Tiles Based Map & Nav

version 2.3
by Leslie Young
http://www.plyoung.com/
http://plyoung.wordpress.com/

More info on this package and links to videos are available on the Unity Forum at. I will only go over key topics in this document. It is best to watch the videos, linked on the Unity Forum.
http://forum.unity3d.com/threads/138355-Tile-Based-Map-amp-Nav

Table of Contents

Introduction

Map & Nav will help you get started with a game where you need a tile based grid system. This is especially useful for strategy games and board games.

Feature list

- Full source code
- Hexagon and Square tile layouts
- Tools to quickly create new layouts/maps
- Custom inspector and Editor windows
- Easy setup of 'tile-type masks' (ex. land, floor, water, wall)
- Variable Height Nodes & Tools to easy setup of node heights
- Unit movement options (hug/align to floor/terrain mesh and jump)
- iOS/Android supported
- Various sample scenes

Updates

Version 2.3
Changed and optimised the way TNEMovementModifier works. Added ability to turn on or off the link between two nodes (TNELinksOnOffSwitch). Useful for doors or even cases where you permanently want the link between two nodes disabled. Updates sample4 (dungeon) to show how a door could be implemented.

Version 2.2
The TileNode prefabs are no longer hard-coded in the editors, you can now specify the Prefab to be used. Improved Auto Mask-Setup Tool. Movement modifier options added, this will allow you to set certain tiles to cost more movement points to move onto.

Version 2.1
Variable height tile-nodes was added and new movement options for units.

Version 2.0
Release of version 2, which is a total rewrite and clean-up from version 1, while adding square tile layout support and the Radius Marker object.

First Steps

The first thing you need to do is sort out the Layers. For the sample scene I set layer 20 to “tile” and layer 21 to “unit”. Edit this by going Unity menu → Edit → Project Settings → Tags, and updating user layers ..
20 = “Tile” and 21 = “Unit” (for all samples)
22 = “Terrain” (for samples 2 and 3)
23 = “Floor” (for sample 4)
23 = “Wall” (for sample 4)

This package makes use of iTween to move units around. You can choose to delete the included script if you already have this script in your project. It is saved at “\Tile Based Map and Nav\Scripts\Common”. You can find the latest version of iTween on the Unity Asset Store.

~~Please update the PREFABS_PATH variable in \Tile Based Map and Nav\Editor\MapNavEditor.cs” is you move things around or rename folders.~~ (not needed since v2.2)

The Sample Scenes

Sample01 shows the basics of the package with units moving around on a flat-plane terrain. It also shows a possible way of handling turns and how you could implement units that can attack other units.

Sample02 is the 1st of the samples showing how tile nodes can be set at variable heights with units hugging the terrain and rotating according to terrain mesh normals when moving around.

Sample03 is the 2nd variable height tile node sample which shows the Jumping movement feature of units.

Creating a new Grid Layout

Select from Unity menu → Window → Map and Nav → New MapNav
A window will come up where you need to set a few things.

The following fields are also present in the NavNode Inspector.

Tile Layer: is the layer that tiles sits on. (layer 20:tiles in the sample)
Units Layer: is the layer for units. (layer 21:units in the sample)
(You need to set these else you won't be able to click on Units and Tiles.)

The following do not have to be set, but can be if you want to create a layout now and not via the Inspector.

Tiles Layout: Select the layout you would like to use.

- Hex: is hexagon layout of tiles and tiles themselves would be in a hexagon form. In this layout each tile node links with up to 6 neighbouring tiles.
- Square (4): is a square layout of tiles. Each tile node links with 4 other tile nodes, thus ignoring neighbouring tiles that lays diagonally from the specific tile.
- Square (8): is a square layout of tiles where each tile node links with up to 8 neighbouring tiles around it.

(The linking of tile nodes helps the path finder figure out how units may move over nodes)

Tile Spacing: How far each tile node should be spaced from its neighbours.
Tile Size: How big each tile node should be. This is influences not only how big the tile renders but also the underlying collider that is used to detect clicks on a tile.

Initial Tile Mask: The initial mask to apply to each tile node's tile layer/level mask. Please node that these values can later be changed per tile, so only put here what you would like the default/initial values to be.

This mask is what defines what a tile is or what kind of units may move over/onto it. You can define more values in the TileNode.cs script by making changes to the **TileType** enum. You will notice that The default values are Normal (land), Air, Water and Wall. So, a tile that includes Normal and Air in its mask would allow normal (land) and air units to move over it. The MapNav system will allow both an air and a land unit to occupy the same tile node at the same time, seeing as they are on different layers/level (air and normal). If you wanted for example underwater unit support, you could easily add another value in the TileType enum, for example Underwater=16; and then set the tile nodes that represents water, to Undewater | Water | Air; which would mean 3 units could occupy the same tile at a time. MapNav will not allow units of the same tile-type to occupy or move through tiles that already contain a unit of that type. For a unit you set the tyle type in its tileLevel field. See NaviUnit.cs.

Width x Length: These values indicate how big the grid should be.

Hit create and your new MapNav grid will be created.

The provided prefabs for hexagons and squares are created such that the white marker you see is offset a little on the y axis. This is so that the art for the marker don;t show up inside your terrain that might be sitting at height(y)=0. Please check out the FBX files in your favourite 3d modelling tool to see what I've done with the pivots.

Radius Marker

You might have noticed there is another option on the menu under “Map and Nav”, the option to create a “New Marker”.

The radius marker is something you can choose to use if you like and I've included this tool to help make such a marker easily. In the sample scene I've used it to indicate the tiles where a selected unit can attack at.

It is basically a grid layout in hex or squares with a certain max radius. With script contains a few functions which you can then use to show markers up to a certain radius from the center of the marker, or show markers.

The options when creating a new marker are the same that you will see in its Inspector.

Marker Node Prefab: The prefab to use for each marker node. This can be something simple like a plane with a texture on it.
Marker Layout: Choose the layout, same as with NavMap creation.
Marker Node Spacing: How far each node is spaced from the other.
Marker Node Size: Size of each node.
Marker Radius: How far out, from center, should the nodes extend.

Have a look at attackRangeMarker in GameController.cs to see how I used this to indicate what tiles the selected unit can attack at.

MapNav (the grid)

MapNav is the grid layout and contains a bunch of TileNodes. You can click on each of these TileNodes to edit their Tile Type Mask values. Also make sure that these TileNodes are sitting in the same layer you defined as the “Tile” layer and set in the MapNav.tilesLayer proeprty.

In MapNav's Inspector you will find a few useful tools.

Gizmos

You can select to show a little cube gizmo where each node is and also to show/hide the links between nodes. The colour coding is useful to quickly see what kind of tile type masks each of the tile nodes are set to. Makes ure to check out and update TileNode.GizmoColourCoding as needed if you made changes to the TileNode.TileType enum.

Create Tool

This tool is used for creating the tile nodes of the MapNav. This will destroy any nodes currently in the MapNav and create new ones with the selected values. Please see the previous topics for a description of the properties here.

Marker Tool

This simple tool allows you to hide or show the markers of the tile nodes. Remember, markers are not gizmos. The gizmos are useful to check out the tile nodes in the editor, but Markers are what you will see in the game when a player click on a unit. They are the white markers that comes up in the sample to show where a unit may be moved.

Mask Reset Tool

This simple tool simply reset the masks for all tile nodes to the specified mask.

Mask Setup Tool

This tool can save you a lot of time if you setup the scene in a specific way. Its job is to run through a bunch of transforms, cast a ray down from each and check what tile nodes are hit; when it hits a node it will set to or add to that node's mask the mask you specified.

Lets say I wanted to automate the setup of masks for nodes under trees. I would make sure to contain all tree meshes inside one GameObject. I then drag this trees container to the “Parent Object” property of the tool, select “Walls” from the “Mask” drop-down and hit “Set” so that the masks for the nodes under the trees are set to this new mask. “Add” would add to the mask, which I do not want since by default all node masks are set to (Normal + Air) and just adding (Wall) would mean that normal(land) and air units could actually still move onto those nodes seeing as they still 'allow' those kind of units.

Since v2.1: There is now also an option where you can test against objects on a certain layer. This us useful where your object/transform might occupy more than one tile node, in which case the above description would only change the mask for one of the tiles. With this new method you need to provide a collider since it will cast rays from the nodes to check against colliders/objects on the specified layer and then change the tile type mask of the node if needed.

Height Setup Tool

This tool is used to help setup the height of nodes when you have a floor (terrain) that is not flat. Make sure that the mesh for your floor or terrain has a collider that follows the contour of the mesh and that this object is using some unique layer, for example layer 22 (named Terrain) in the examples.

In the tool you can then select this layer (22, Terrain) and hit 'Run Setup' and the tile nodes' heights will be updated. Use the "Node Deletion" option to get rid of nodes that are not perfectly aligned over the terrain.

TileNode Extensions

You simply place these onto nodes that must be extended (these extensions are components), and the game systems will take care to query for the extensions that they know how to handle. For example, the path calculator will ask for the *TNEMovementModifier* since to see if additional movement points are needed to move over tile nodes.

TNEMovementModifier

This simple extension is used to modify the movement points of a unit when it moves onto a tile. This way you can make tile nodes that require two or more movement points to be available before a unit may move onto them.

TNELinksOnOffSwitch

This TileNode specific component (extension) will allow you to switch the link between two neighbouring nodes to on or off.

eof