HexMap Tools

Thank you for buying HexMap Tools.

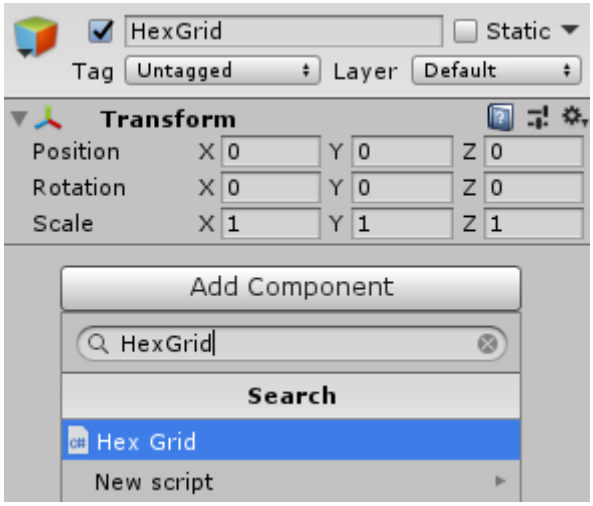
The asset is a set of classes that are supposed to make creating games, based on hexagons easier.

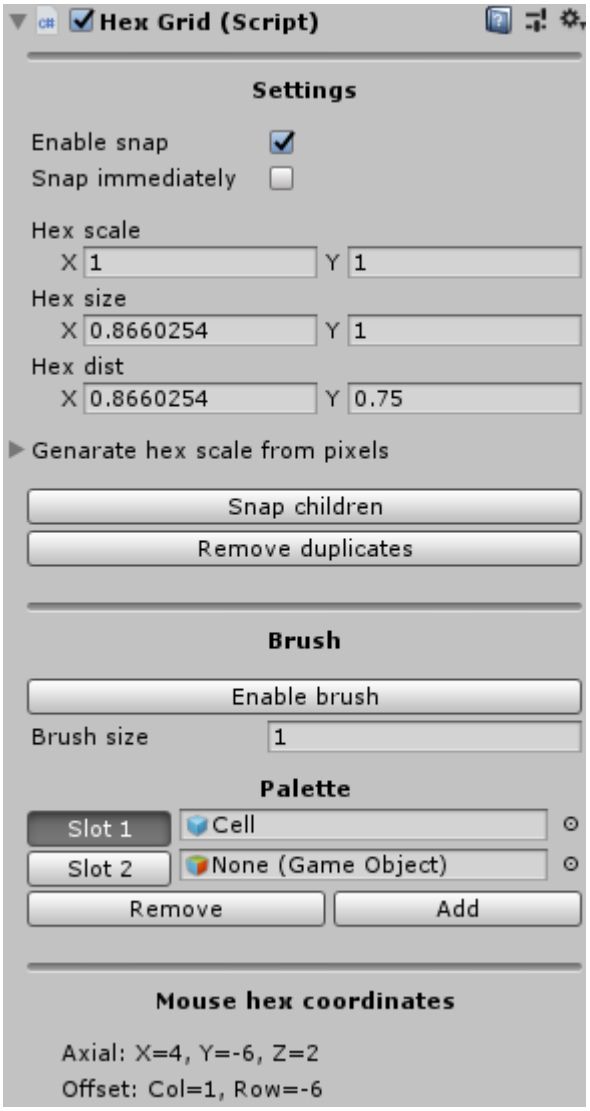
The main features of HexMap Tools are:

- align objects to customizable hex grid
- brush for hex prefabs
- 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.





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.

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 prefabs on the hex grid. Painted prefabs 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 prefab will be created. To create prefab 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.

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.

Scripting

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

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

struct **HexCoordinates** – stores hex coordinates, can be used as 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

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 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()

Examples

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

- Hexxagon – simple game based on Hexxagon
- NeighbourTest – sample scene with script showing how hex directions work

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