Table of Contents

An Introduction

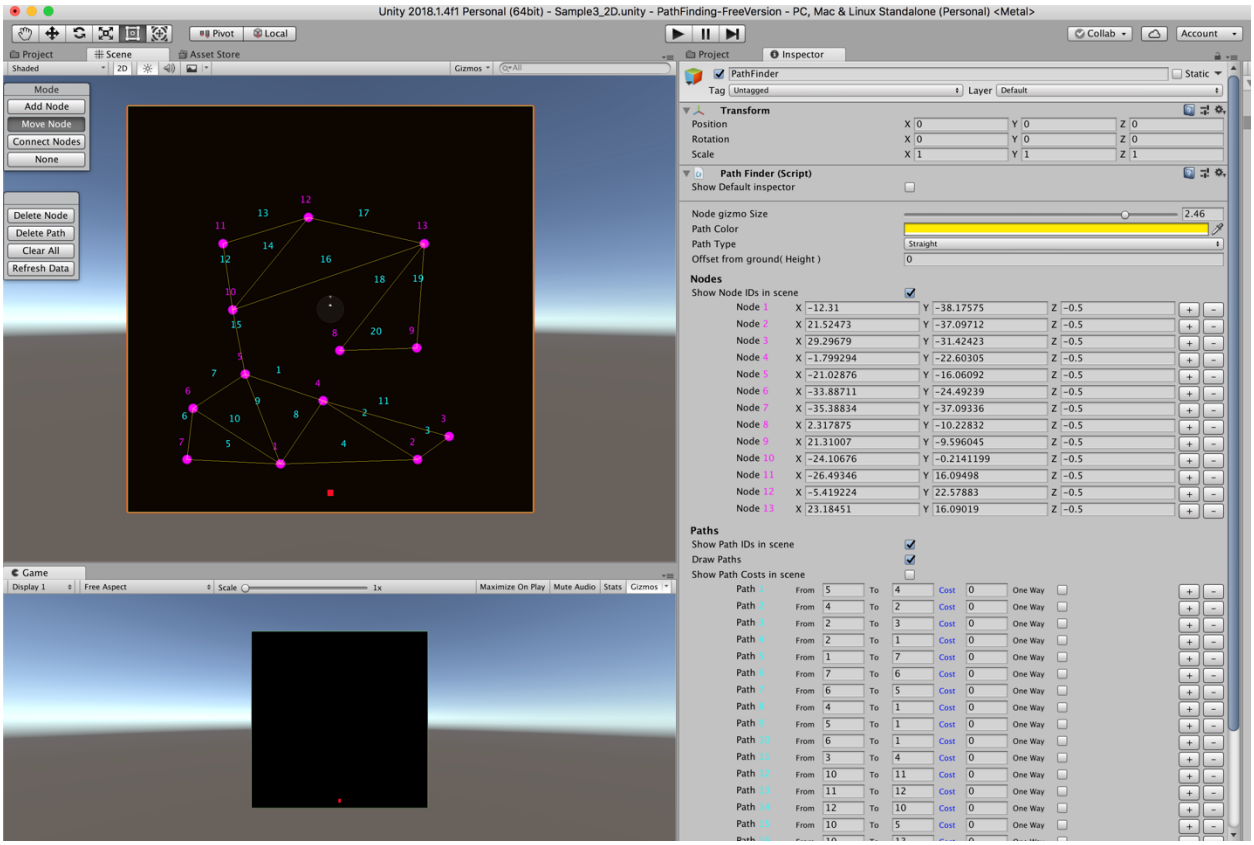
I have used A* Path-finding Heuristic algorithm which is an extension of Dijkstra's Shortest Path First.

What this is Not : This is not a grid based path finding algorithm. We need to set up Nodes and Paths.

What this is: This is Node based path finding.

You have to set up multiple Nodes and you inter-connect them like in the image below. Once you have that, this algorithm will provide the shortest path from one point on the ground* to another.

Ground* - is the collider on which we can add nodes.



What are Nodes and Paths ?

1. Nodes are like Junctions in real life. Each Node has a specific ID which is auto generated.
2. Path is a connection between 2 Nodes. Paths are like Roads in real life. Each Path has a specific ID, which is auto generated. Do not mix Node ID with Path ID. They are both different and marked in different colors.

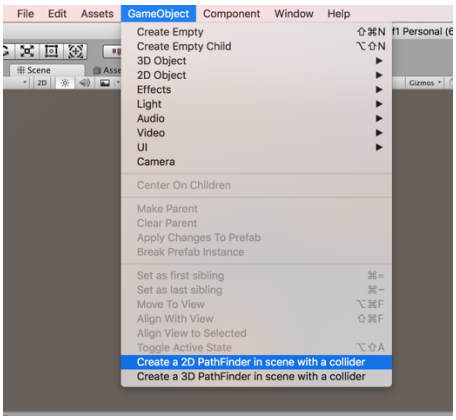
3. (Optional) Once you have the paths, you can give cost per each path. Path finding prefers to use paths with Lower cost. Or you can set few Paths as One-ways.

Note: Remember that Path finding with not only take Path's Cost into consideration, it will also consider distance between the nodes.(Heuristic Distance).

How to create Nodes and Paths for 2D

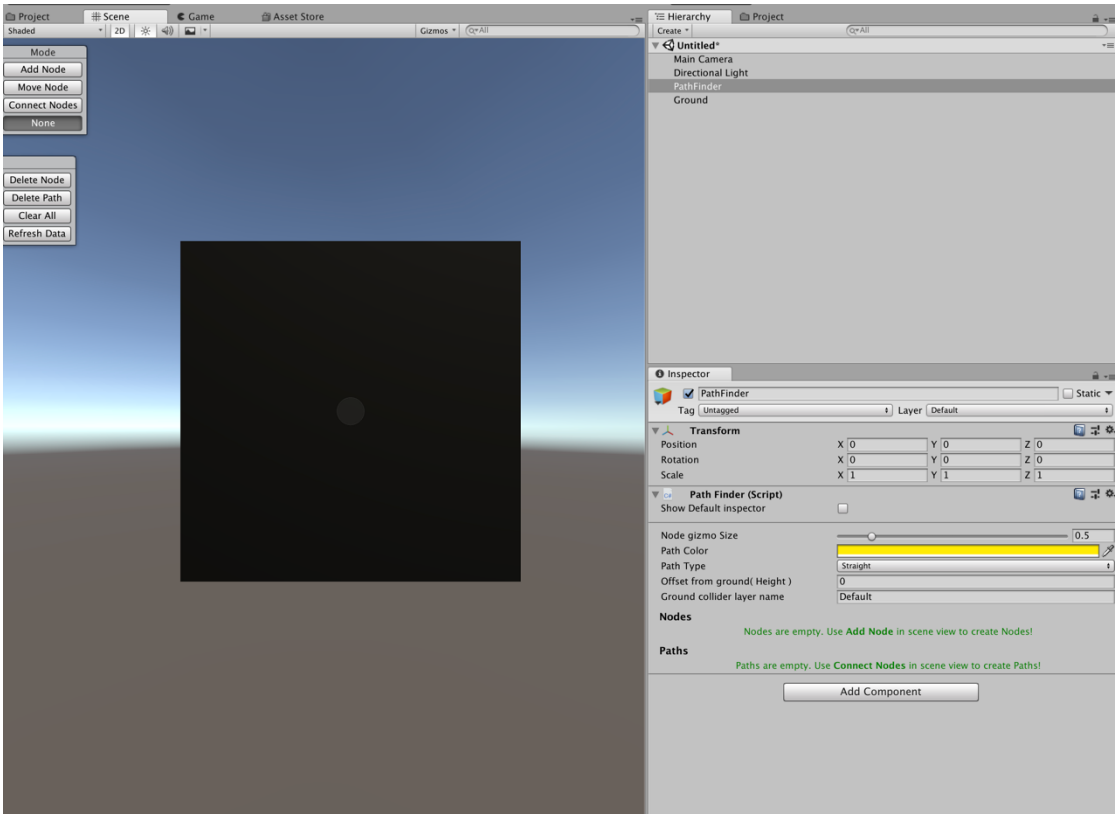
Step 1: Create PathFinder

Create PathFinder script and Ground colliders in the scene

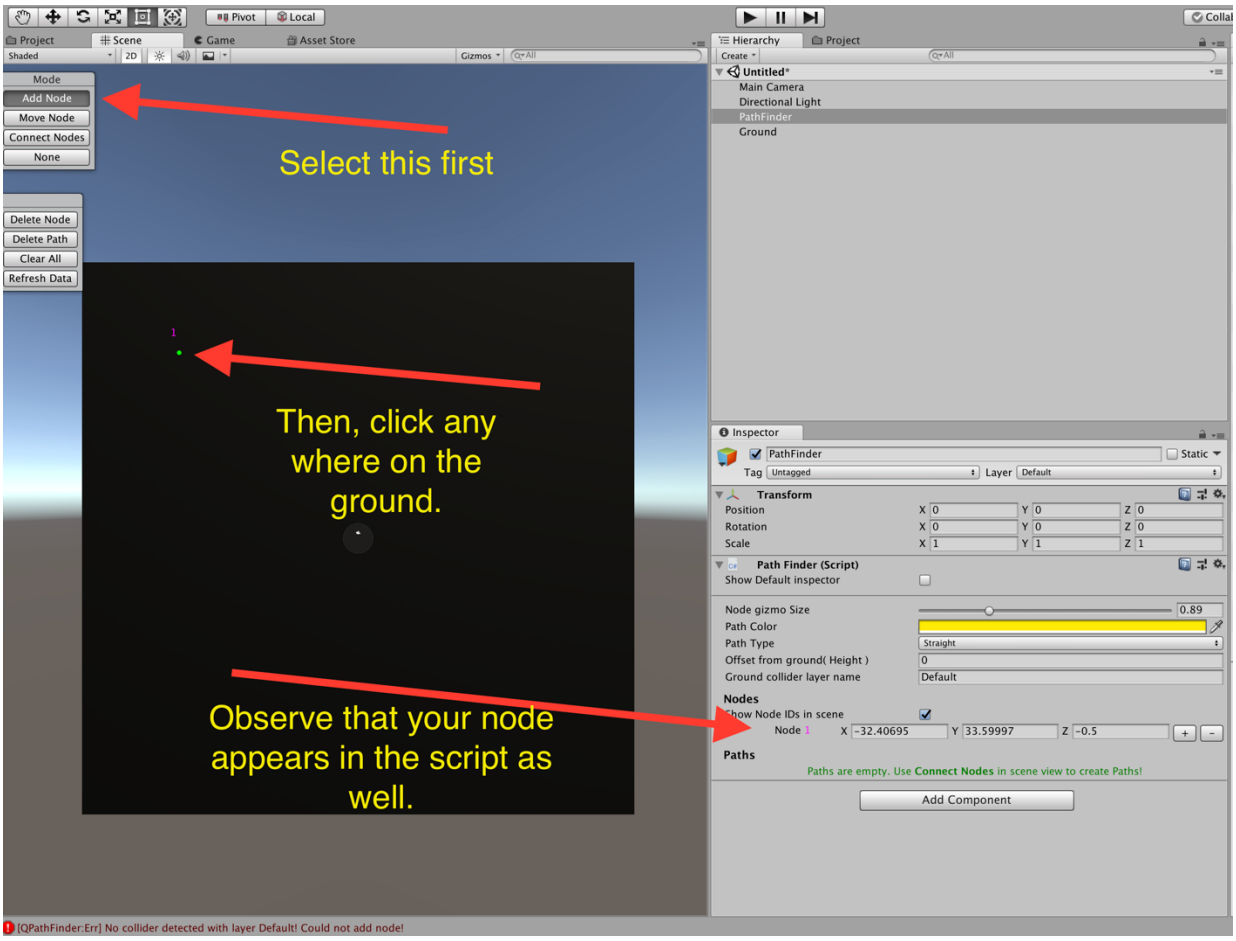


Step 2: Select PathFinder

Select PathFinder Gameobject in hierarchy. This brings ups the Pathfinder GUI on the scene view.

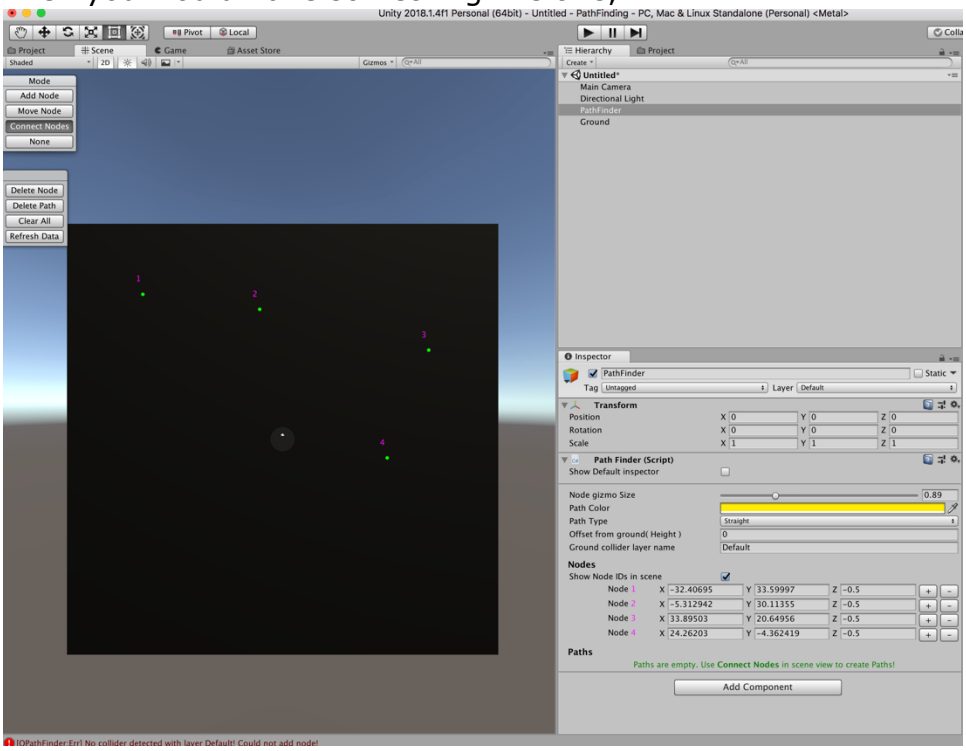


Step 3 : Create nodes



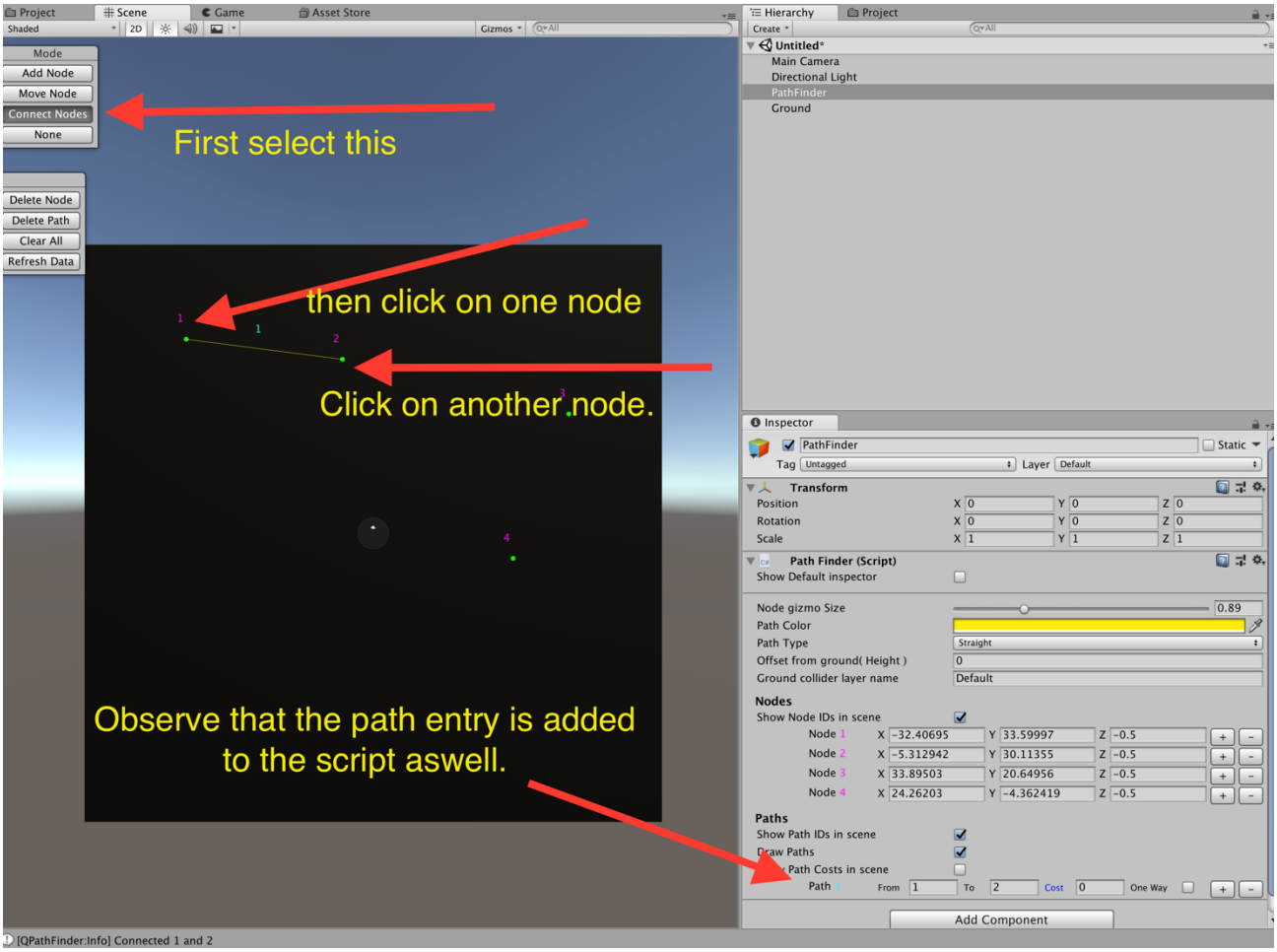
Repeat the process multiple times.

Then you would have something like this,



Step 4 : Create Paths (Connections)

This is a very important step, which u create paths between nodes.



Step 5 (Optional) : Enter costs for each path

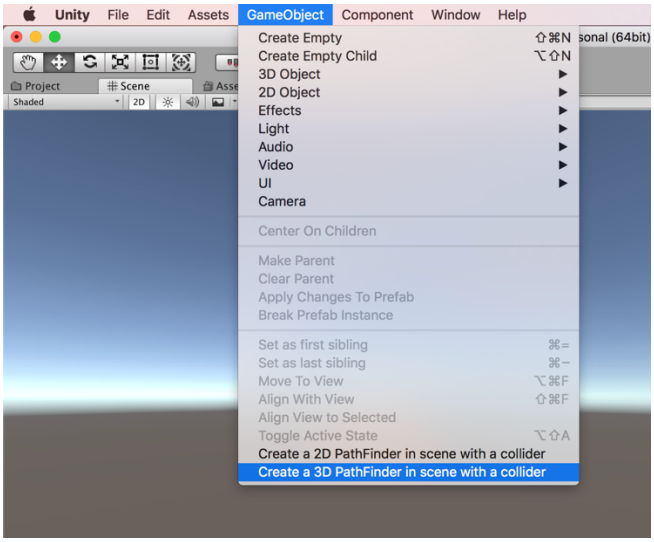
When u select PathFinder Gameobject, you can Enter the cost for each path.

Or set few Paths are one-way. One-Ways go from “From” to “To” Nodes.

How to create Nodes and Paths for 3D

Step 1 : Create PathFinder

Create PathFinder gameobjects and ground collider.

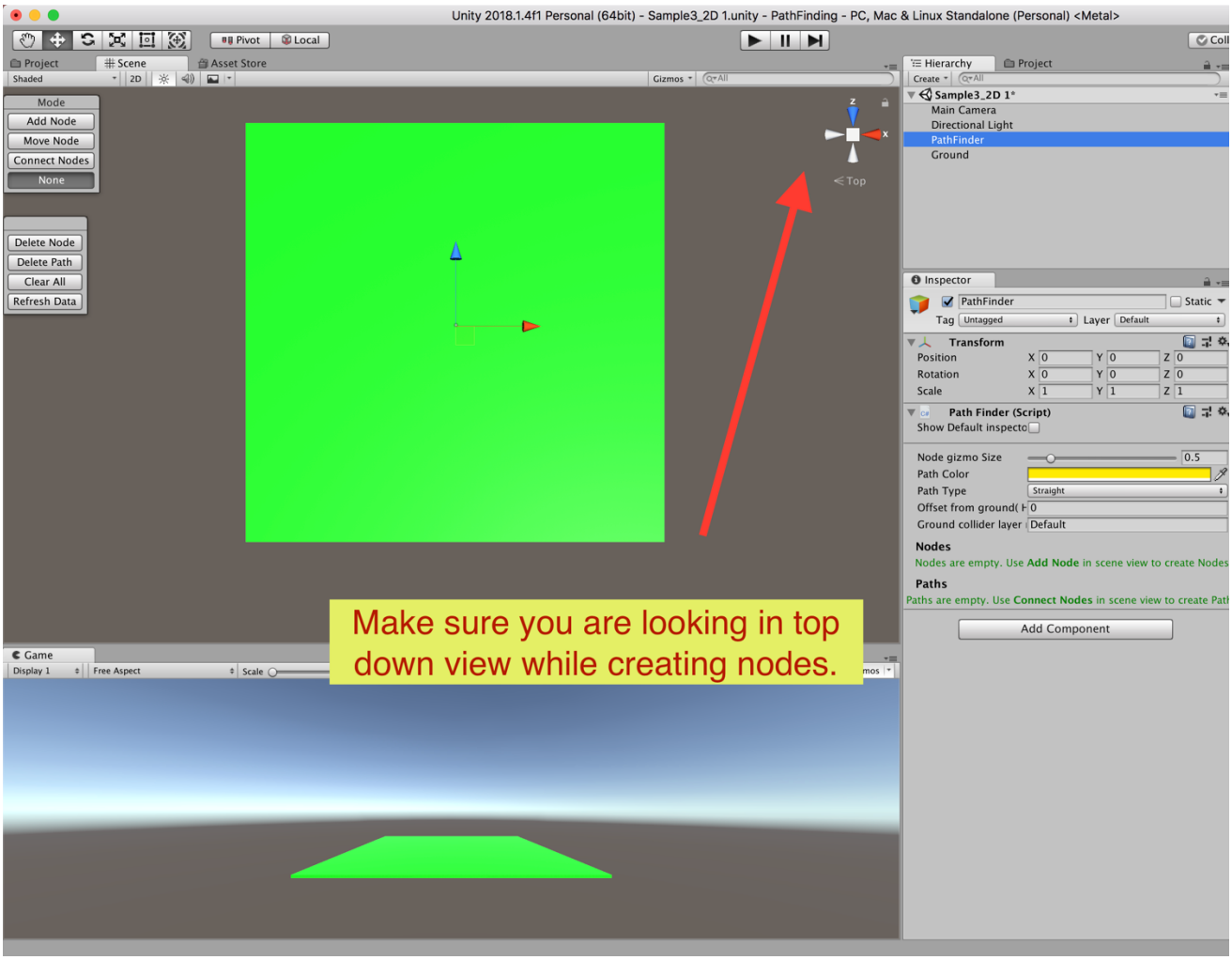


Step 2: Select PathFinder

Select "PathFinder" Game object from the Hierarchy. This will show the PathFinder GUI on the scene view.

Caution about 3D node creation:

Make sure you are in topdown view while adding nodes. This will make sure the nodes you create are exactly where they should be.



Step 3,4,5:

Remaining steps are same as in 2D. Please refer those.

How to Trigger PathFinder from code

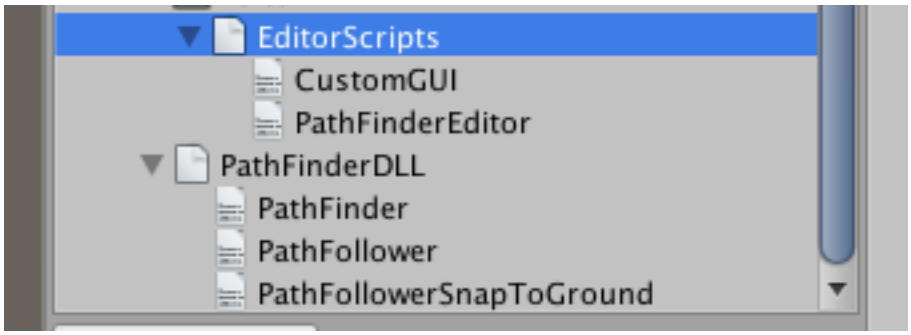
Once you have created all Nodes and Paths. Use PathFinder class to access all the methods necessary.

Use namespace QPathFinder.

```
using QPathFinder;
```

As long as you have PathFinder script in the scene, you can access PathFinder.Instance directly from the code.

Note: If you are using Free-Version, you can expand the pathfinder DLL in unity and find the scripts there. You can attach them to gameobjects just like any other scripts.



There are 3 Methods to fetch the paths PathFinder Class

1.

```
/// Finds shortest path between Nodes.
/// Once the path if found, it will return the path as List of nodes ( not positions, If you need positions, use FindShortestPathOfPoints ).
/// <returns> Returns list of **Nodes**</returns>
/// <param name="fromNodeID">Find the path from this node</param>
/// <param name="toNodeID">Find the path to this node</param>
/// <param name="executionType">Synchronous is immediate. This method locks the control till path is found and returns the path.
/// Asynchronous type runs in coroutines with out locking the control. If you have more than 50 Nodes, Asynchronous is recommended</param>
```

```
/// <param name="callback">Callback once the path is found</param>
```

public void FindShortestPathOfNodes (int fromNodeID, int toNodeID, Execution executionType, System.Action<List<Node>> callback);

2.

```
/// Finds shortest path between Nodes.
/// Once the path if found, it will return the path as List of Positions ( not Nodes, If you need Nodes, use FindShortestPathOfNodes ).
/// <returns> Returns list of **Positions**</returns>
/// <param name="startNodeID">Find the path from this node</param>
/// <param name="endNodeID">Find the path to this node</param>
/// <param name="pathType">Path type. It can be a straight line or curved path</param>
/// <param name="executionType">Synchronous is immediate. This method locks the control till path is found and returns the path.
/// Asynchronous type runs in coroutines with out locking the control. If you have more than 50 Nodes, Asynchronous is recommended</param>
/// <param name="OnPathFound">Callback once the path is found</param>
```

public static void FindShortestPathOfPoints (this PathFinder manager, int startNodeID, int endNodeID, PathLineType pathType, Execution executionType, System.Action<List<Vector3>> OnPathFound);

3. (Overloaded method)

```
/// Finds shortest path between Nodes.
/// Once the path if found, it will return the path as List of Positions ( not Nodes, If you need Nodes, use FindShortestPathOfNodes ).
/// <returns> Returns list of **Positions**</returns>
/// <param name="startNodeID">Find the path from this node</param>
/// <param name="endNodeID">Find the path to this node</param>
/// <param name="pathType">Path type. It can be a straight line or curved path</param>
/// <param name="executionType">Synchronous is immediate. This method locks the control till path is found and returns the path.
/// Asynchronous type runs in coroutines with out locking the control. If you have more than 50 Nodes, Asynchronous is recommended</param>
/// <param name="searchMode"> This is still WIP. For now, Intermediate and Complex does a tad bit more calculations to make the path shorter</param>
/// <param name="OnPathFound">Callback once the path is found</param>
```

public static void FindShortestPathOfPoints (this PathFinder manager, Vector3 startPoint, Vector3 endPoint, PathLineType pathType, Execution executionType, SearchMode searchMode, System.Action<List<Vector3>> OnPathFound);

Note that some return list of Nodes and some return list of positions. Based on what you need, you can trigger the methods which provide those.

Simple example

If I need to get path from node 1 to 10. Do this

```
PathFinder.instance.FindShortestPathOfNodes( 1, 10, Execution.Asynchronously, OnPathFound );
```

Once the result is found, callback will be called with list of nodes or positions.

```
void OnPathFound ( List<Node> nodes )
{

}
```

You can find more examples under QPathFinder/Samples/Scripts.

How to move along the Path (Optional) :

This is the next step after, we have got the path from PathFinder.

This is Optional. You have write your own path follower classes to move your character based on what your game needs. But I have included a simple path finder to do basic path following.

How to Trigger PathFollow from the code

1. To move an object along the list of positions.

```
/// <Summary>
/// This will move the game object through the points specified.
/// </Summary>
```

```
/// <param name="transform">The object you want to move along the path</param>
/// <param name="points">List of positions along which the object is moved.</param>
/// <param name="moveSpeed">Movement speed</param>

public static PathFollower FollowPath( Transform transform, List<Vector3> points, float moveSpeed
);
```

2. To move an object along list of positions, but also the object is snapped to the ground. So if you Nodes are a little above the ground, the object snaps to the ground.

```
/// <Summary>
/// This will move the game object through the points specified, Also, it will keep the gameobject
snapped to the ground.
/// So if your Nodes are a little above the ground, your target will still move on the ground.
/// We are doing this by raycasting from above the player to the ground. At the ray cast hit
position, we are snapping the player.
/// </Summary>
/// <param name="transform">The object you want to move along the path</param>
/// <param name="points">List of positions along which the object is moved.</param>
/// <param name="moveSpeed">Movement speed</param>
/// <param name="directionOfRayCast"> We use raycasting to find the ground position. If your
ground is down, the ray has to go down, so use Vector3.down. </param>
/// <param name="offsetDistanceToFloatFromGround"> If you want your character to float a little
above the ground, give the offset value here </param>
/// <param name="groundGameObjectLayer">This is the ground Gameobject's layer. When we use
raycast we target to hit this layer</param>
/// <param name="offsetDistanceFromPoint">this is to calculate the raycast origin, from where we
shoot rays. raycast origin is generally above the player, casting rays towards ground. For most
cases, you can leave this as default.</param>
/// <param name="maxDistanceForRayCast">this is the distance of ray from the raycast origin. For
most cases you can let this be default value. </param>


public static PathFollower FollowPathWithGroundSnap( Transform transform, List<Vector3> points,
float moveSpeed, Vector3 directionOfRayCast, float offsetDistanceToFloatFromGround, int
groundGameObjectLayer, float offsetDistanceFromPoint = 10, int maxDistanceForRayCast = 40 )
```

3. To stop a moving object.

```
/// <summary>
/// Stops the gameobject while moving along the path.
/// </summary>
/// <param name="transform">The gameobject which needs to stop moving</param>
```

```
public static void StopFollowing( Transform transform )
```

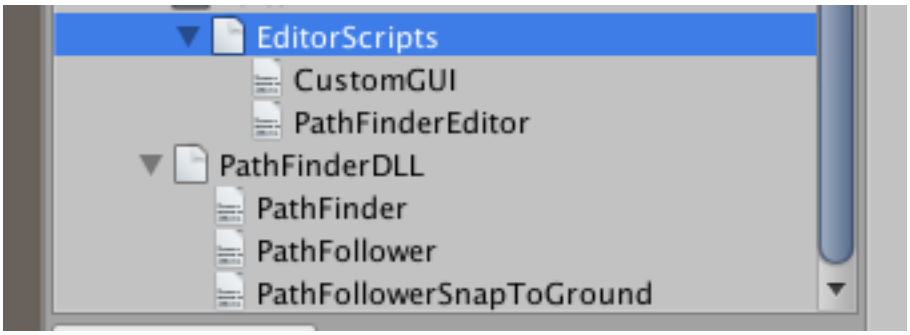
Note: All these Pathfollower methods are in PathFollowerUtility class. You can find more examples under QPathFinder/Samples/Scripts.

FAQ

To all the users using Free-Version

There is absolutely no difference between free-version and full-version except that I have provided dlls instead of source code. Other than that, you can do everything a full-version can do. There are no restrictions.

When you expand your DLL, you can find the class declarations there. You can drag these into scene just like any other script.



While trying to Create nodes, I am clicking on collider, but nodes are not created:

Make sure the collider name in PathFinder script and your collider layer name are the same.

Logging and other debugging

```
QPathFinder.Logger.SetLoggingLevel( debugLogLevel );

QPathFinder.Logger.SetDebugDrawLineDuration ( debugDrawLineDuration );
```

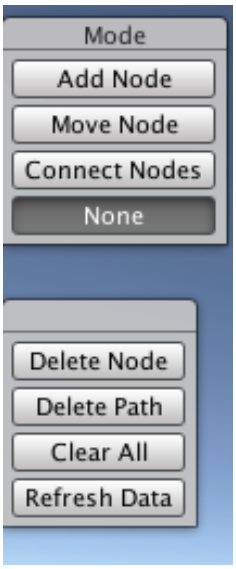
You can enable multiple levels of logs, using these.

Note: when log level is set to “Info”, you can also see Debug Lines drawn, which shows the path and other information using Lines in scene view and game view.

Note: Make sure you Turn on Gizmos.

I cannot see PathFinder GUI

If you cannot see this,



1. Make sure you select “PathFinder” gameobject in the scene. The GUI appears only on the selection.
2. GUI appears only in Scene view. Not in game view.
3. If you do not have PathFinder in scene, make sure you create one (the process is described above)

Cheers and thank you