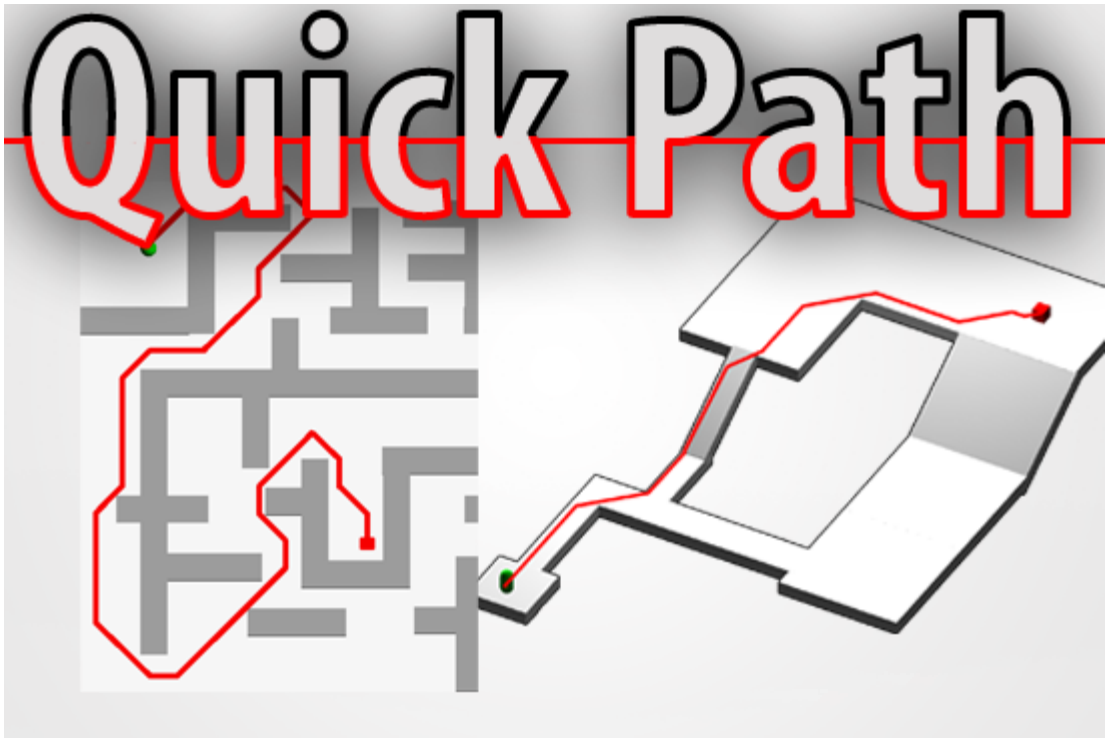
User Guide



Introduction

Quick Path is a quick and easy-to-use pathfinding solution built for Unity.
In this User Guide we will go through a step-by-step guide on how to use the asset.

If you have any questions you can contact me at alekhine@hush.com
If you want to see a quick video demonstrating how to use this asset click the following link.
https://www.vimeo.com/78474737
If you want to see Code Documentation click the following link.
http://www.alekhinedev.com/QuickPath/Documentation/annotated.html

Contents

Step-by-step Guide

To create a scene in which objects can move around we need nodes and we need move objects. Each node will have a list of connections which determine where a moveobject can go to as the move object will navigate through these node connections to find the quickest path using an A* algorithm.

Step 1: Generate Nodes (either through a grid or a waypoint network)
Step 2: Make a Moving Object

Step 1 – Generate Nodes(either through a grid or a waypoint network)

To make objects move you need nodes which the objects can navigate between, there are two ways to generate these nodes. One is via the qpGrid script which generates and places nodes based on its specified settings – this is a quick way as it generates thousands of nodes and node connections in a brief moment. The other one is via manually placing qpNodeScripts wherever you want nodes in your scene and then adding connections to each node – this is a slow and laborious way but gives you complete control of each node's placement and connections

Thus I present two possible methods of generating nodes:

A: How to make a Grid

1. Create an empty GameObject
2. Rename it "Grid"
3. Add the component "qpGrid" to it.

You should see the script shown below on your new GameObject. Here's a brief description of what each variable.

Draw In Editor: Should the grid be drawn in the editor?
Start Point / End Point: In the area in between these points there will be raycasts and nodes will be generated.
Up Direction: The direction the raycasts will go toward.
Highest Point: Where the rays will start.
Lowest Point: Where the rays will stop.
Node Spread: The distance between each node, if lowered there'll be more nodes placed closer together, if raised there'll be fewer nodes placed further apart.
Disallowed Tags: No node will be generated where a ray collides with a GameObject tagged with any string from this list.
Ignore Tags: Similar to Disallowed Tags, however a node may still be placed if the ray collides with another GameObject, which isn't tagged with any string from neither Ignore Tags nor Disallowed Tags.

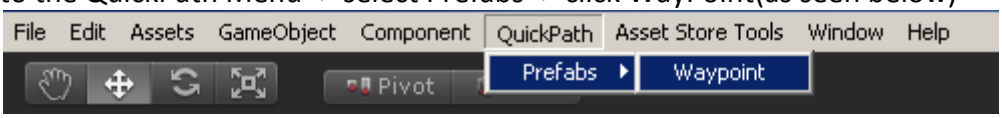
The Grid will be automatically baked at the beginning of the scene when its run. But you can examine how your grid looks by baking it before running it.

Now customizes the qpGrid to fit your needs, once this is complete you should have the necessary nodes to start creating move objects.

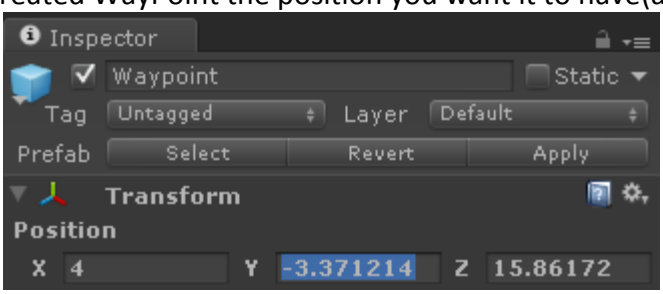
Relevant scenes:
3dGrid, 2dGrid and AstarTests.

B: How to make a Waypoint Network

1. Go to the QuickPath Menu -> select Prefabs -> click WayPoint(as seen below)

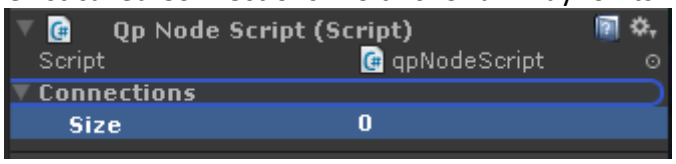


2. Give the newly created WayPoint the position you want it to have(as seen below).

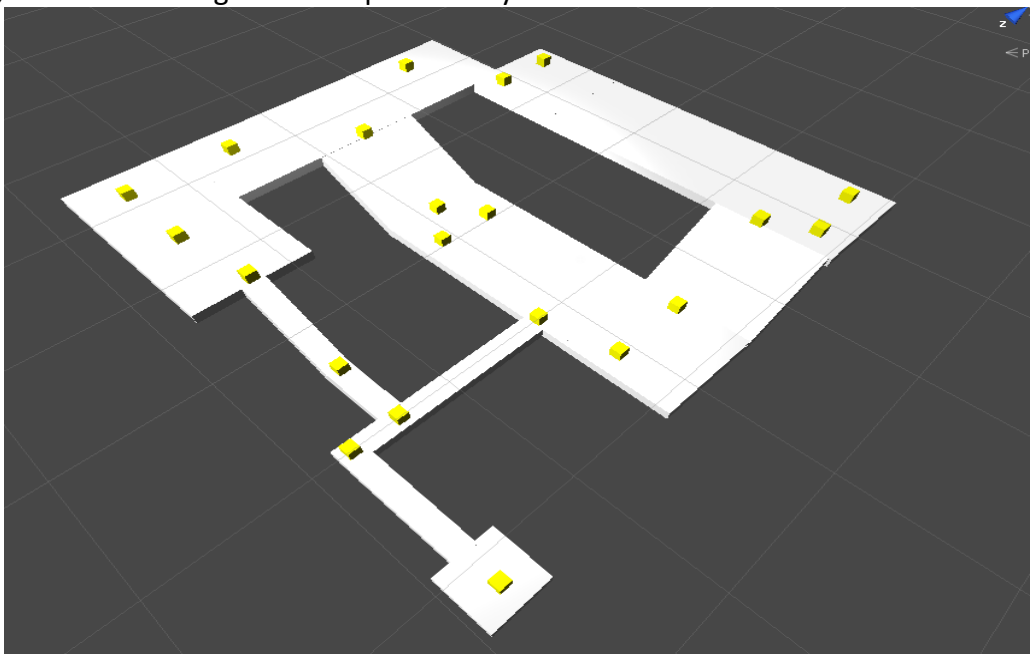


Continue Step 1 and 2 until you have all the nodes you want, then continue to step 3.

3. Select the first WayPoint, and drag and drop all other nodes you want to be connected to this node into the list called Connections. Do this for all WayPoints.



You should now have a scene containing connected waypoints, and thus your WayPoint should be complete, below is an image of a completed WayPoint network.



Relevant scenes:
3dWayPoints.unity

Step 2 – Make a Moving Object

There's a range of predefined Move Objects available you can see these at the 'List of Predefined Move Objects' Chapter. If you wish to use any of these, simply create a new object and give it your desired predefined move object component, and you should be functional.

If you wish for a more advanced move object you can create your own. If you do this I suggest you make your new custom class inherit from qpMoveObject, and remember to call overridden base methods including Start and FixedUpdate.

List of Predefined Move Objects

There is a range of Predefined Move Objects available in this Asset Package.
These are :

- qpMoveObject
- qpFollowTargetObject
- qpFollowMouseObject
- qpPatrolObject

The commonality of the last 3 in the list is that they inherit from qpMoveObject, the base class.

qpMoveObject

This is the base class which contains the most basic instruments for making an object navigate and move around.

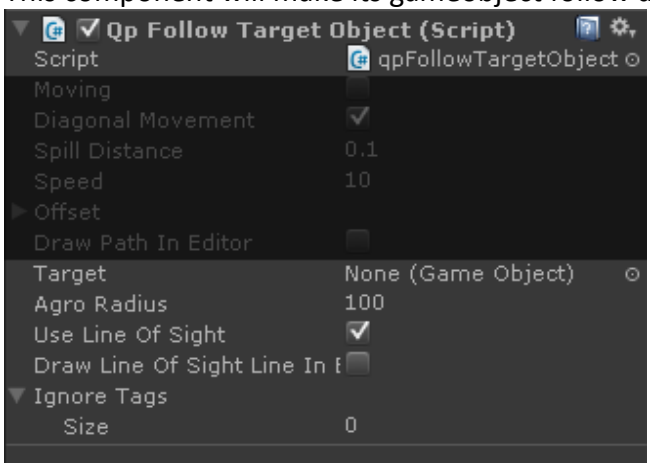
Moving: Is this object currently moving?
Diagonal Movement: Should this Object move Diagonally on grid nodes?
Spill Distance: How close this object has to be to a node before it reaches its destination.
Speed: How fast this object moves.

Offset: If the anchor point of this object is not 0,0,0 you can define it here.

Draw Path In Editor: Should the current path be drawn in the editor?

qpFollowTargetObject

This component will make its gameobject follow a designated Target.

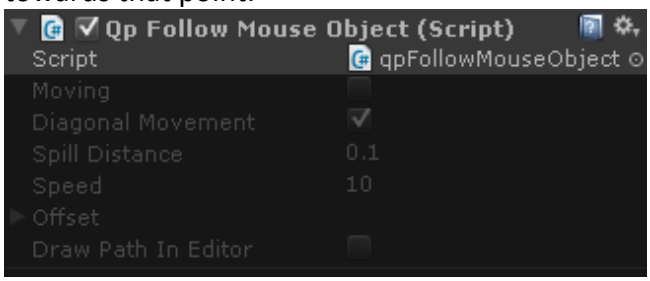


Agro Radius: The radius within which it will follow its target.
Use Line Of Sight: Should this object use raycasting to test for line of sight, and only chase when the target is in line of sight?
Draw Line Of Sight in Editor: Should the raycasting to test of line of sight be drawn in editor?
Ignore Tags: These tags will be ignored when using line of sight raycasting.

Target: The target that this object will follow.

qpFollowMouseObject

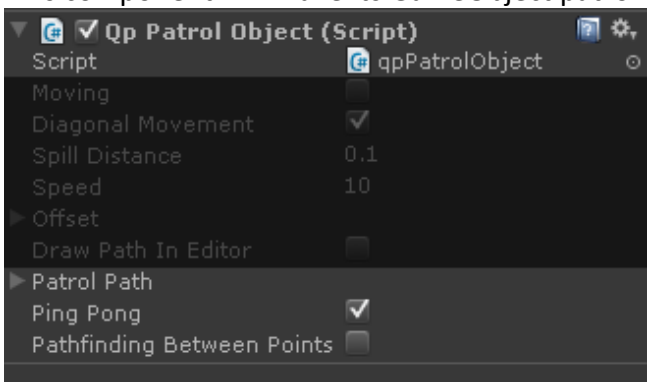
This component will make its GameObject go to the point where the mouse is, when the mouse is clicked. It will do this by making a raycast at the mouse position, and then testing for collisions, if the raycast collides this component will find the node closest to the collision point and build a path towards that point.



This class has no additional public properties aside from what those it has inherited from its super class qpMoveObject.

qpPatrolObject

This component will make its GameObject patrol between a list of given coordinates.



Patrol Path: A list of coordinates, the object will patrol between these coordinates starting with the cordinate at element 0.
Ping Pong: Should this object reverse the path and walk it when it has completed a path?
Pathfinding Between Points: Should an A* search be performed between each point in the list to find the quickest path between each point?