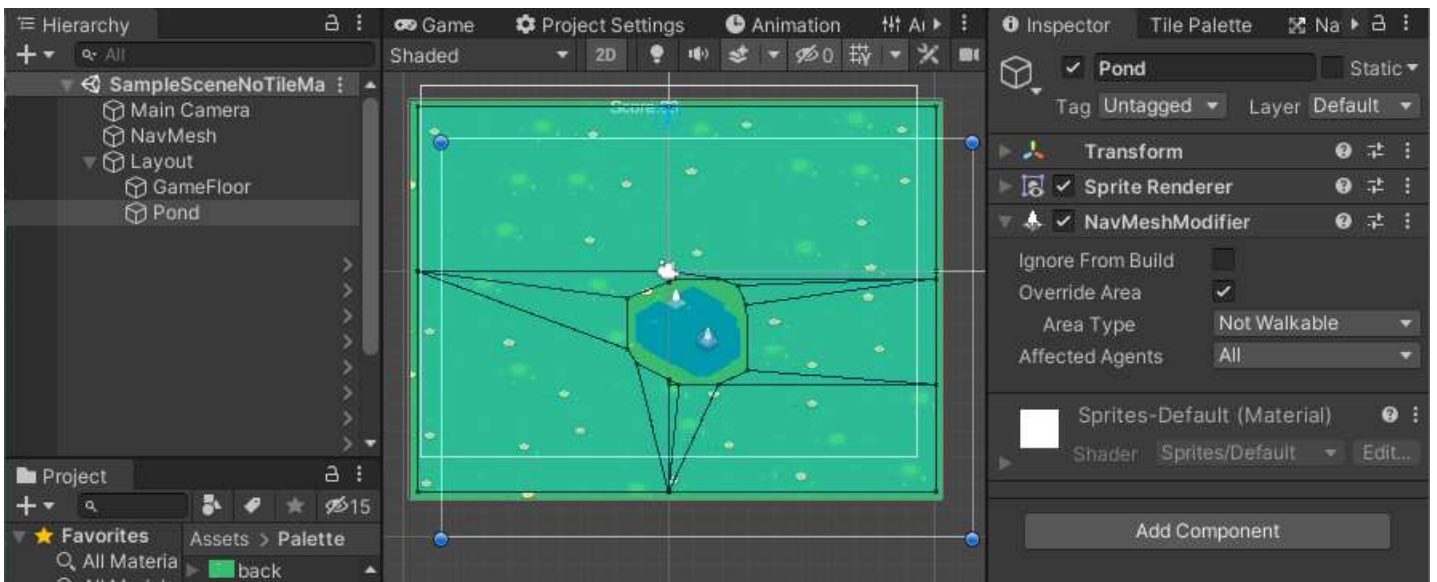
To begin with, Create a new project from Unity's "2D Template". Download NavMeshPlus and copy NavMeshComponents into your Assets folder. Now, in your scene, right click and select Create Empty from a menu to add an empty GameObject into scene's root. Name your new empty GameObject, for example as "NavMesh".

## NAV MESH BASICS

Create a new root GameObject named "Layout" where we will store our level layout and add child GameObject as "GameFloor". Drop any sprite onto "GameFloor", SpriteRenderer component will be automatically added, now add NavMeshModifier component to the object to mark it as navmesh source. With your "NavMesh" object selected, in the inspector click Add Component and choose NavMeshSurface. If you hit bake nothing will happen, cause we need to add another component NavMeshCollectSources2d, click Rotate Surface to XY and then Bake to bake walkable plain.



Now add another object to "Layout", call it "Pond", drop new smaller sprite and similarly click Add Component, choose a NavMeshModifier component. In the NavMeshModifier component check Override Area and in the Area Type field choose Not walkable. Now hit Bake again (within the "NavMesh" object). You should see that the "Pond" have been carved out of the navmesh areas.



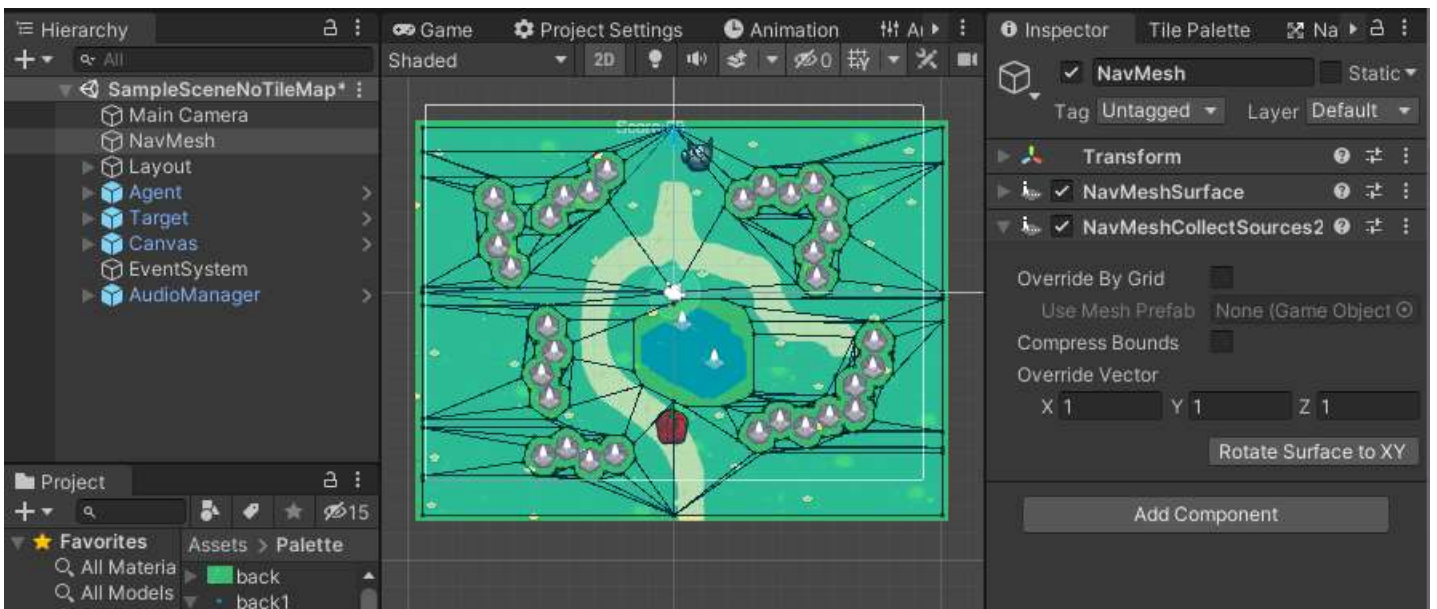
At this point, you can add GameObjects with a NavMeshAgent component, which are capable of interacting with (i.e., walking on) your NavMesh.

By default NavMeshAgent component tends to rotate the GameObject as soon as you hit play. This can be undesirable (as in, your object may be rotate away from your plane of view and hence become invisible), so you should fix its rotation by adding something like the following to a new or existing script:

```
void Start() {
    var agent = GetComponent<NavMeshAgent>();
    agent.updateRotation = false;
    agent.updateUpAxis = false;
}
```



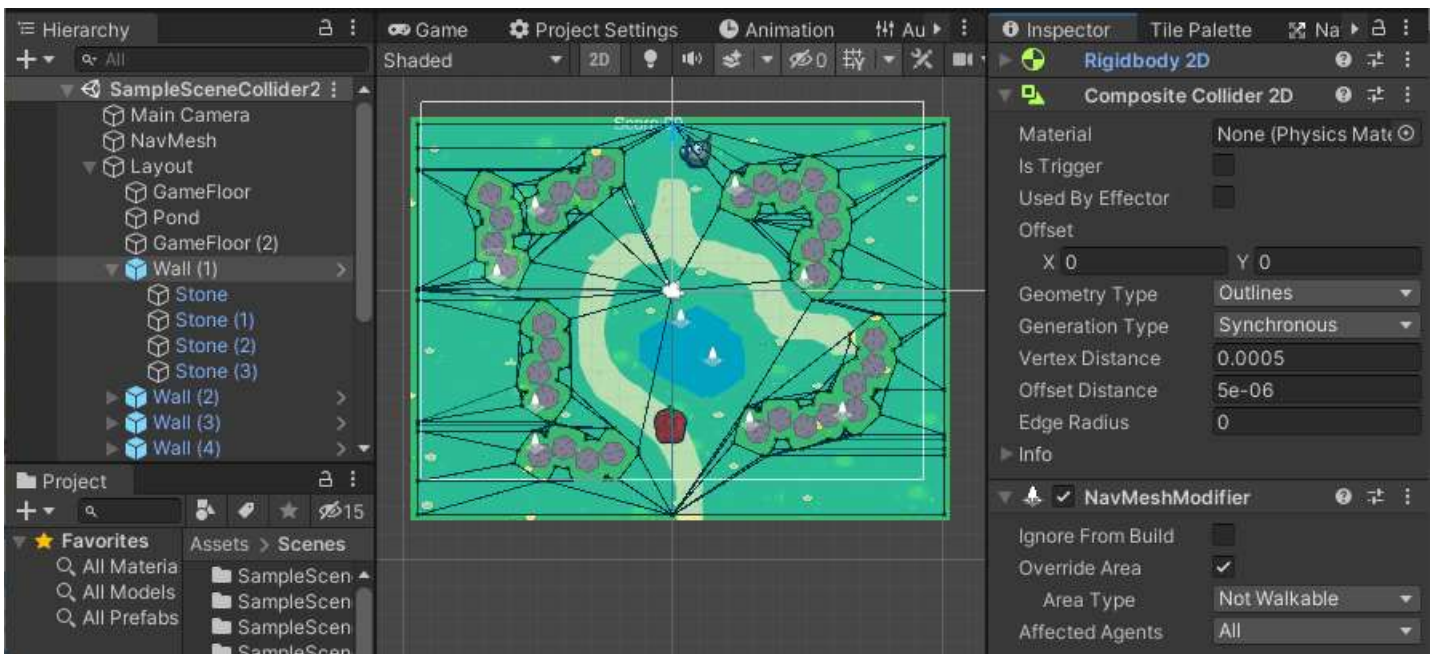
Once this is added to the start method for all of your agents, you are ready for action.



**Important note:** Currently there is no way to define world bound except to introduce explicit world bound GameObject with NavMeshModifier

## GEOMETRY OPTIONS

With new API introduced in Unity 2019.3 there is an ability to choose which geometry source to use. Place all 2D colliders with NavMeshModifier added into "Layout" GameObject as child, change Use Geometry option from Render Meshes to Physics Colliders and hit bake.

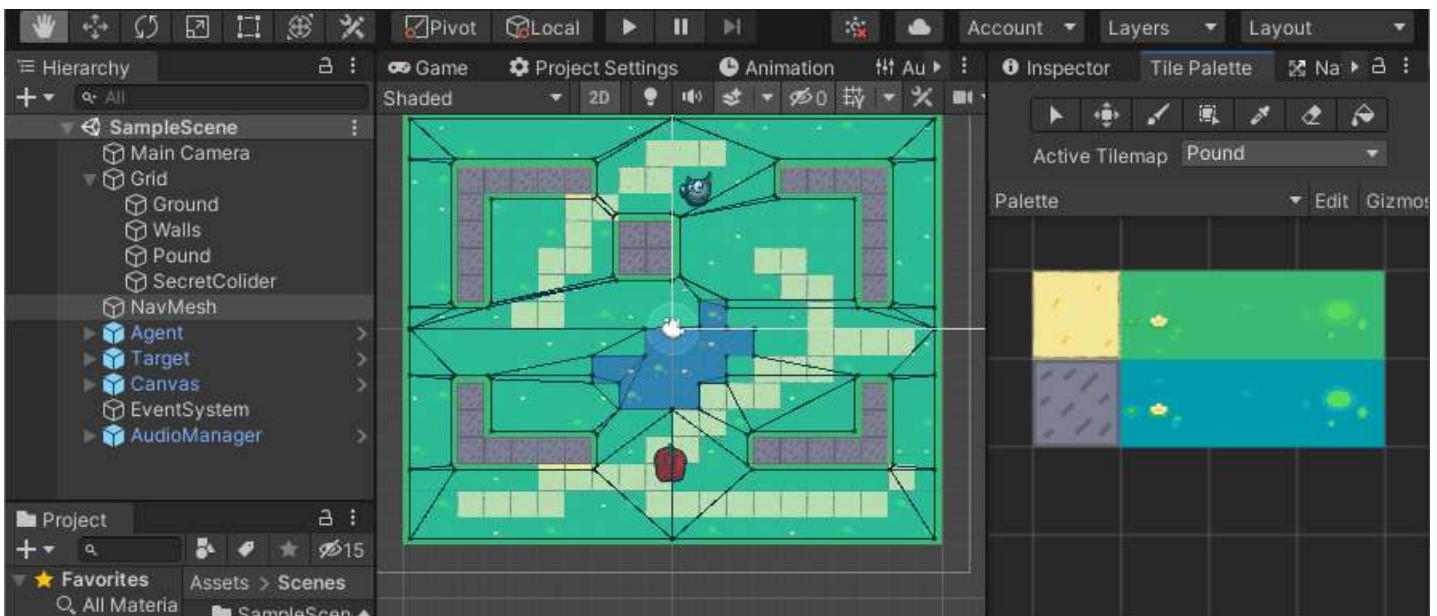


**Important note:** By default Use Geometry option is set to Render Meshes, which mean NavMeshSurface use sprite's Mesh to build its navigation surface, so surface will stick to sprite "outline". The other option available is Physics Colliders, this option allows navmesh to build surface form 2D colliders polygons. Options are mutually exclusive.

## 🏠 TILED

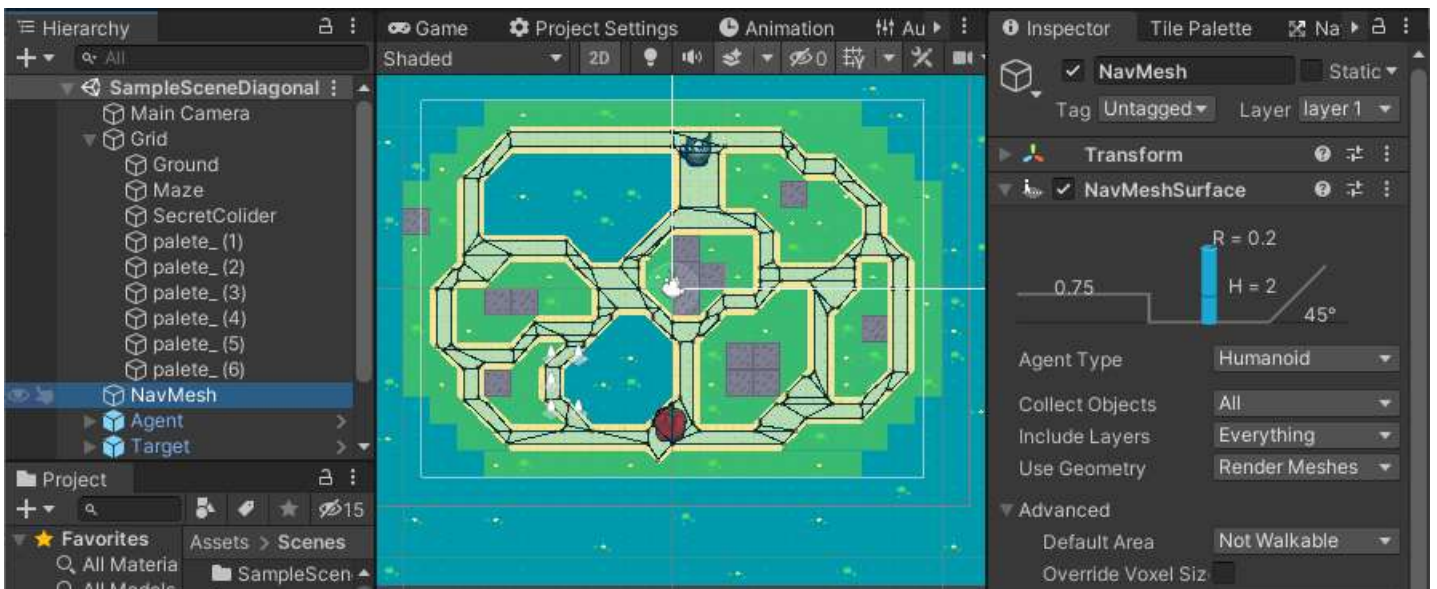
With NavMeshCollectSources2d you can bake objects that are a part of Tilemap system. Add a Tilemap GameObject to your scene (By right-clicking your Scene hierarchy and choosing the 2D Object folder, and then Tilemap within). It should create a Grid object in the root of your scene and child Tilemap game object. Add two more Tilemaps (for a total of 3) and name them "Ground", "Walls" and "Pond".

Using sprites imported into your palette, choose "Ground" as the Active Tile Map, and paint the ground in your scene view. Change Active Tile Map in your palette to "Walls" to paint some obstacles. Then select "Pond" and do the same, drawing a body of water. As with basics navmesh add NavMeshModifier to each of tilemaps, override its areas to walkable and not-walkable respectively. Bake the navmesh.



## 🏠 MAZES

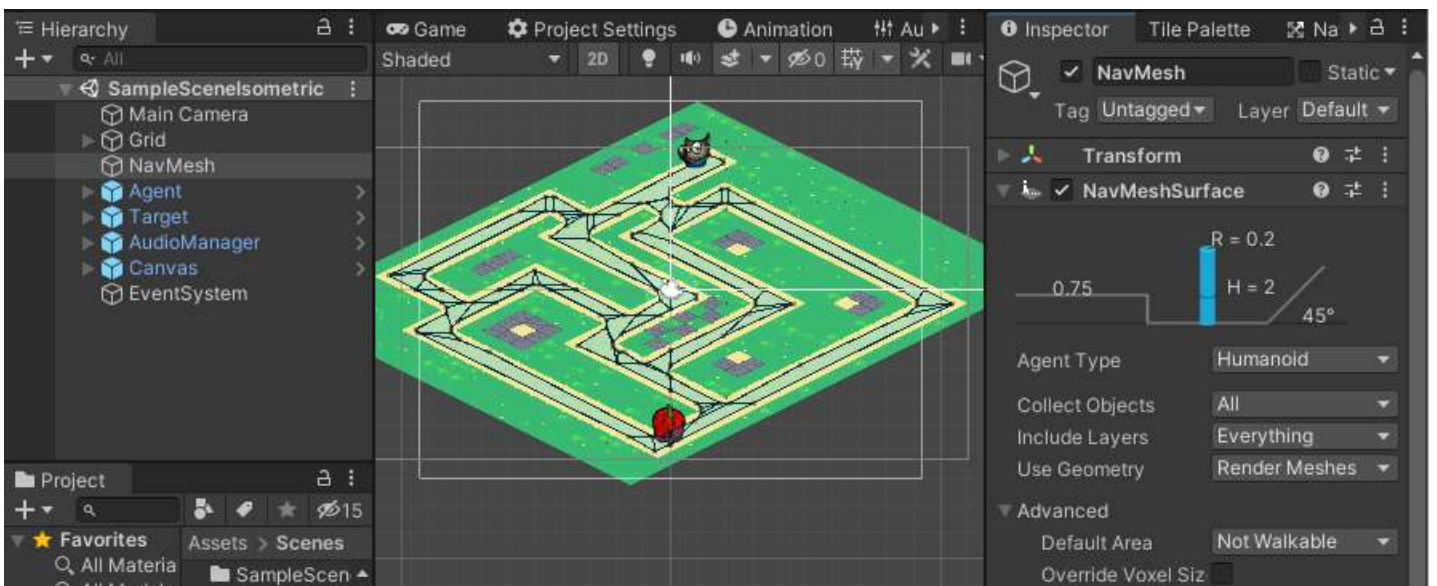
Another common task is not to carve out obstacles, but to navigate through a maze, where our agents can only follow a strict path. To do that, Bake your "NavMesh" with Default Area selected to Not walkable, and the NavMeshModifier component set to be walkable, and you will get this:



Any sprite within the Grid object with a NavMeshModifier will be added into Bake process. (Sprite must have a mesh). You can build bridges or barricades, and these objects will be added into your NavMesh as static data.

## OTHER GRIDS

Some games want to use hexagonal, isometric, or diamond-shaped grids. NavMeshPlus supports baking other types of grid as well, as long as sprites share the shape as your grid.

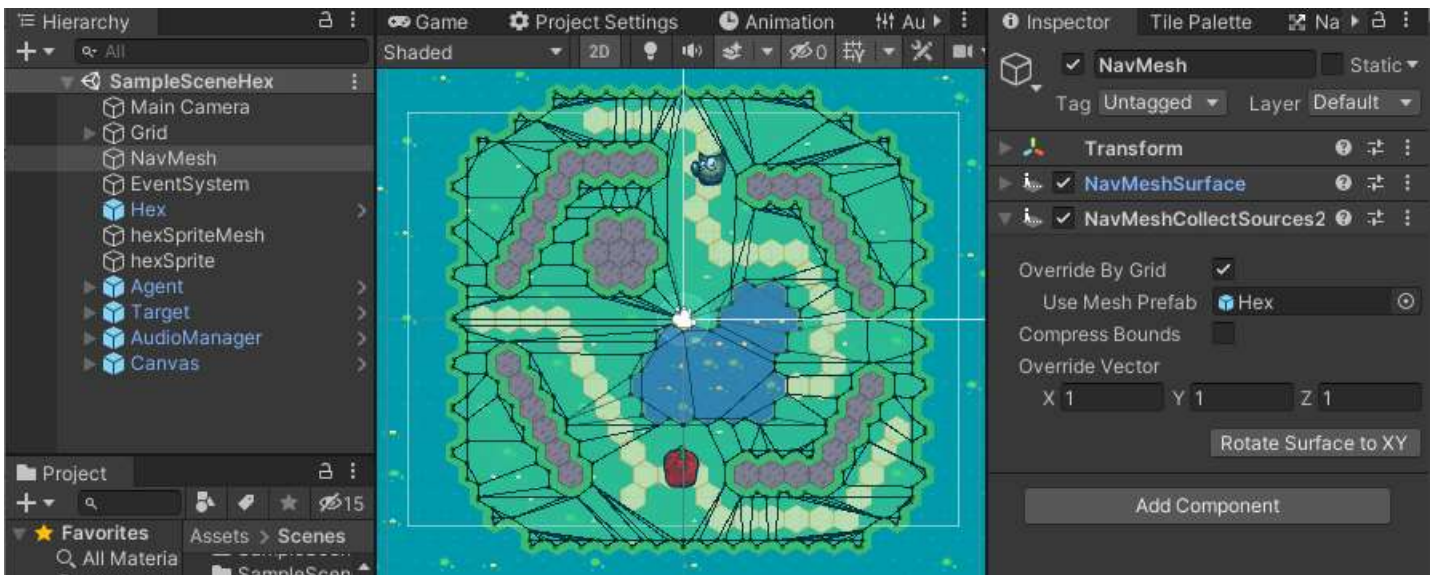


## OVERVERRIDE BY

In some cases the sprite can differ from your grid shape, it may have excessive outline or be concave. When we want clean navigation though the tilemap we need clean to up hundred sprites of different shapes. To streamline tedious editing of each sprites outline we can override sprite mesh altogether.

First we need to have a mesh that has the shape of a tile. For this hexagonal tile example, we'll import a mesh shaped like a hexagon. (Use Blender or any other 3D software)

Add your hexagon to the scene, rotate it to face the camera and scale it down to match the size of your cells. In the NavMeshCollectSources2d component, check Override By Grid and add prefab. Hit Bake.



**Important Note:** If we remove the prefab and leave the override option active, all tiles will be considered as squares. Which is great for performance.

## BAKE AT RUNTIME

Often games have procedural world building which requires bake NavMesh at runtime. To achieve this you need to call a single method `BuildNavMeshAsync` in `NavMeshSurface` component to bake NavMesh at runtime in Unity async manner. This method can be called on `Start()` method.

```
public NavMeshSurface Surface2D;

void Start()
{
    Surface2D.BuildNavMeshAsync();
}
```



For dynamic worlds to save some CPU utilization we can call `UpdateNavMesh` instead of building NavMesh every frame. Code may look like:

```
public NavMeshSurface Surface2D;

void Update()
{
    Surface2D.UpdateNavMesh(Surface2D.navMeshData);
}
```

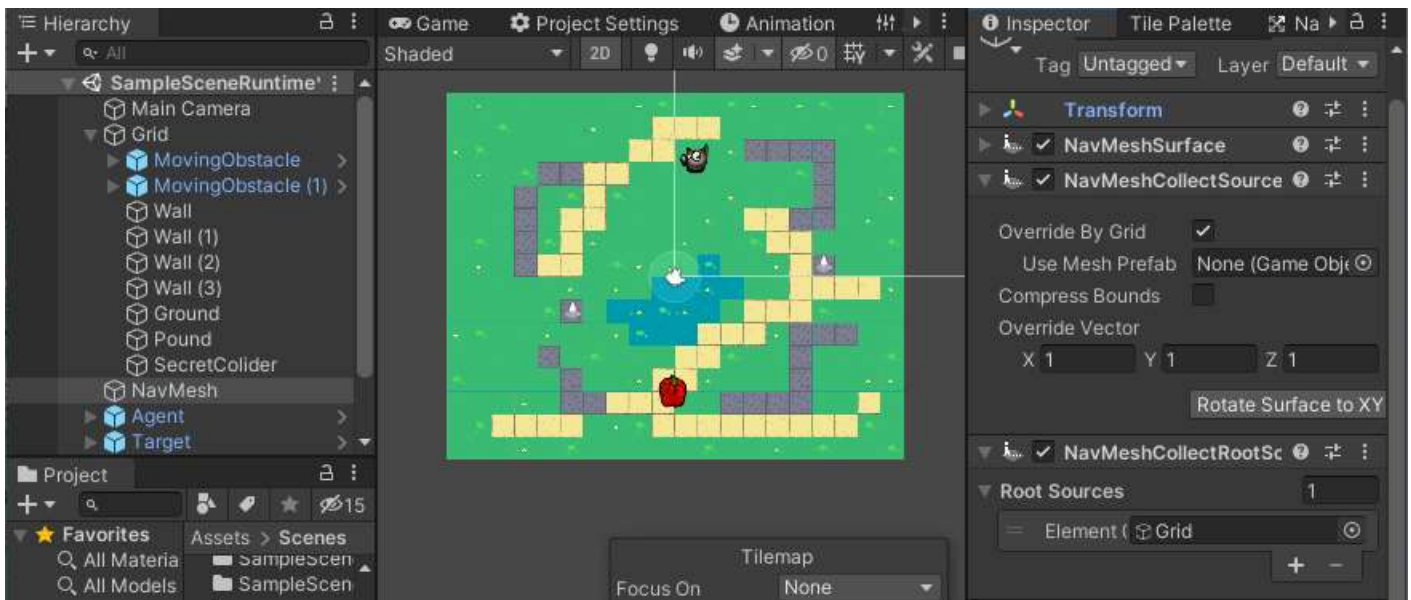


But it does not cache collected 2D sources between updates, so you might use its internal method with your caching mechanism:

```
NavMeshBuilder.UpdateNavMeshDataAsync(data, GetBuildSettings(), sources, sourcesBounds);
```



Also to save some CPU cycles consider to use `NavMeshCollectRootSources2d` to specify navmesh root sources explicitly.



**Important note:** Collider's bounds not getting updated to match the transform until 'LateUpdate()' is called. Force the update by calling `Physics2D.SyncTransforms()` before calling 'BuildNavMesh()' if you baking navmesh at runtime.

**Important note:** Physics updated only in fixed update, that is running in different time period than regular update. So its is not possible to generate physics in Awake/Start and call 'BuildNavMesh()' before 'LateUpdate()' to bake new geometry at runtime.

More samples can be found in "NavMeshComponents" repository on [GitHub](#) and NavMeshBuilder methods documentation at [Unity](#) site

## 🔗KNOWN ISSUES

The agent gets stuck while moving on Y axes, or setting agent. Velocity doesn't do anything if  $x=0$  - it seems to be a bug in NavMeshAgent component. There is no fix available right now, but there are workarounds. The idea of workaround is simple, adjust Agent movement on X so its never straight up. By default navmesh now have a small tilt 89.98 to make agent work. If for some reason its doesnt solve the problem, you need ad a drift. Here is a snippet for SetSestination():

```
void SetDestination(GameObject target)
{
    var agentDrift = 0.0001f; // minimal
    var driftPos = target.transform.position + (Vector3)(agentDrift * Random.insideUnitCircle);
    agent.SetDestination(driftPos);
}
```



## 🔗THE END

Try out a fully-featured 2D "run-and-chase" game I made to demo NavMesh 2D: [\[RedHotSweetPepper\]](#).



References:

1. NavMeshSurface - <https://docs.unity3d.com/Manual/class-NavMeshSurface.html>
2. NavMeshSurface Extensions - <https://github.com/h8man/NavMeshPlus>
3. The Game Demo - <https://github.com/h8man/RedHotSweetPepper>
4. Forum to discuss - <https://forum.unity.com/threads/2d-navmesh-pathfinding.503596/>
5. NavMeshComponents Runtime - <https://github.com/Unity-Technologies/NavMeshComponents/tree/master/Assets/Examples/Scenes>