HexMap Tools

Thank you for buying HexMap Tools.

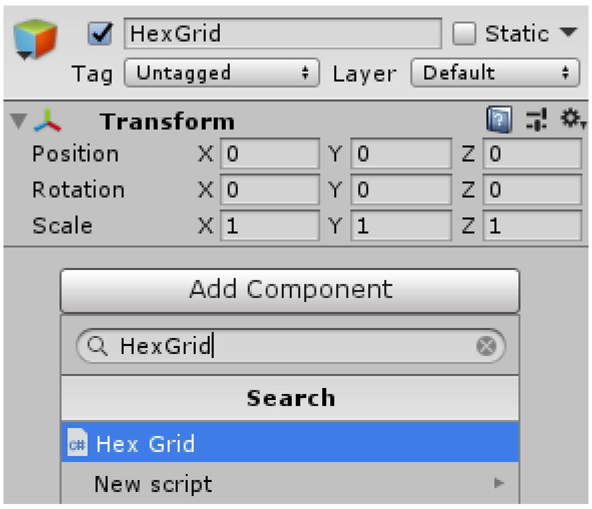
The asset is a set of classes that are supposed to make creating games, based on hexagons easier.

The main features of HexMap Tools are:

- align objects to customizable hex grid
- brush for hex prefabs
- customizable path finding(A* implementation)
- container for hexes
- conversion between world position and hex coordinates
- hidden math and many utility functions

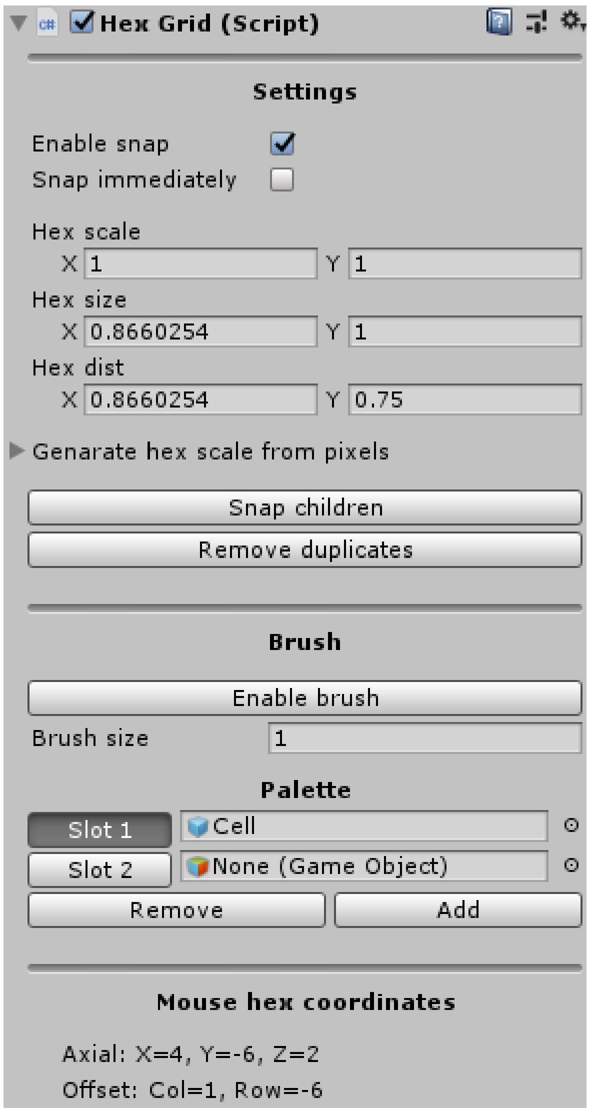
Creating hex grid

To create hex grid, add HexGrid component to your chosen GameObject. This object will become a container for your hexes.



HexGrid ☐ 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



✓ Hex Grid (Script)

Settings

Enable snap ✓
Snap immediately ☐

Hex scale
X 1 Y 1
Hex size
X 0.8660254 Y 1
Hex dist
X 0.8660254 Y 0.75

Brush

Brush size 1

Palette

Slot 1 Cell
Slot 2 None (Game Object)

Mouse hex coordinates

Axial: X=4, Y=-6, Z=2
Offset: Col=1, Row=-6

Settings

You can configure grid in HexGrid inspector under section Settings.

Enable snap – objects which are direct child of container will be automatically snapped to grid, when selected.

Snap immediately – snap mode, changes how snapping looks.

Create prefab connection – If selected, brush creates prefab connections, many connections can(in few cases) strongly slow down editor and undo operation. It is best to keep it false for more than 200 objects in grid.

You have two options to configure size of hexes:

1. Easy to use hex scale generator under “Generate hex scale from pixels”

- **Hex size** – hex size in pixels, width, and height,
- **Overlapping pixels** – if you want your tiles to overlap you can set how many pixels will be overlapping,
- **Distance** – if the settings above doesn't work for you because your hexagons aren't regular you can set horizontal distance between hexes in the same row and vertical distance between hexes in the same column. It may be useful for small pixel art hexes(e.g. 10x10 pixels) This property is calculated/connected with Overlapping pixels.

Warning:

Generator only works if you have set Pixels Per Unit in sprite settings to be the same as height of the hex sprite.

After settings you can click Generate button.

2. If you have done 'complicated' math, or you want to configure at a guess:

- **Hex scale, Hex size** – by default hex is 1 unit high and $\frac{\sqrt{3}}{2}$ unit wide, with this setting you can change its scale/size. Those settings are calculated one from another, so change in one field will affect the other one,
- **Hex dist** – distance between centre of neighbouring hexes, calculated from settings above.

There are also two buttons:

- **Snap children** – snaps all children to the grid,
- **Remove duplicates** – removes all hexes that are on the same coordinates.

Brush

HexGrid component allows to paint with objects on the hex grid. Painted objects will be created as children of container and will be automatically snapped to grid.

To enable brush press "Enable brush" button. After that when you move mouse over scene view you will see on which position object will be created. To create object press left mouse button.

- **Brush size** – Changes brush radius. For performance reasons the biggest radius is 10.

- **Palette** – For each slot you can select prefab. To change slot press “Slot” button or press numeric keys on top of the alphanumeric keyboard.
    - **Add** – creates new slot for prefab.
    - **Remove** – removes bottom slot.

Replace

This section allows you to replace objects on grid by a name with a new object.

Mouse hex coordinates

This section shows hex coordinates, on which is actually mouse. It works only for scene view.

- **Axial** – mouse position in axial coordinates.
- **Offset** – mouse position in offset coordinates.

Path finding

To find path between given coordinates, firstly you need to create instance of HexPathFinder class.

Path finder doesn’t know which hexes are accessible and it doesn’t know costs between them. As a result the first and the most important argument of the constructor is delegate, that takes two HexCoordinates and returns float. This method is supposed to return cost from the first hex to the second(given hexes are adjacent). If second hex is inaccessible it is supposed to return infinity.

When instance of HexPathFinder is created, you can call on it ‘FindPath’ method.

It returns bool, whether path was found and takes three arguments: start hex, target hex and List<HexCoordinates> as an output, which returns path to the target. If path wasn’t found method returns false and List<HexCoordinates> contains path to the closest found hex to the target.

To learn more about HexPathFinder you can read ‘Scripting’ section, comments in the source code or check ‘PathFinding’ demo scene.

Scripting

HexMap Tool defines the following structures with their most important methods:

struct HexCoordinates – stores hex coordinates, can be used as a key in Dictionary

- `HexCoordinates(int x, int z)` – creates from axial coordinates
- `HexCoordinates FromOffsetCoordinates(int x, int y)` – creates from offset coordinates.
- `int Row, Col` – get offset coordinates
- `int X, Y, Z` – get cube/axial coordinates
- Vector operations with overloaded operators: Add(+), Subtract(-), Scale(*), Length

```
enum HexDirection { NE, E, SE, SW, W, NW }
```

static class HexUtility – utility methods to operate on hexes

- `int Distance(HexCoordinates a, HexCoordinates b)`
- `HexCoordinates GetDirection(HexDirection direction)`
- `HexCoordinates GetNeighbour(HexDirection direction)`
- `HexCoordinates[] GetNeighbours()`
- `HexDirection NeighbourToDirection(HexCoordinates coords, HexCoordinates neighbour)`
- `List<HexCoordinates> GetRing(HexCoordinates center, int radius)`
- `List<HexCoordinates> GetInRange(int range)`

class HexCalculator – converts HexCoordinates to world position and vice versa

- Vector3 HexToLocalPosition(HexCoordinates coords) – Converts hex to local position of container
- Vector3 HexToPosition(HexCoordinates coords) – Converts hex to world position
- HexCoordinates HexFromLocalPosition(Vector3 pos) – Converts local position of container to hex
- HexCoordinates HexFromPosition(Vector3 pos) – Converts world position to hex

class HexGrid – stores grid configuration

- HexCalculator HexCalculator – property, gets HexCalculator instance

class HexContainer<T> - stores objects, which are accessed by HexCoordinates, overloads operator [] and implements IEnumerable<KeyValuePair<HexCoordinates, T>>. It is based on Dictionary and as a result there is no need to set size of the grid. It can store only Unity components(or any class derived from MonoBehavior). Those restrictions allows for some additional operations.

- HexContainer(HexSnapComponent hexSnap, bool autoSetCorrectPosition = true, bool autoDestroyGameObject = false) – Creates new HexContainer
  - autoSetCorrectPosition – If true sets position, calculated from key when inserting object
  - autoDestroyGameObject – If true automatically destroys GameObject when removing from the container
- void FillWithChildren() - Fills with all existing children of HexSnapComponent
- T At(HexCoordinates coords)
- void Insert(HexCoordinates coords, T newHex)
- void Remove(HexCoordinates coords)
- IEnumerable<T> GetCells()

class HexPathFinder – finds path between chosen HexCoordinates

- delegate float HexCostFunc(HexCoordinates start, HexCoordinates target) – Delegate that returns edge cost, cost from the first hex to the second hex. It returns infinity if the second hex is inaccessible

- `delegate float HeuristicFunc(HexCoordinates start, HexCoordinates pathDestination)` - Delegate that returns estimated cost of move between given coordinates.
- `HexCostFunc HexCost` – Property: get, set. Function used to return cost of move to the adjacent hex.
- `HeuristicFunc Heuristic` – property: get, set. Function used to estimate cost of move between hexes
- `float HexCostModifier` – Property: get, set. Modifies terrain cost.
  - 0 – all hexes cost 1 (or are inaccessible if cost is infinity)
  - 1 – normal costs are used
- `float HeuristicWeight` – Property: get, set. Modifies the weight of heuristic
  - 0 – heuristic is ignored
  - 1 – terrain cost and path length is ignored
- `int MaxIterations` – Property: get, set. Defines how many iterations(roughly equals to the amount of checked hexes) will be performed until algorithm decides that the path wasn't found.
- `PathFinder(HexCostFunc hexCostFunc, float hexCostModifier = 1f, float heuristicWeight = 0.5f, int maxIterations = 10000, HeuristicFunc heuristicFunc = null)` – To create path finder instance you have to pass HexCostFunc. This function is supposed to decide which hexes ale accessible and which aren't. To learn more about other arguments check HexPathFinder properties.
- `bool FindPath(HexCoordinates start, HexCoordinates end, out List<HexCoordinates> path)` – returns true if path from start to end hex was found. Argument 'path' returns path to end hex or to the closest found hex to end. Method uses A* implementation.

Examples

To better understand how to use HexMap Tools, there are included some example scenes in *HexMap Tools/Examples* directory.

- Hexxagon – simple game based on Hexxagon
- PathFinding – path finding demo

Support

I hope that HexMap Tools will help you to create you dream game.

If have any question or find a bug please feel free to contact with me. I plan to further develop HexMap Tools so any feature suggestion would be also appreciated.

nqprojectstudio@gmail.com