NAVMESH PLUS

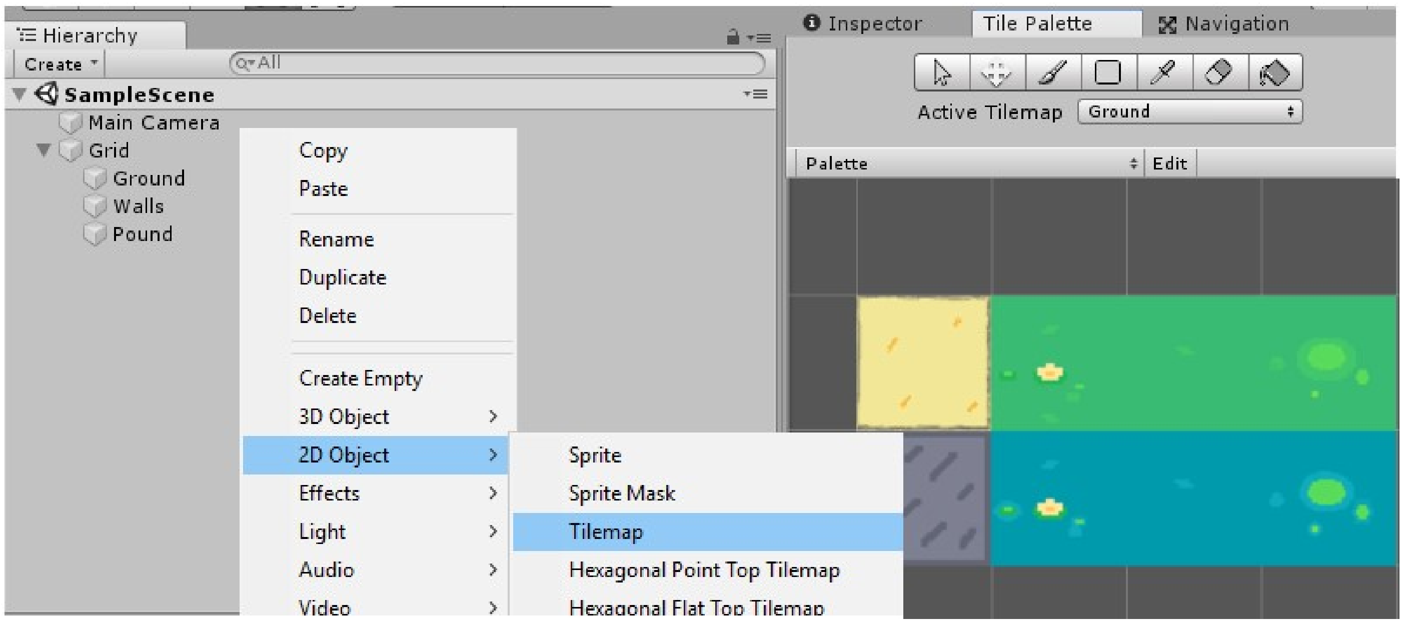
by h8man

INTRO

This short tutorial will focus on achieving the most basic functionality with `NavMeshPlus` .

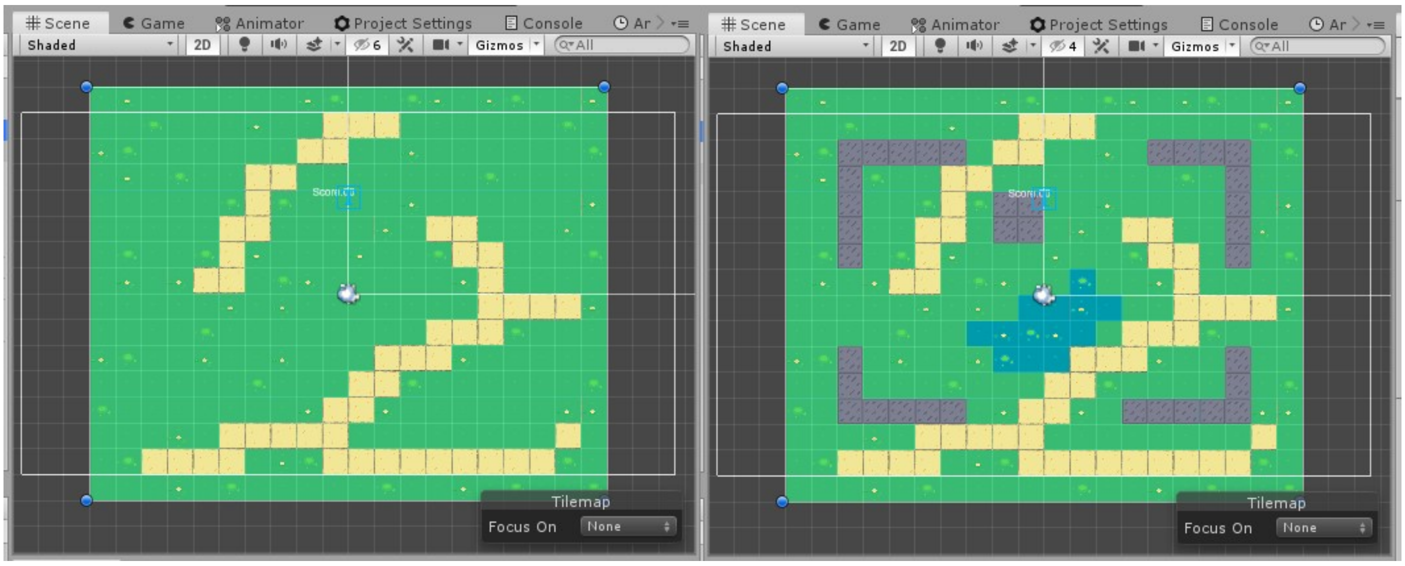
To begin with, Create a new project from Unity's "2D Template". Add a `Tilemap` GameObject to your scene (By right-clicking your Scene hierarchy and choosing the `2D Object` folder, and then `Tilemap` within). The first one you create should simultaneously create a `Grid` GameObject in the root of your scene and nest your new `Tilemap` underneath it. Add two more `Tilemap` s (for a total of 3) and and name them "Ground", "Walls" and "Pond".

Import sprites that will be used as a palette. For example, you could just use 3 solid sold-colored sprites: a green color for ground, a black for walls, and blue for the pond. Create a new palette and drag in your imported sprites.



Using the sprites imported into your palette, choose "Ground" as the `Active Tile Map` , and then paint the ground into your seen any way you'd like. In our example, we're using the solid "green" brush.

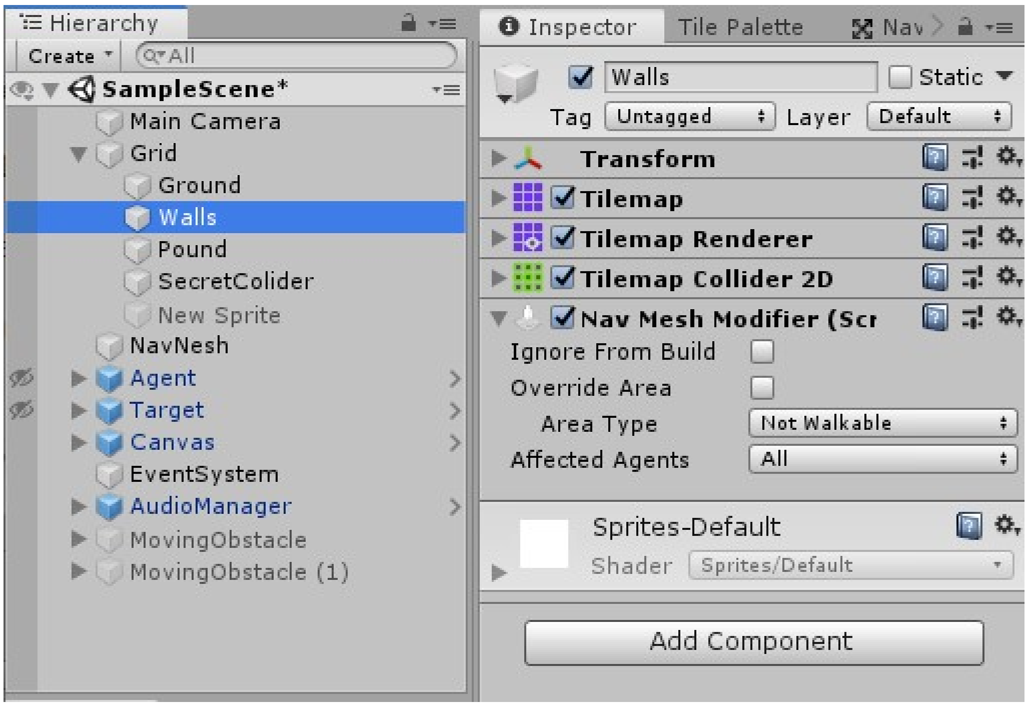
Next, change `Active Tile Map` in your palette to "Walls". Paint some obstacles. Then select "Pond" and do the same, drawing a body of water.



NAV MESH BASICS

Download NavMeshPlus and copy NavMeshComponents into your Assets folder. Now, in your scene, right click and choose `Create Empty` to add an empty GameObject into the root of your scene. Name your new empty GameObject, name it "NavMesh", and rotate it `-90, 0, 0`.

With your "NavMesh" object selected, in the Inspector click `Add Component` and choose `Nav Mesh Surface 2d`. Now select your "Walls" Tilemap GameObject (from within the `Grid` GameObject), and similarly, `Add Component` here, too. Choose a `Nav Mesh Modifier` component.



Hierarchy
- SampleScene*
  - Main Camera
  - Grid
    - Ground
    - Walls
    - Pound
    - SecretColider
    - New Sprite
  - NavNesh
  - Agent
  - Target
  - Canvas
  - EventSystem
  - AudioManager
  - MovingObstacle
  - MovingObstacle (1)

Inspector

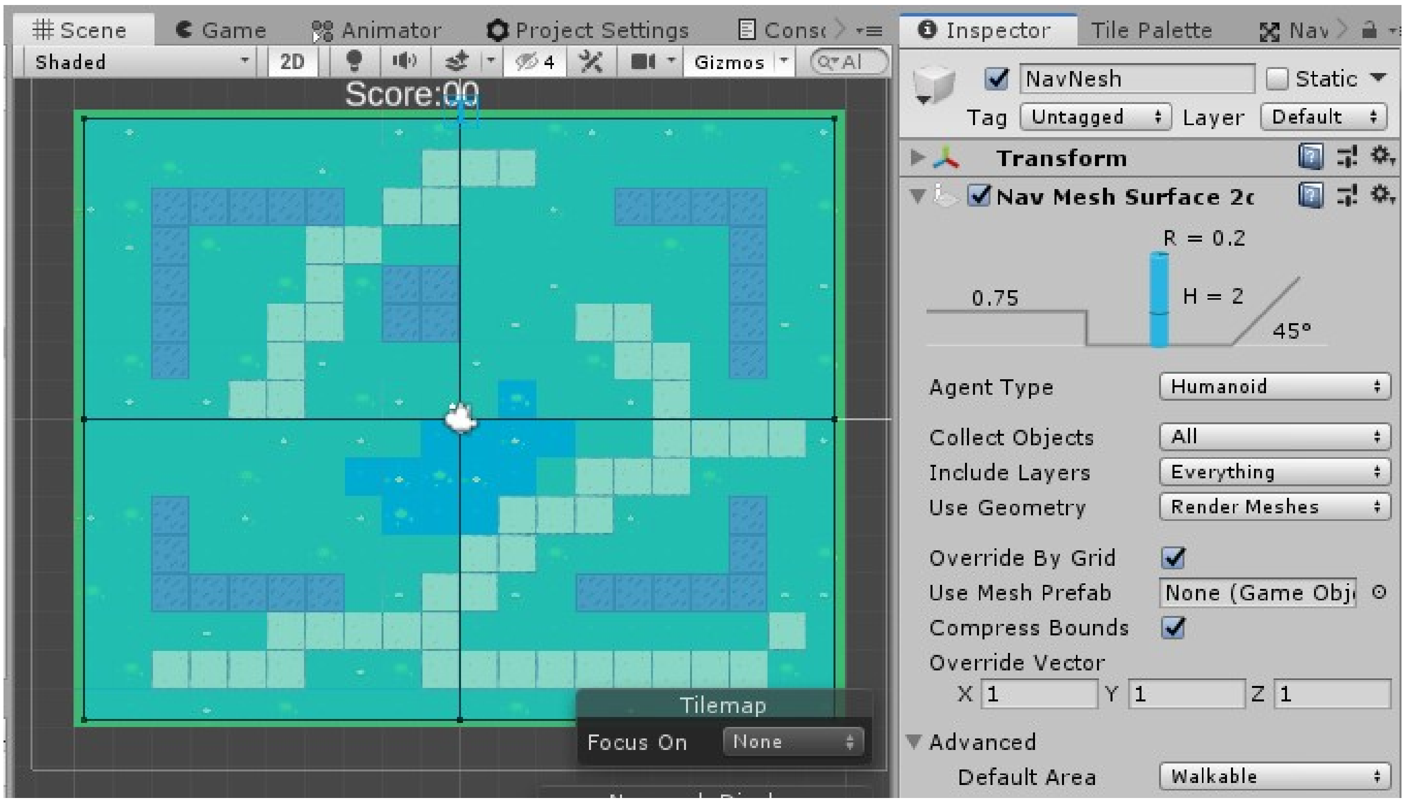
Walls — Static
Tag: Untagged — Layer: Default

- Transform
- Tilemap
- Tilemap Renderer
- Tilemap Collider 2D
- Nav Mesh Modifier (Scr
  - Ignore From Build: ☐
  - Override Area: ☐
  - Area Type: Not Walkable
  - Affected Agents: All

Sprites-Default
Shader: Sprites/Default

Add Component

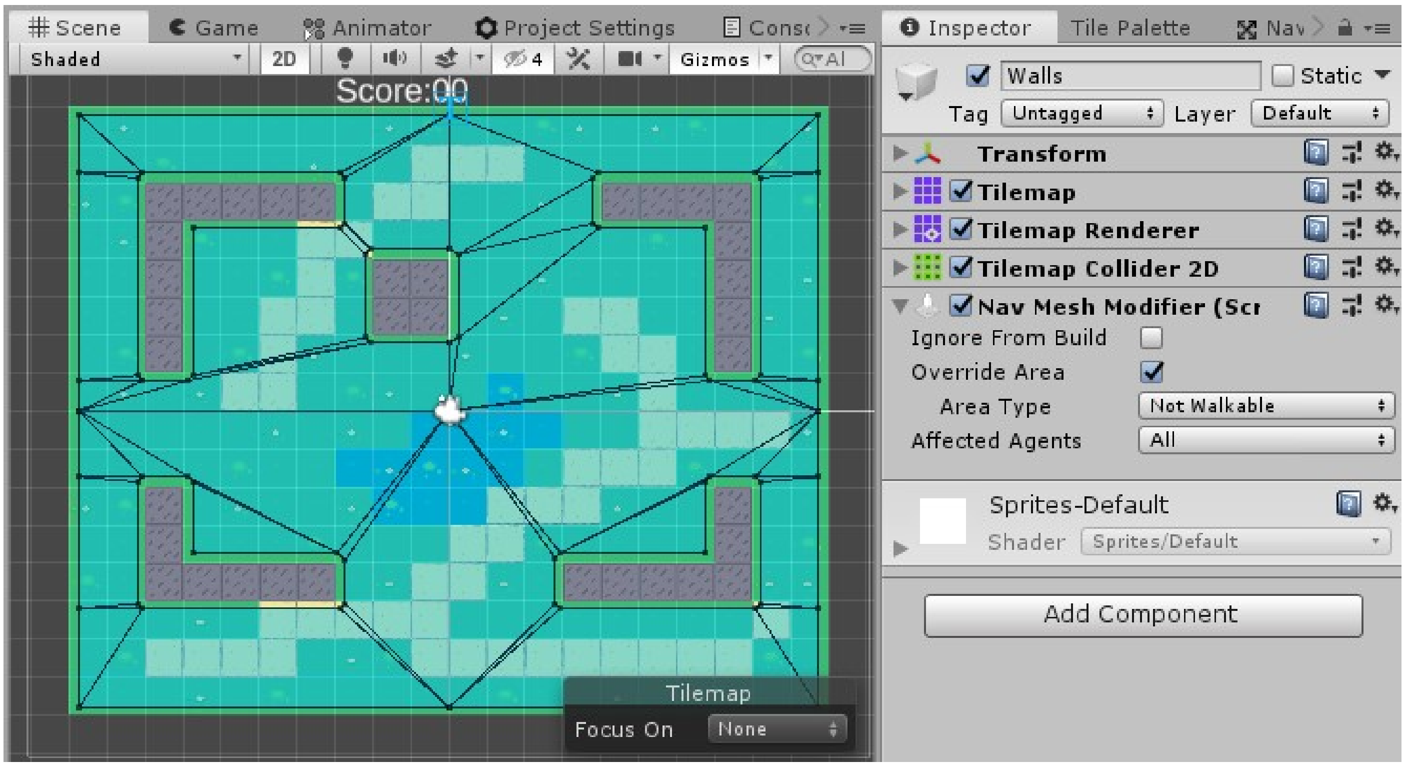
Go back to your `NavMesh` and, within the `Nav Mesh Surface 2d` component section of the inspector, hit `Bake`.



NavMesh
Tag: Untagged
Layer: Default
Transform
Nav Mesh Surface 2c
Agent Type: Humanoid
Collect Objects: All
Include Layers: Everything
Use Geometry: Render Meshes
Override By Grid: ✓
Use Mesh Prefab: None (Game Obj
Compress Bounds: ✓
Override Vector: X 1 Y 1 Z 1
Advanced
Default Area: Walkable

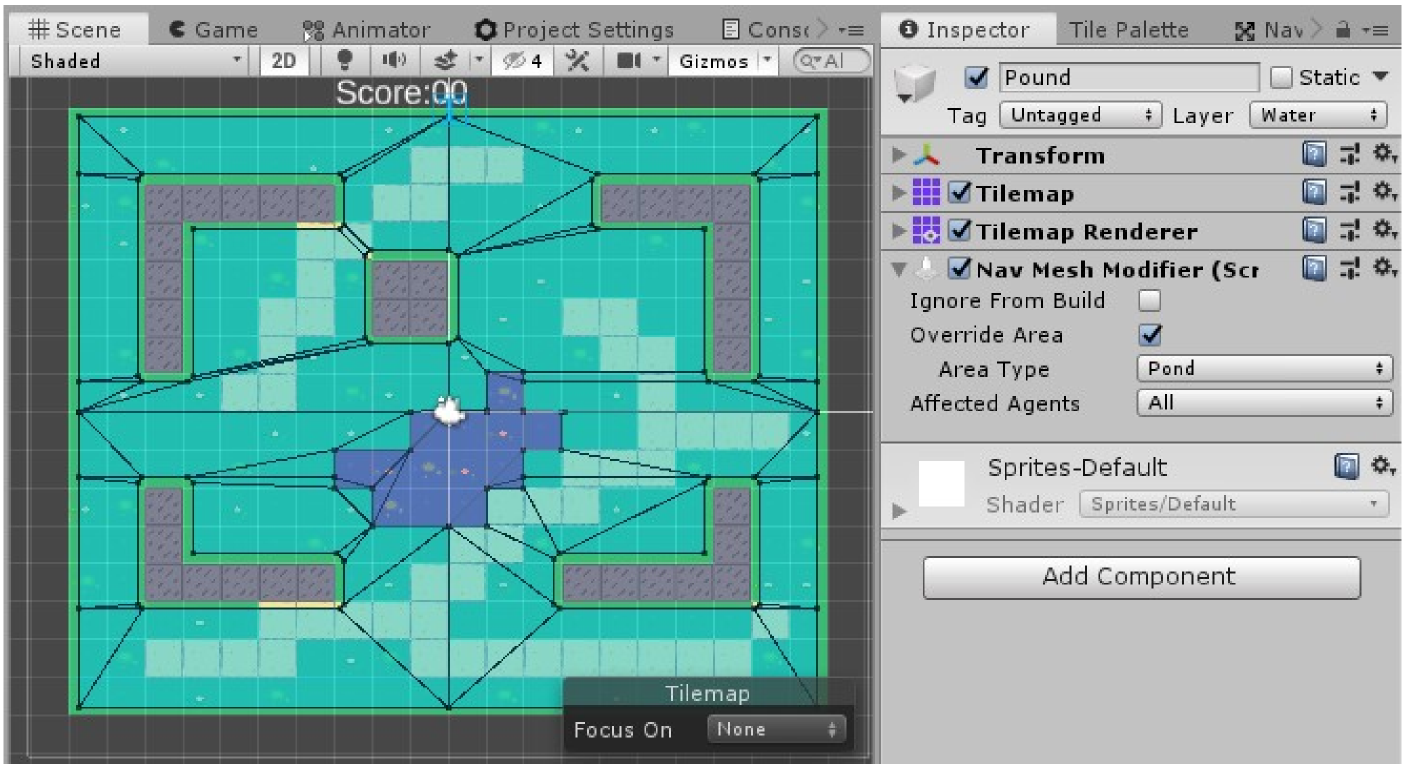
As you may be able to see, clicking `Bake` caused the whole scene to be highlighted. Hence, the entire area is considered "walkable" at this point.

Let's override it. In the `Nav Mesh Modifier` component -- remember, that's on your "Walls" -- 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 Walls you painted have been carved out of the highlighted areas. They are no longer considered "walkable".



Walls
Tag: Untagged
Layer: Default
Transform
Tilemap
Tilemap Renderer
Tilemap Collider 2D
Nav Mesh Modifier (Scr
Ignore From Build: ☐
Override Area: ✓
Area Type: Not Walkable
Affected Agents: All
Sprites-Default
Shader: Sprites/Default

Now add a `Nav Mesh Modifier` Component to your "Pond" Tilemap. Override area like you did with Walls, and choose "Pond" for `Area Type`. Bake again, and see that the "Pond" is also carved out from your nav mesh.



Pound — Static
Tag: Untagged — Layer: Water

Transform
Tilemap
Tilemap Renderer
Nav Mesh Modifier (Scr

Ignore From Build	☐
Override Area	✓
Area Type	Pond
Affected Agents	All

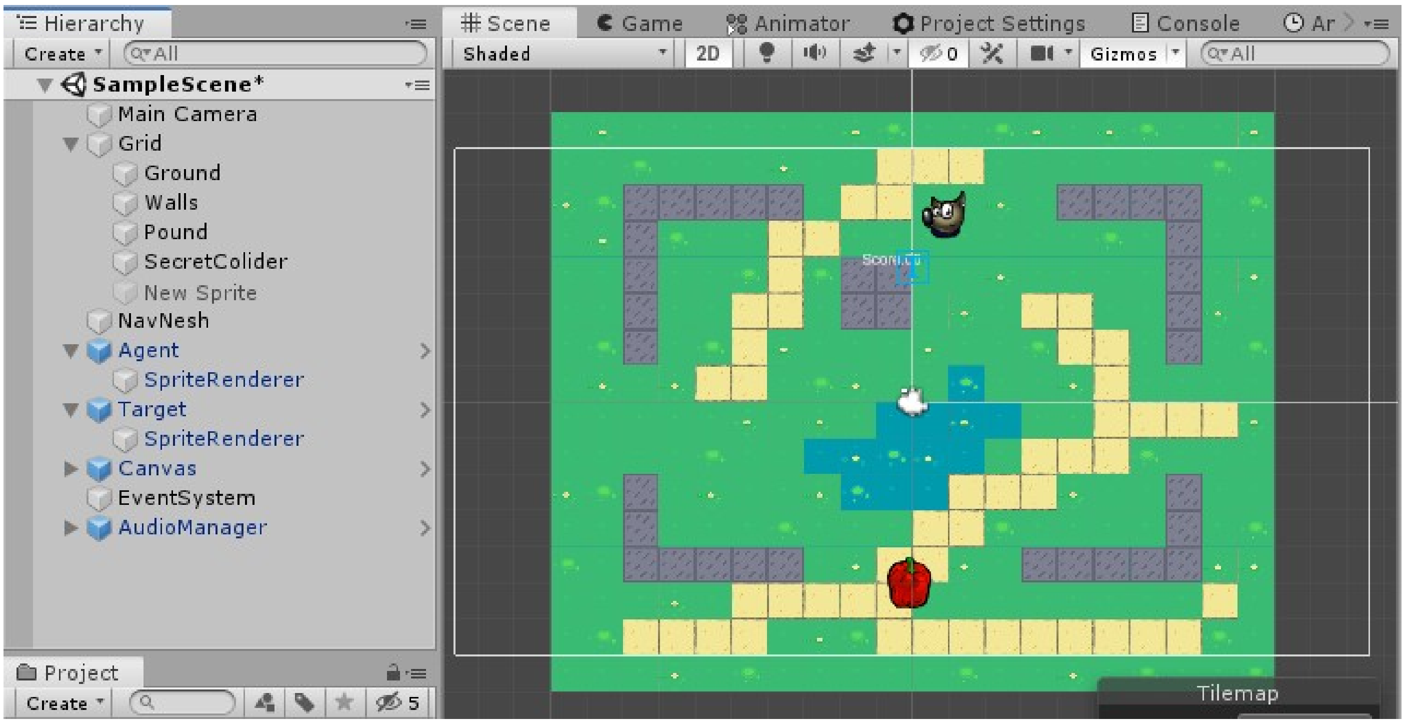
Sprites-Default
Shader: Sprites/Default

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

Important note: When a "Nav Mesh Agent" starts to interact with your scene (e.g., when they start to move in-game) the system tends to rotate the GameObject upon which the component is placed. 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.

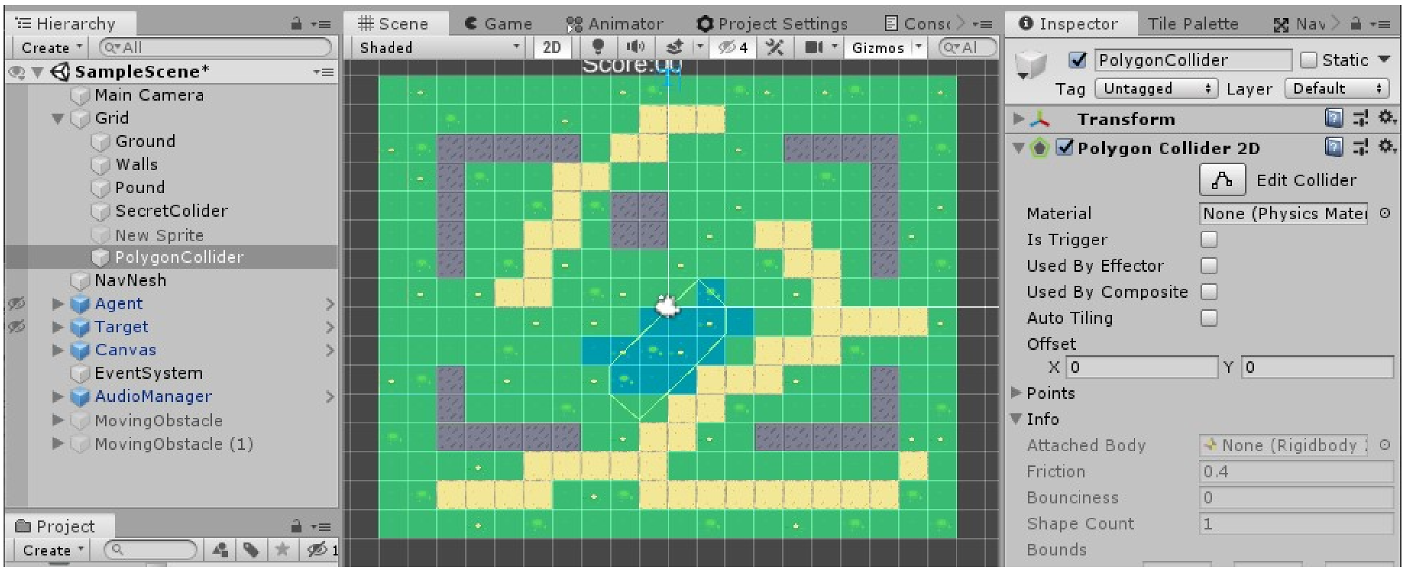


GEOMETRY OPTIONS

With new API introduced in Unity 2019.3 there is an ability to choose which geometry source to use. By default `Use Geometry` option is set to `Render Meshes` , which mean `NavMeshSurface2d` 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. To do that place all 2D colliders with `NavMeshModifier` added into navmesh GameObject as child, change `Use Geometry` option from `Render Meshes` to `Physics Colliders` and hit bake.

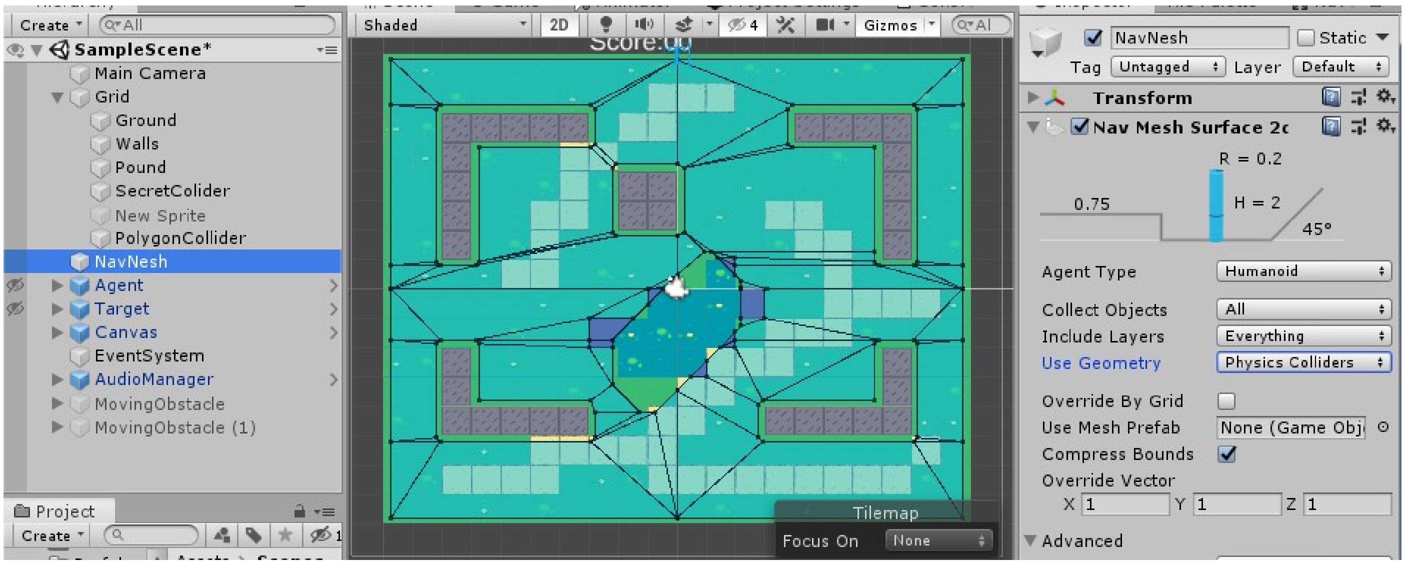
Important note: `NavMeshSurface2d` should be a parent for all colliders if this option enabled. See NON-TILED below

Lets create new "Pound" collider inside `Grid` object. Add `Polygon Collider 2d` and `NavMeshModifier` to it, override its area to "Pound". Now click `Edit Collider` to make desired pound's shape.



PolygonCollider — Static
Tag: Untagged — Layer: Default
Material: None (Physics Mater
Is Trigger: ☐
Used By Effector: ☐
Used By Composite: ☐
Auto Tiling: ☐
Offset X 0 Y 0
Points
Info
Attached Body: None (Rigidbody
Friction: 0.4
Bounciness: 0
Shape Count: 1
Bounds

Select "NavMesh" object and change `Use Geometry` option from `Render Meshes` to `Physics Colliders`, click `Bake` and new Mesh will appear.

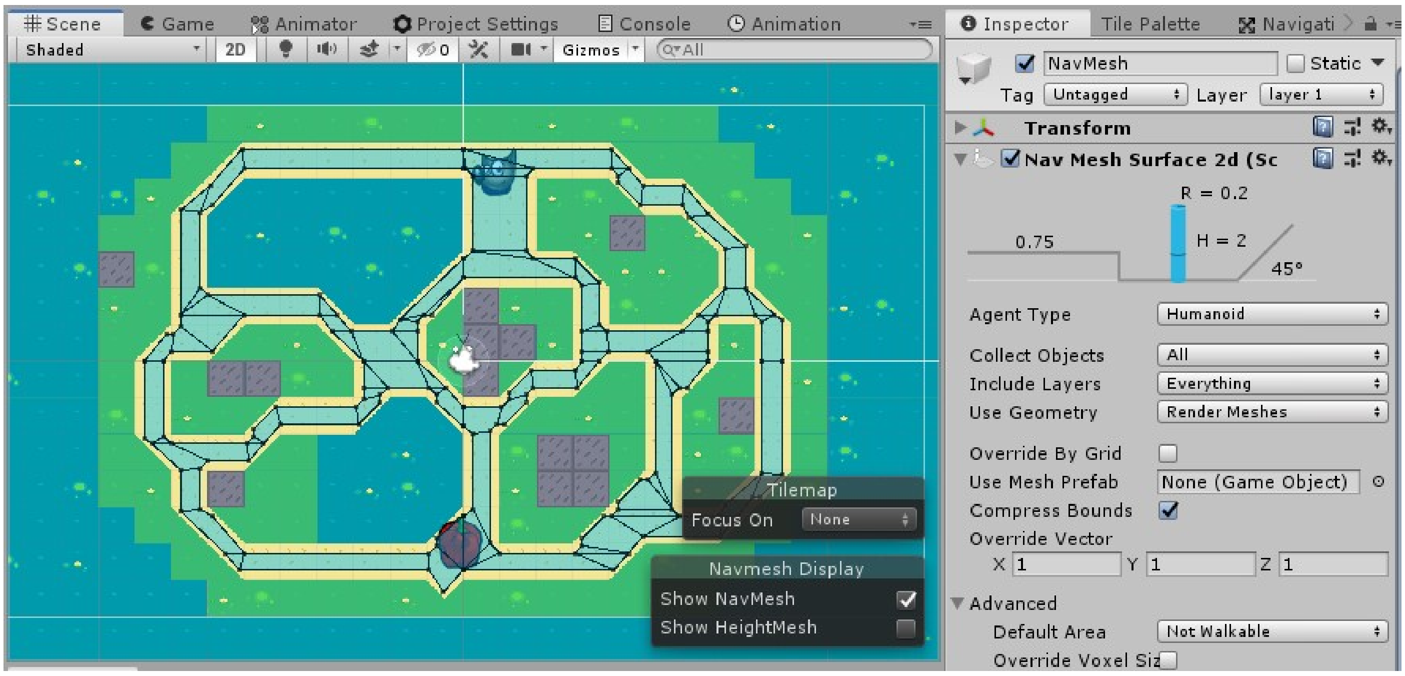


NavMesh — Static
Tag: Untagged — Layer: Default
Agent Type: Humanoid
Collect Objects: All
Include Layers: Everything
Use Geometry: Physics Colliders
Override By Grid: ☐
Use Mesh Prefab: None (Game Obj
Compress Bounds: ✓
Override Vector X 1 Y 1 Z 1
Advanced

"Walls" stay the same, because object had composite collider with exact same box shape as sprite mesh. But "Pound" now shaped not from tilemap sprites, but from `Polygon Collider 2D`.

BAKING A MAZE

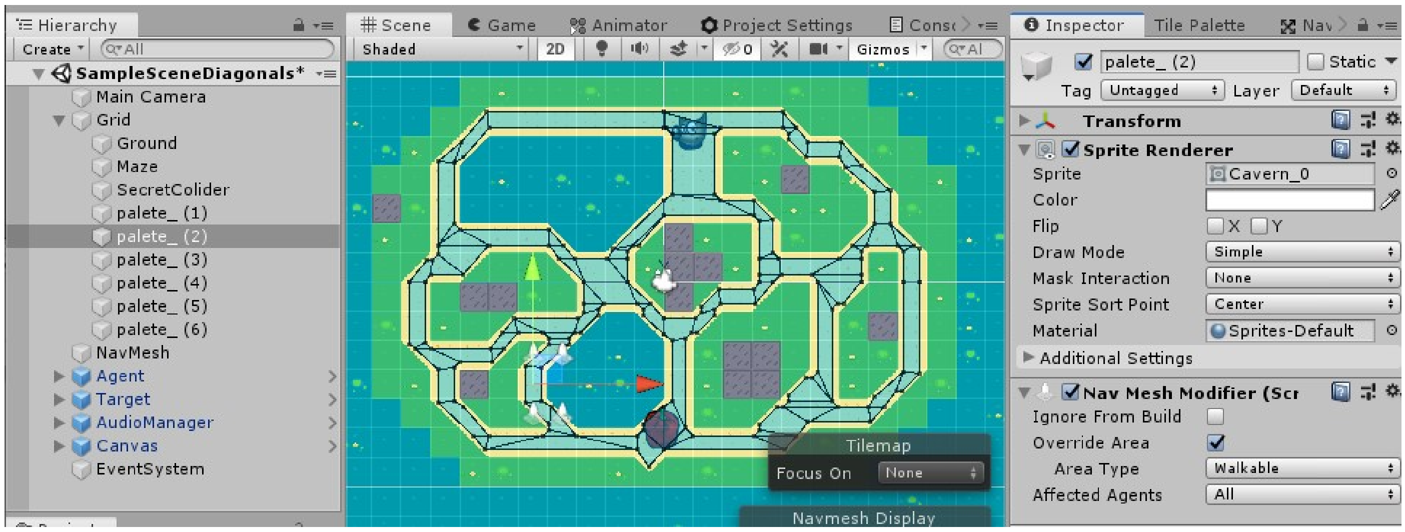
Another common task is not to carve out obstacles, but to navigate through a maze, where our agents can only follow a strict path. To do that, Bake your `NavMesh` with `Default Area` selected to `Not Walkable`, and the `Nav Mesh Modifier` component set to be `Walkable`, and you will get this:



NavMesh ☐ Static
Tag Untagged Layer layer 1
Transform
☑ Nav Mesh Surface 2d (Sc
R = 0.2
0.75
H = 2
45°
Agent Type: Humanoid
Collect Objects: All
Include Layers: Everything
Use Geometry: Render Meshes
Override By Grid: ☐
Use Mesh Prefab: None (Game Object)
Compress Bounds: ✓
Override Vector: X 1 Y 1 Z 1
Advanced
Default Area: Not Walkable
Override Voxel Siz ☐

Tilemap
Focus On: None
Navmesh Display
Show NavMesh: ✓
Show HeightMesh: ☐

Any sprite within the `Grid` object with a `Nav Mesh Modifier 2d` 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.



SampleSceneDiagonals*
Main Camera
Grid
Ground
Maze
SecretColider
palete_ (1)
palete_ (2)
palete_ (3)
palete_ (4)
palete_ (5)
palete_ (6)
NavMesh
Agent
Target
AudioManager
Canvas
EventSystem

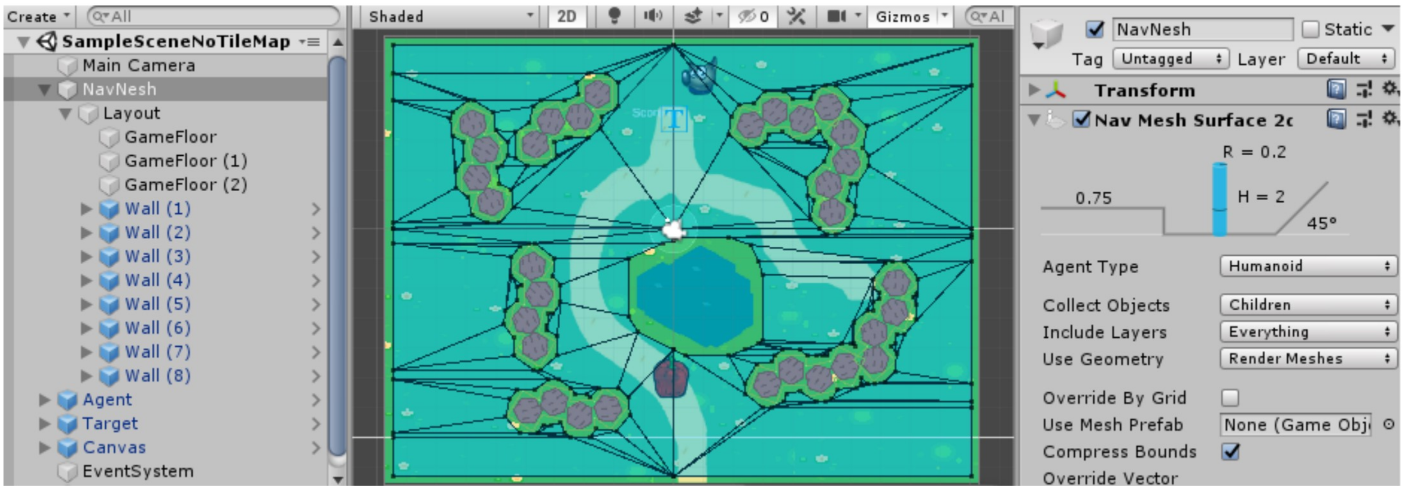
palete_ (2) ☐ Static
Tag Untagged Layer Default
Transform
☑ Sprite Renderer
Sprite: Cavern_0
Color
Flip: ☐ X ☐ Y
Draw Mode: Simple
Mask Interaction: None
Sprite Sort Point: Center
Material: Sprites-Default
Additional Settings
☑ Nav Mesh Modifier (Scr
Ignore From Build: ☐
Override Area: ✓
Area Type: Walkable
Affected Agents: All

Tilemap
Focus On: None
Navmesh Display

NON-TILED

`NavMeshSurface2d` can bake objects that are not a part of `Tilemap`, or even if there is no `Tilemap`. To do that place all `GameObject` s with `NavMeshModifier` that defines your layout as child object to `NavMeshSurface2d`, and select appropriate option `Collect Objects - Children`

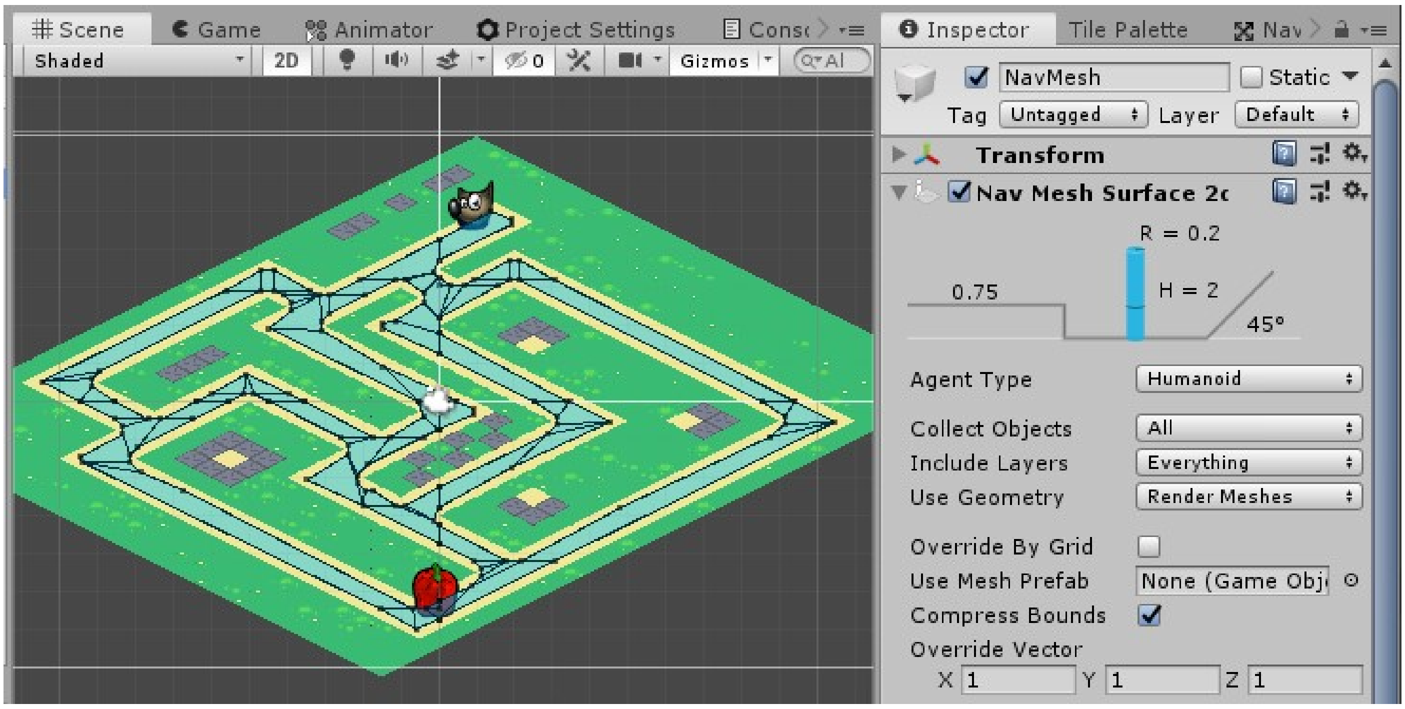
Important note: Currently there is no way to define world bound except to introduce explicit world bound `GameObject` with `NavMeshModifier`



NavMesh Static
Tag Untagged Layer Default
Nav Mesh Surface 2c
Agent Type Humanoid
Collect Objects Children
Include Layers Everything
Use Geometry Render Meshes
Override By Grid
Use Mesh Prefab None (Game Obj
Compress Bounds ✓
Override Vector

OTHER GRIDS

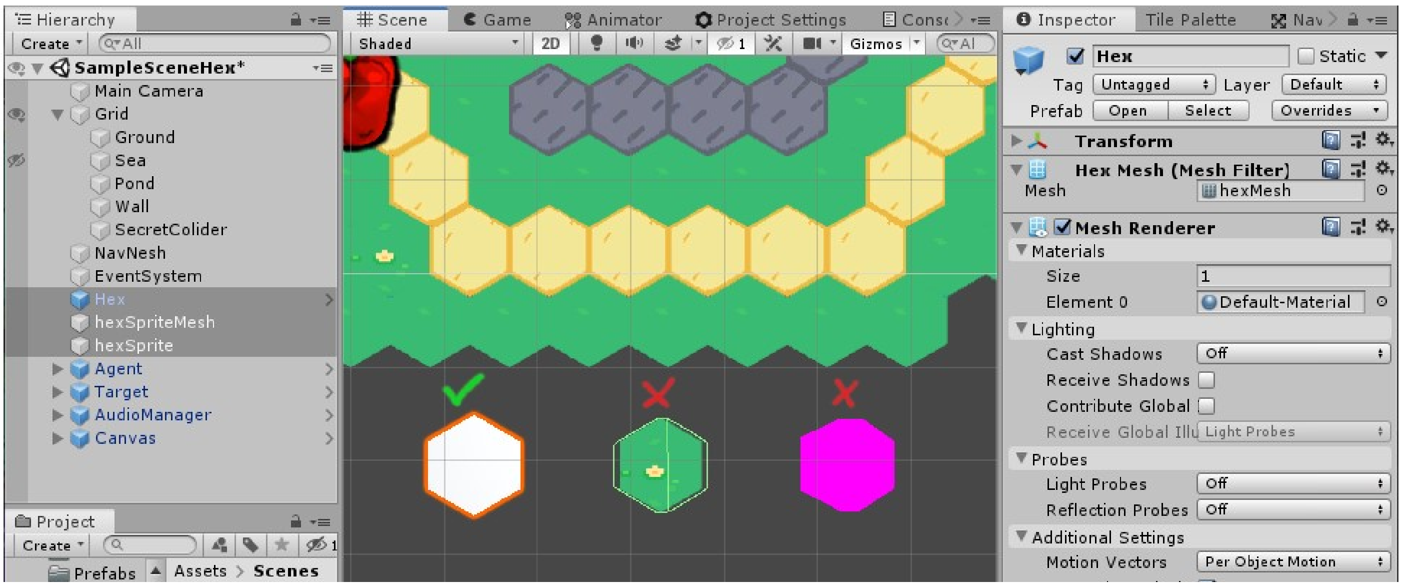
Some games want to use hexagonal, isometric, or diamond-shaped grids. NavMeshPlus supports baking other types of grid as well, as long as spites share the shape as your grid.



NavMesh Static
Tag Untagged Layer Default
Nav Mesh Surface 2c
Agent Type Humanoid
Collect Objects All
Include Layers Everything
Use Geometry Render Meshes
Override By Grid
Use Mesh Prefab None (Game Obj
Compress Bounds ✓
Override Vector
X 1 Y 1 Z 1

OVERRIDE BY

In some cases the sprite can differ from your grid shape, it may have excessive outline where we want clean navigation though the tilemap or sprites in hundred different shapes where it tedious to edit each outline. For that case we can override sprite meshes altogether.

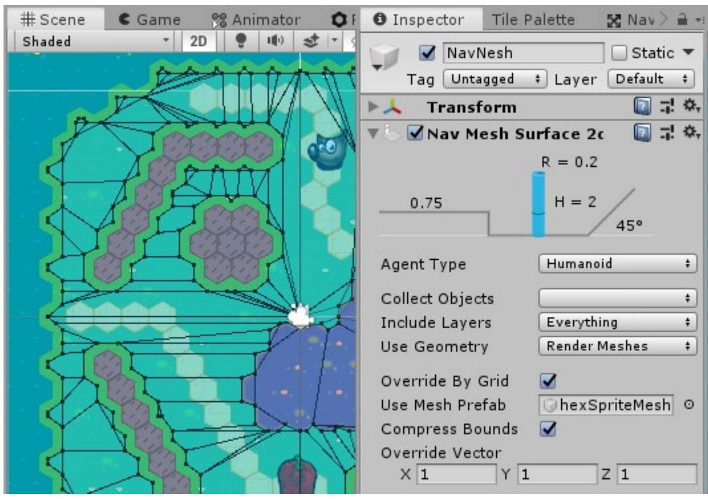


SampleSceneHex*
Main Camera
Grid
Ground
Sea
Pond
Wall
SecretColider
NavNesh
EventSystem
Hex
hexSpriteMesh
hexSprite
Agent
Target
AudioManager
Canvas

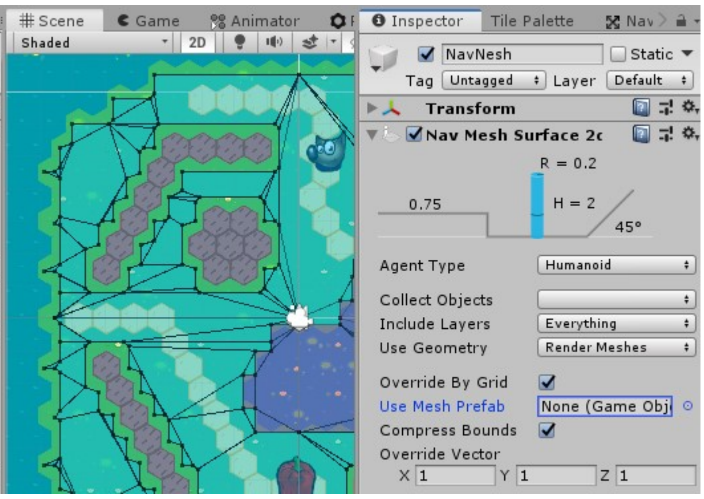
Hex — Static
Tag: Untagged — Layer: Default
Prefab: Open — Select — Overrides
Transform
Hex Mesh (Mesh Filter)
Mesh: hexMesh
Mesh Renderer
Materials
Size: 1
Element 0: Default-Material
Lighting
Cast Shadows: Off
Receive Shadows
Contribute Global
Receive Global Illu: Light Probes
Probes
Light Probes: Off
Reflection Probes: Off
Additional Settings
Motion Vectors: Per Object Motion

We have option to override all tiles within a grid with a prefab. First we need to have a mesh that has the form 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 `Nav Mesh Surface 2d` component, check `Override By Grid` and all tiles will have a shape of used prefab. Even if this option disabled, all sprites without a proper mesh will be substituted with this one.



NavNesh — Static
Tag: Untagged — Layer: Default
Transform
Nav Mesh Surface 2d
R = 0.2
0.75
H = 2
45°
Agent Type: Humanoid
Collect Objects:
Include Layers: Everything
Use Geometry: Render Meshes
Override By Grid: ✓
Use Mesh Prefab: hexSpriteMesh
Compress Bounds: ✓
Override Vector
X 1 Y 1 Z 1



NavNesh — Static
Tag: Untagged — Layer: Default
Transform
Nav Mesh Surface 2d
R = 0.2
0.75
H = 2
45°
Agent Type: Humanoid
Collect Objects:
Include Layers: Everything
Use Geometry: Render Meshes
Override By Grid: ✓
Use Mesh Prefab: None (Game Obj
Compress Bounds: ✓
Override Vector
X 1 Y 1 Z 1

If we remove the prefab and leave the override option, all tiles will be considered as squares.

BAKE AT RUNTIME

Often games have procedural world building which requires bake NavMesh at runtime. To achive this call a single method `BuildNavMeshAsync` in `NavMeshSurface2d` component to bake NavMesh at runtime in Unity async manner. This method can be called on `Start()` method.

```
public NavMeshSurface2d Surface2D;
```

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

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

```
    public NavMeshSurface2d Surface2D;

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

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

```
        NavMeshBuilder.UpdateNavMeshDataAsync(data, GetBuildSettings(), sources, sourcesBou
```

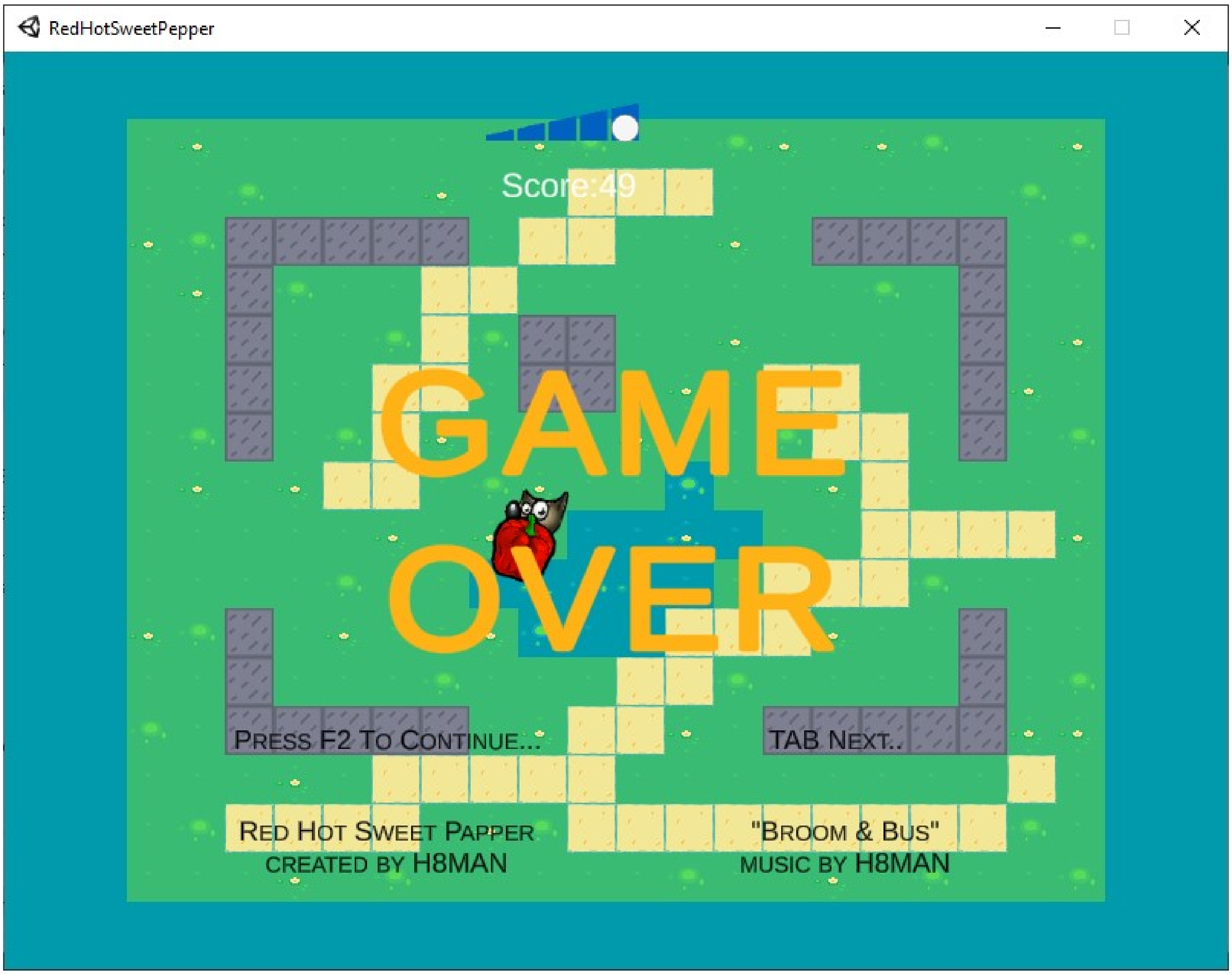
Important note: Collider's bounds not getting updated to match the transform until 'LateUpdate()' is called. Force the update by calling Physics2D.SyncTransforms() before calling 'BuildNavMesh()' if you baking navmesh at runtime.

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

THE END

Try out a fully-featured 2D "run-and-chase" game I made to demo NavMesh2d: [RedHotSweetPepper].



References:

1. NavMeshSurface - https://docs.unity3d.com/Manual/class-NavMeshSurface.html
2. NavMeshSurface2d - 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