FlowField Path Documentation

Setup

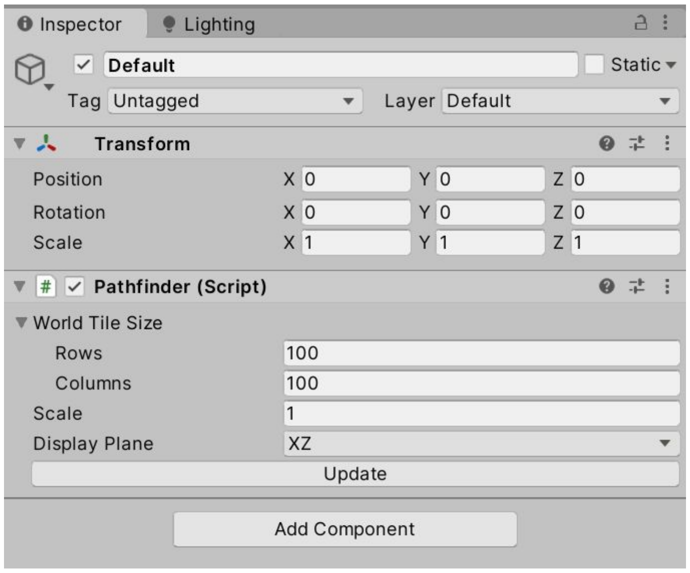
PathfindingService

Add a PathfinderService component to your scene. This component will contain references to all pathfinders in your scene and will serve as a container to get a pathfinder of your desire in code. Make sure PathfinderService is positioned on 0,0,0 coordinates.

Add Pathfinders

To create Pathfinder, create a new GameObject as a child to PathfinderService GameObject. Set the name of this GameObject to your desired pathfinder name. This name will be used to retrieve the pathfinder from PathfinderService.

Inspect the pathfinder component and set up your world size, scale and plane it projects onto. **Now click the Update button**. This will generate pathfinding data for this particular pathfinder.



Default
Static
Tag Untagged
Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Pathfinder (Script)

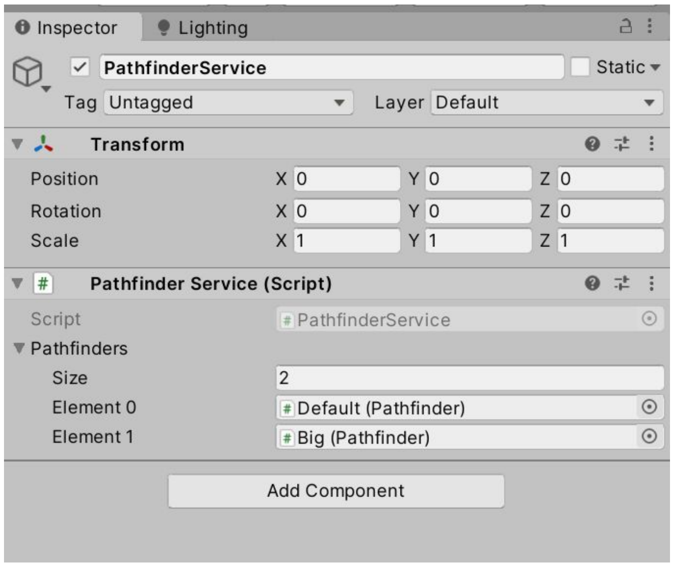
World Tile Size

Rows	100
Columns	100
Scale	1
Display Plane	XZ

Update

You need to reference this Pathfinder in PathfinderService. You can do so by simply selecting PathfinderService and drag & drop the PathfinderComponent into Pathfinders property.

You can add multiple pathfinders, with various sizes and scales - to support unit of different sizes.



PathfinderService
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Pathfinder Service (Script)

Script	PathfinderService
Pathfinders	
Size	2
Element 0	Default (Pathfinder)
Element 1	Big (Pathfinder)

Important
Whenever you change any property of Pathfinder it is important to click the **Update** button.

Important
Ensure that you have at least one pathfinder named '**Default**'. This ideally would be a pathfinder with scale set to 1.

Properties of Pathfinder

WorldTileSize
Size of the world in tiles.

Scale
Relates to scale between Unity unit and tile size. This property is introduced to provide finer control over size of Moving components. If the unit is too big or too small you might want to adjust the scale. Scale == 2, means 1 Tile in the pathfinding world will be 2x2 Unity units big.

DisplayPlane
Switch between 2D and 3D games. XZ is suitable for most 3D games, where XY is the standard plane for Unity 2D games.

CollisionMask
CollisionMask is a Unity Layer mask that is used to check if tile is collided / blocked.

Getting Path

Only several lines of code are required to get our Path object ready to be used and move along its path.

```
1.
var pathfinder = PathfinderService.Default.Get("Default");
2.
var flowPath = new FlowField(pathfinder);
3.
var worldPosition = pathfinder.CreateWorldPosition(transform.position);
4.
var worldIndexes = new List<int> {worldPosition.WorldIndex};
5.
flowPath.CreateDestinationGateways(worldIndexes);
```

Let's break it down.:
1.
First step is to get a pathfinder of our desire. This should correspond with our unit size [in case you have multiple pathfinders, based on size of unit].

In this case we have retrieved pathfinder of Default size, which in provided examples always correspond to 1 tile == 1 unity unit,

2.
Then we create FlowField and provide it with our Pathfinder we have retrieved. FlowField is our actual path, in a world defined by the provided Pathfinder.

3.
Then we need to define our destination. However for faster calculations, pathfinder uses its own structure to hold positional value. This is called WorldPosition. Each pathfinder provides sets of functions to create world position from various values. In this case. We simply pass in the position of our destination point.

4.
Now we create List<int> that represents any tile in the world that can be considered as a destination tile.

Note.: We are using a list, because in some use-cases there can be multiple destination tiles. Let's use the RTS game as an example scenario. If we want to find the shortest path to a building, the appropriate destination is every tile that surrounds our building. Thus the destination can contain dozens of tiles. It also opens a possibility of multiple destinations.

5.
Now we simply pass our values in FlowPath, so it processes the values internally and prepares all the data for the construction path.

Note.

To be truly effective, FlowField that leads to the same destination, should be shared among all units that share this destination. Creating a new FlowField for every unit will result in calculation heavy pathfinding. For simplicity of setup documentation, this was not included. However if you wish to review sample code where FlowFields are pooled, see DestinationPoint script that is used in all the example scenes.

Getting Movement direction

We have created our path in the previous section, yet we don't know which direction to move in. To get direction to move in from any position in scope of Pathfinder World we simply must do the following.

In our movement component **_flowField** variable holds FlowField. How to retrieve flow field was described in previous section - **Getting Path**

On component we wish to move:

```
if (_flowField.Ready(transform.position))
{
   transform.position += _flowField.GetDirection(transform.position);
}
```

We simply check if FlowField has calculated a path on the provided position. That is done by checking _flowfield.Ready(positionFromWhichWeMove).
Outcomes of this method are:

True:
Path has been calculated and is ready, we can simply ask for direction! We can move in that direction.

False:
Path has not been calculated. FlowField will now try to find a path from the provided position to the destination. If the path does not exist, it will return false and it's not possible to reach the destination from this point. You could terminate the movement, unless there will be a change in pathfinding world, new path will not be found.

Obstacles & Traversal costs

Traversal Costs & Impassable area

Traversal costs are arrays of byte values stored in Pathfinder.

You can set up traversal cost for any tile of the Pathfinder defined world. Traversal cost is taken into account when path is calculated and some traversal expensive tiles can be avoided if possible.

If traversal cost is set to maximum value byte.Max or used by constant Sector.Impassable, tile will be considered as Impassable.

Traversal costs should be always set in the editor or prior to starting the game.
If you want to add or remove obstacles dynamically, see **Dynamic Obstacles** section.

To set a particular cost of traversing on specified WorldPosition, you can simply retrieve the pathfinder you wish to adjust. To adjust cost, use the following function of Pathfinder.

```
public void SetCost(WorldPosition position, byte cost = byte.MaxValue)
```

Example
Following example will retrieve “Default” pathfinder and set tile on position 10,0,10 as impassable.

```
var pathfinder = PathfinderService.Default.Get("Default");
pathfinder.SetCost(pathfinder.CreateWorldPosition(new Vector3(10, 0,
0)), Sector.Impassable);
```

Dynamic Obstacles

In the Pathfinder inspector we can see a property named **CollisionMask**. Every collider with a layer within this mask will be considered an obstacle.

So adding obstacles is as simple as adding and removing objects from the scene, with colliders. However if you add or remove an object, or move an object that is considered an obstacle, you need to inform Pathfinder, to update a certain area.

Following method is used to do just that:

```
public void UpdateArea(Vector3 position, int size)
```

Once called, pathfinder will inform all FlowFields that are using it, and all the paths that are affected by these changes will be recalculated.

Examples:

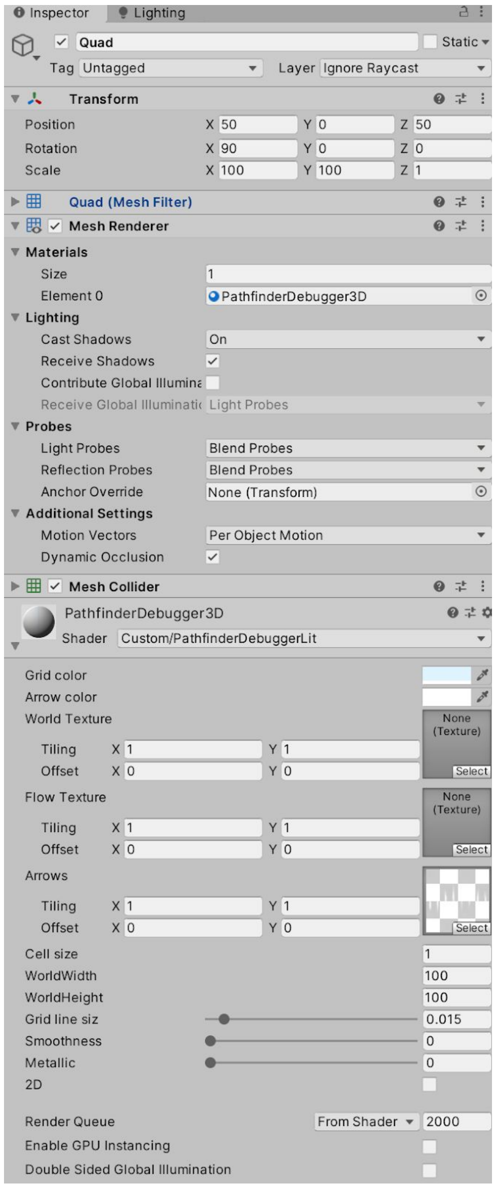
Destination
In examples, destination is a separate component called DestinationPoint. This component is where Movement components retrieve their paths from.

Movement
In examples the movement component is represented by a script Movable. Compared to the simple setup described in this documentation, movable contains some steering behaviors and sliding along walls behaviors..

Setup Debugging

Setting up Debugging is not necessary for the solution to work. This is only if you want to debug and visualise how the Pathfinding world and Path calculation looks like.

Solution comes with material prepared to render pathfinding grid and path information. For 3D space the material is called PathfinderDebugger3D, for 2D space the material is called PathfinderDebugger2D. Simply set the material on the renderer component you want to render pathfinding data on. In case of example scenes, Quad Component was used. However you can set this to the Terrain component as well.



Quad ✓ | Static
Tag: Untagged | Layer: Ignore Raycast

Transform

Position	X 50	Y 0	Z 50
Rotation	X 90	Y 0	Z 0
Scale	X 100	Y 100	Z 1

Quad (Mesh Filter)

Mesh Renderer ✓

Materials
- Size: 1
- Element 0: PathfinderDebugger3D

Lighting
- Cast Shadows: On
- Receive Shadows: ✓
- Contribute Global Illumina
- Receive Global Illuminati: Light Probes

Probes
- Light Probes: Blend Probes
- Reflection Probes: Blend Probes
- Anchor Override: None (Transform)

Additional Settings
- Motion Vectors: Per Object Motion
- Dynamic Occlusion: ✓

Mesh Collider ✓

PathfinderDebugger3D
Shader: Custom/PathfinderDebuggerLit

Grid color
Arrow color
World Texture: None (Texture) — Select
- Tiling: X 1, Y 1
- Offset: X 0, Y 0

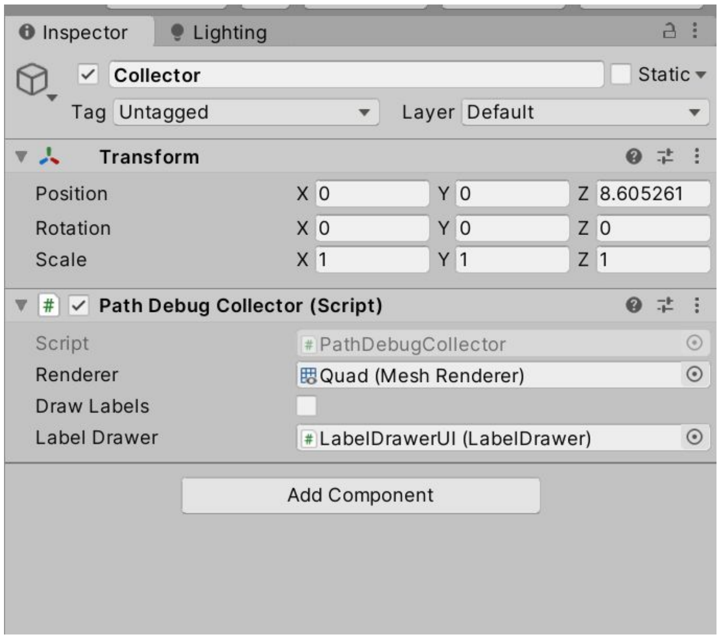
Flow Texture: None (Texture) — Select
- Tiling: X 1, Y 1
- Offset: X 0, Y 0

Arrows — Select
- Tiling: X 1, Y 1
- Offset: X 0, Y 0

Cell size: 1
WorldWidth: 100
WorldHeight: 100
Grid line siz: 0.015
Smoothness: 0
Metallic: 0
2D

Render Queue: From Shader 2000
Enable GPU Instancing
Double Sided Global Illumination

Ensure there is a PathDebugCollector component in the scene. This can be attached to any GameObject. PathDebugCollector is responsible for collecting and sending data into Renderer's material.



Collector · Static
Tag: Untagged · Layer: Default

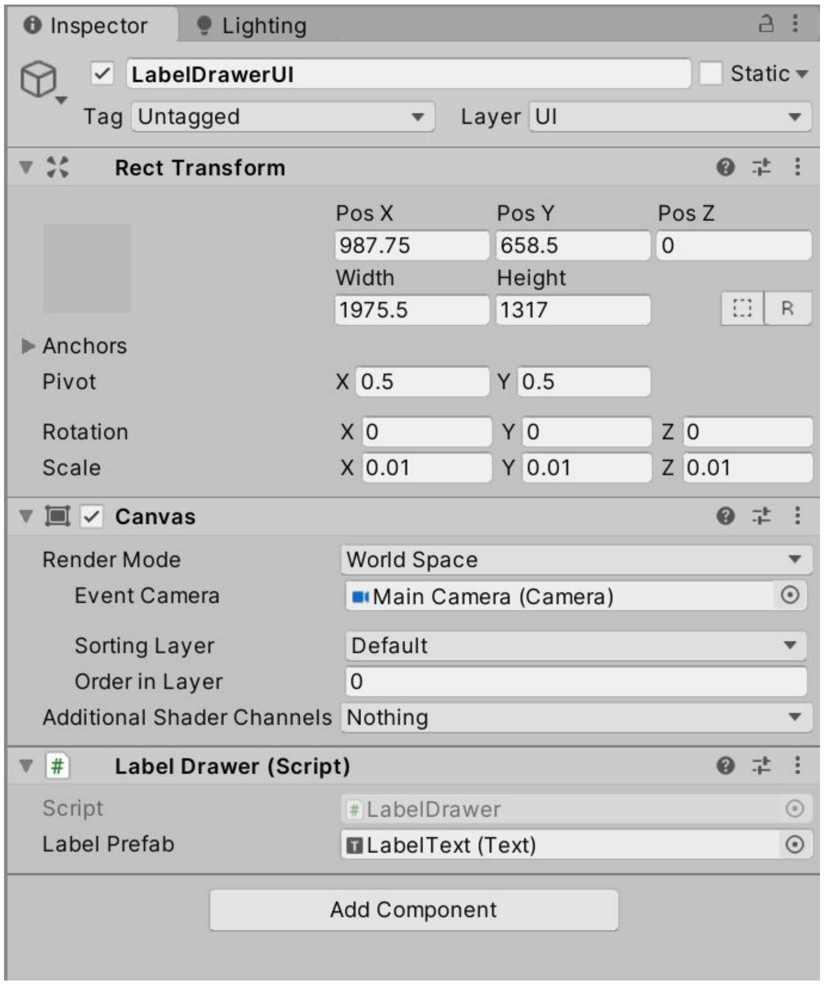
Transform

	X	Y	Z
Position	0	0	8.605261
Rotation	0	0	0
Scale	1	1	1

Path Debug Collector (Script)

Script	PathDebugCollector
Renderer	Quad (Mesh Renderer)
Draw Labels	☐
Label Drawer	LabelDrawerUI (LabelDrawer)

Ensure there is a LabelDrawer component attached to the WordSpace Canvas component in scene.


Inspector | Lighting

LabelDrawerUI | Static

Tag: Untagged | Layer: UI

Rect Transform

	Pos X	Pos Y	Pos Z
	987.75	658.5	0
	Width	Height	
	1975.5	1317	

Anchors

	X	Y	Z
Pivot	0.5	0.5	
Rotation	0	0	0
Scale	0.01	0.01	0.01

Canvas

Render Mode	World Space
Event Camera	Main Camera (Camera)
Sorting Layer	Default
Order in Layer	0
Additional Shader Channels	Nothing

Label Drawer (Script)

Script	LabelDrawer
Label Prefab	LabelText (Text)

Add Component

To enable debugging of particular pathfinder & flow field [can be null, in that case it will only debug pathfinder world, not path] simply call

```
PathDebugCollector.Default.PopulateFlowPath(Pathfinder, FlowField);
```