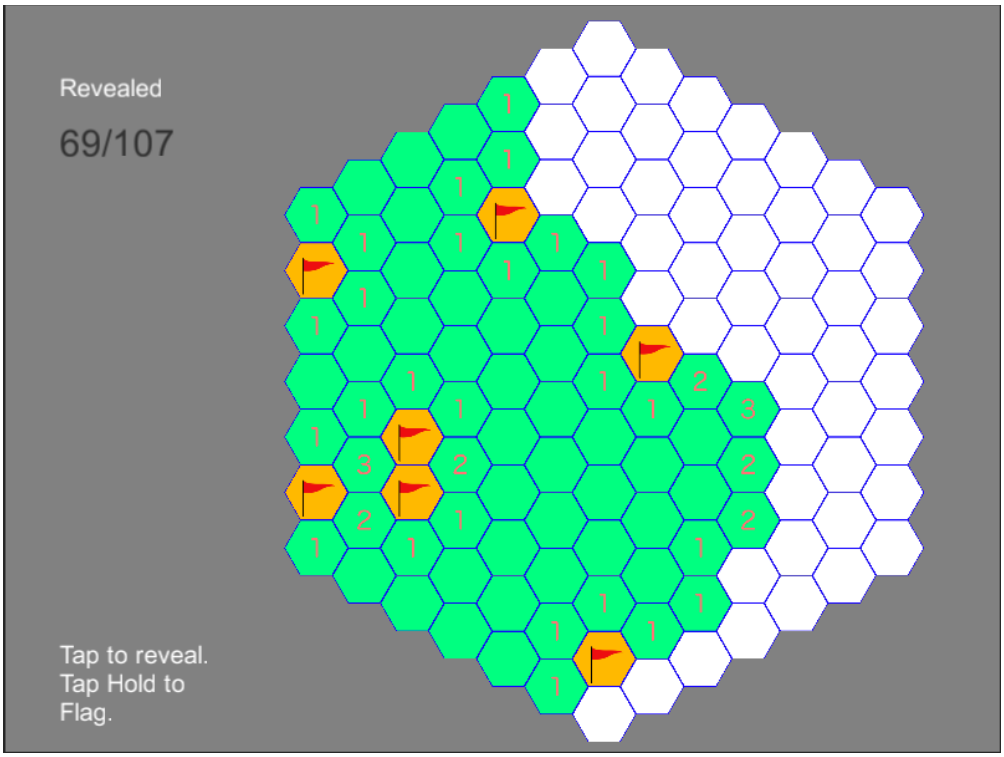
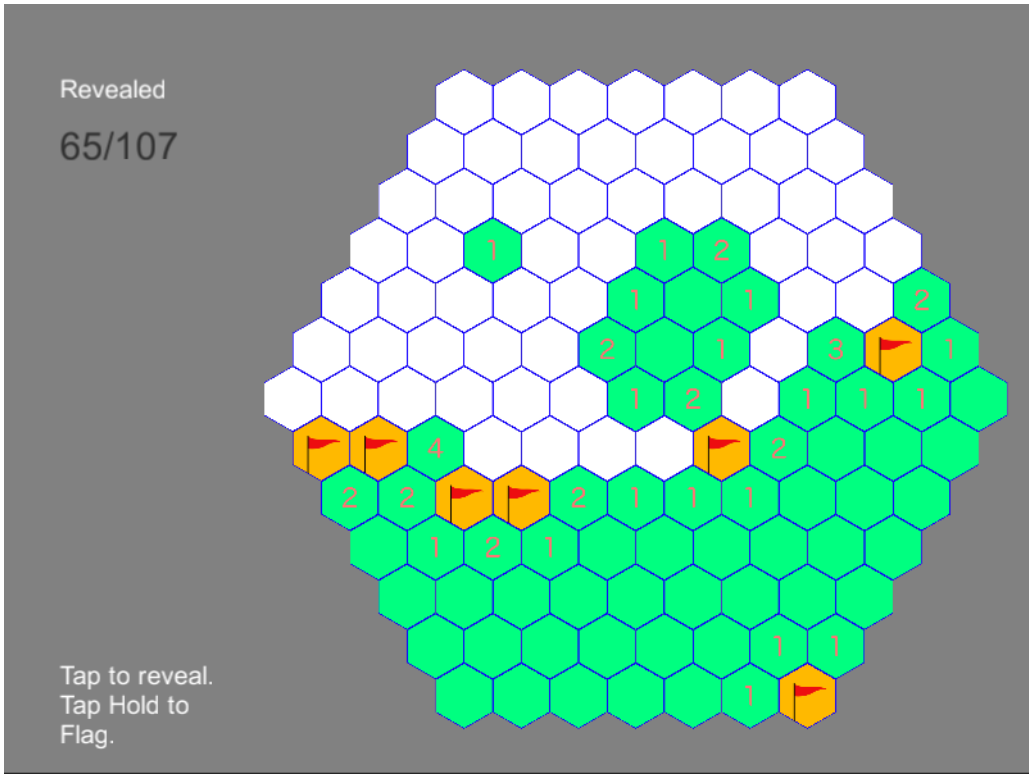
Hexagonal Minesweeper Game (full project) by Juwal Bose





Game Play

Classic minesweeper game but using hexagonal grid. There are 2 different layouts available, horizontal & vertical.
This project can be used for learning purposes as well as to extend the functionality to create commercial games.

Features

- Hexagonal helper classes for vertically aligned & horizontally aligned grids with odd offset
- Helper classes do hexagonal coordinates conversion (offset, axial, cubic)
- Helper classes do screen position to axial coordinate & axial coordinate to screen position conversion
- Helper classes find out the neighbours of a given hexagonal tile
- Minsweeper game logic
- Horizontal & Vertically aligned hexagonal layouts handled
- Grid can be of any shape or size by simply altering the level array
- Fully commented c# source code
- Works on PC/MAC/IOS and Android (WebGL not tested, but should work)
- Beginner friendly

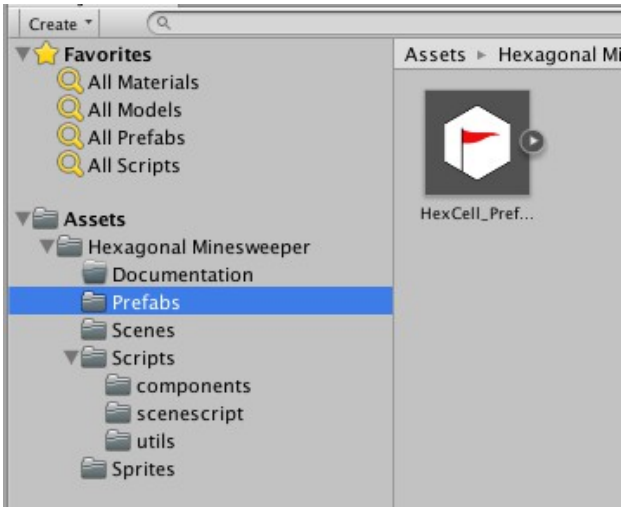
Video of game play - https://youtu.be/r5V75nWsj0I

Support email – juwalbose@gmail.com

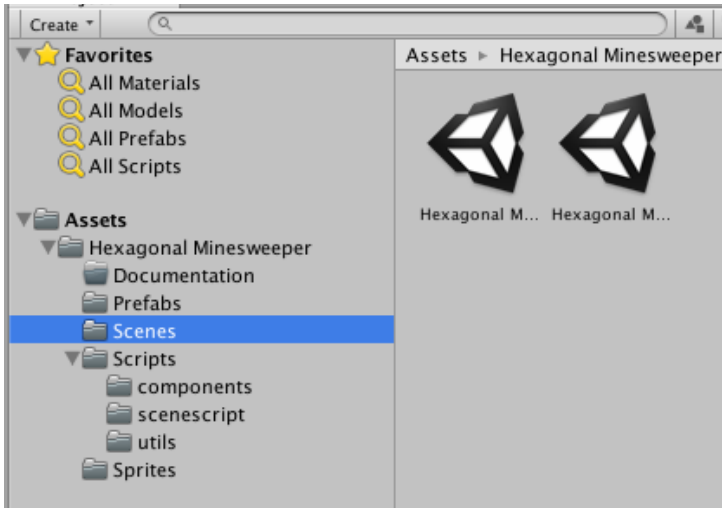
Project

The project folder structure is as shown below. Each folder is explained in the following sections.

The **Prefabs** folder has HexCell_Prefab item for the hexagonal cell component.



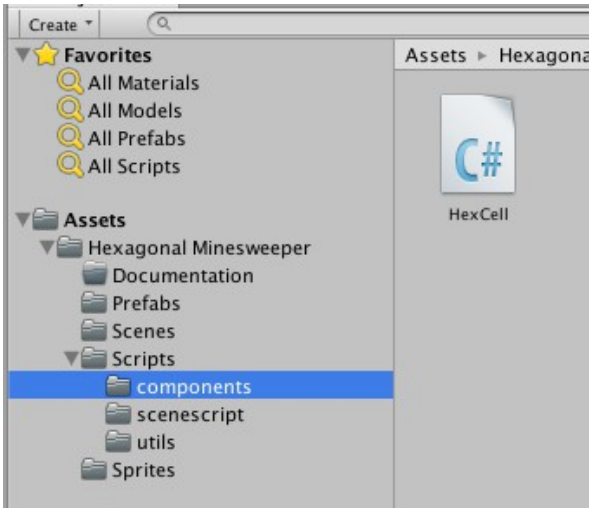
The game scenes can be found in the **Scenes** folder. There are 2 scenes one for horizontal layout & one for vertical layout.



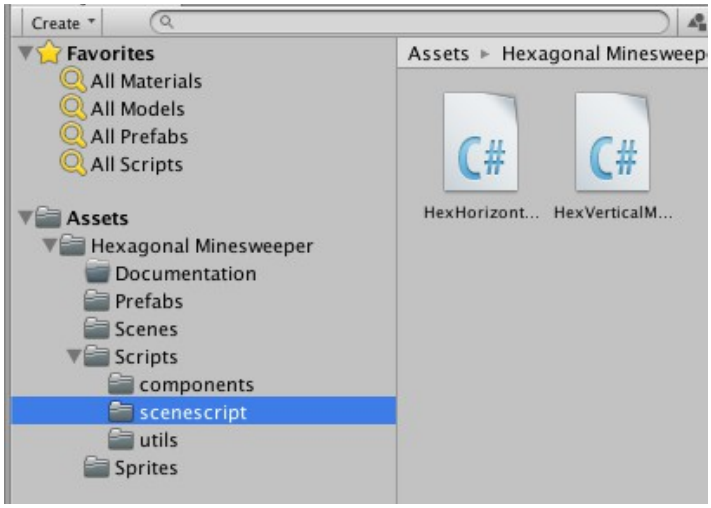
The **Sprites** folder has the few art assets used in the project.

A hexagonal cell can have a maximum of 6 neighbours, hence can have 6 neighbouring mines. So we have images for numbers 1-6 to be shown inside the cell to indicate neighbouring mines as well as the mine & flag images.

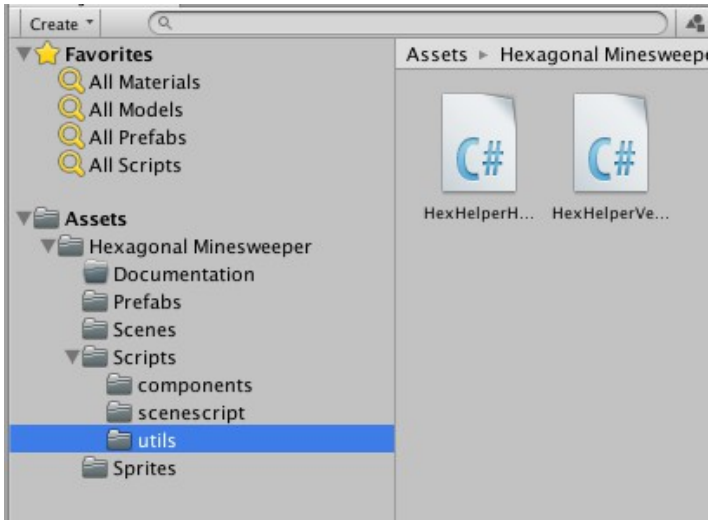
The **Scripts** folder has 3 folders which has the various c# scripts used in the game. The **components** folder has script for the **HexCell** prefab component.



The **scenescript** folder has the main game scripts which is attached to the camera in the game scenes. These **main game scripts are mostly identical** other than the **usage of the different HexHelper classes** & the **transposing of levelData array**.

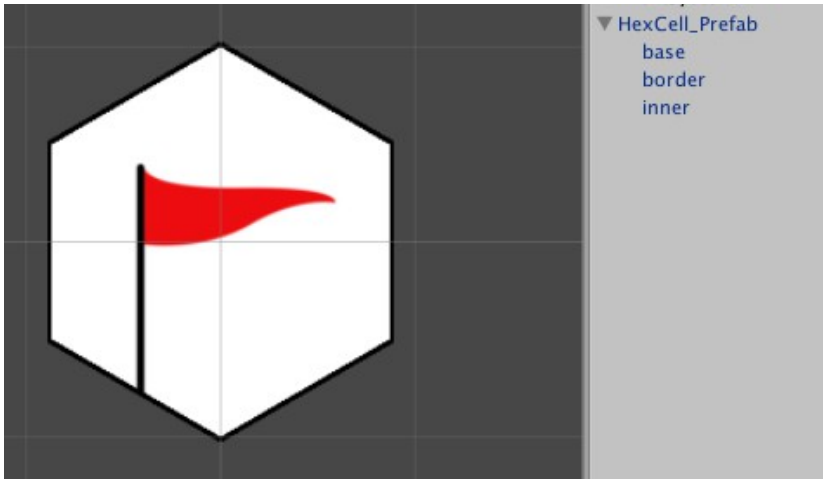


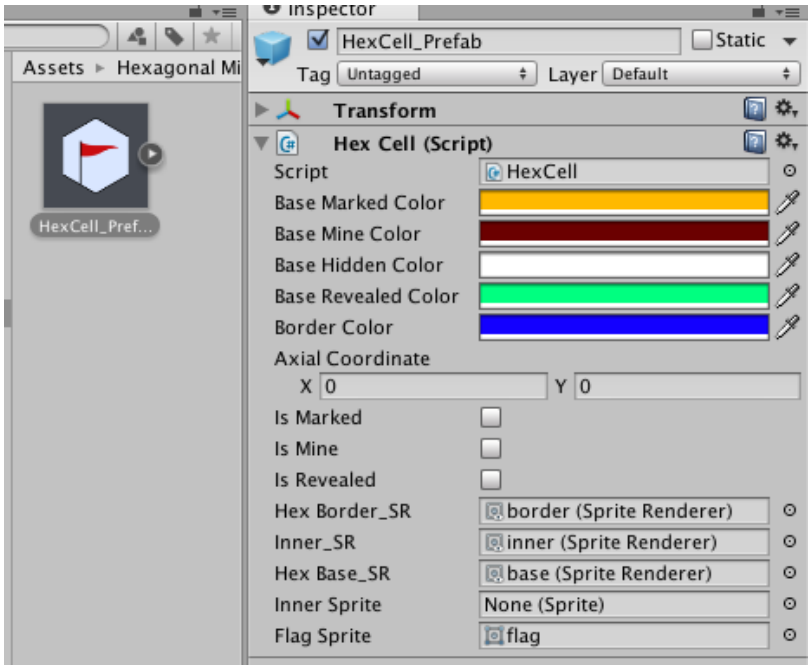
The **utils** folder has the hexagonal helper classes which handles the hexagonal coordinate conversions for the 2 different alignments.



Prefab in Detail

The HexCell_Prefab contains the **HexCell** script attached & 3 sprites as child objects. It contains the hexagon sprite border, hexagon inner fill & a sprite to show the flag/mine/neighbour count.

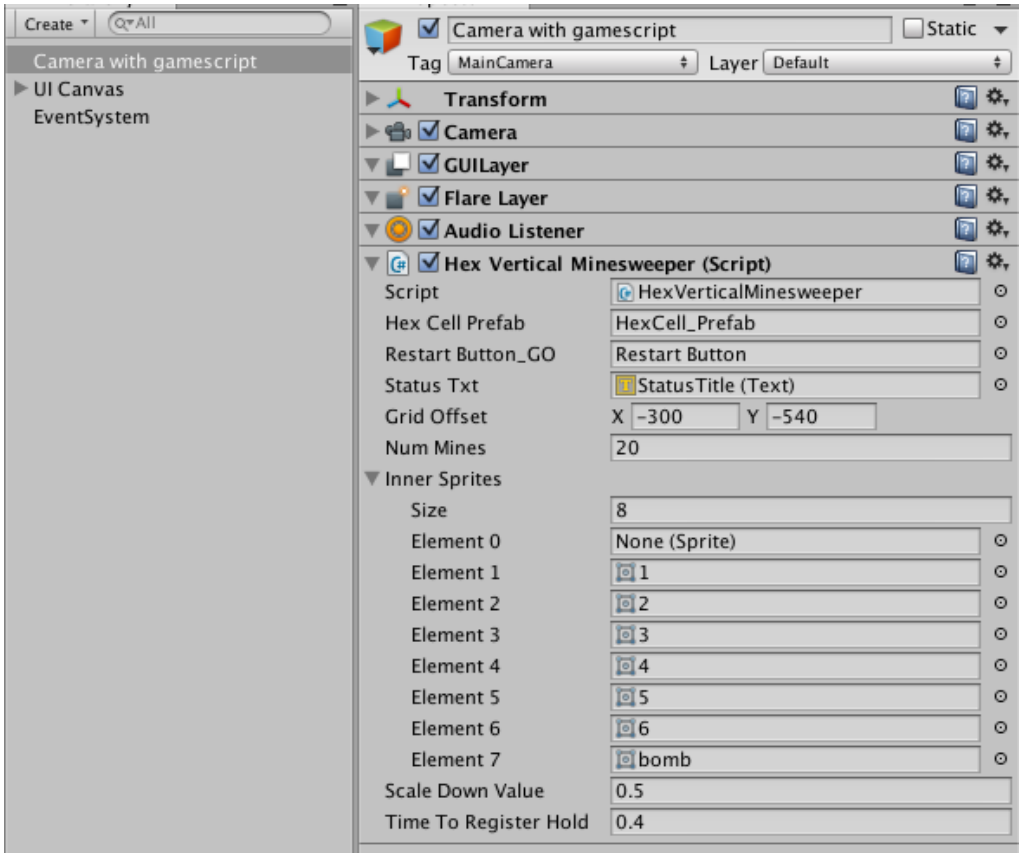




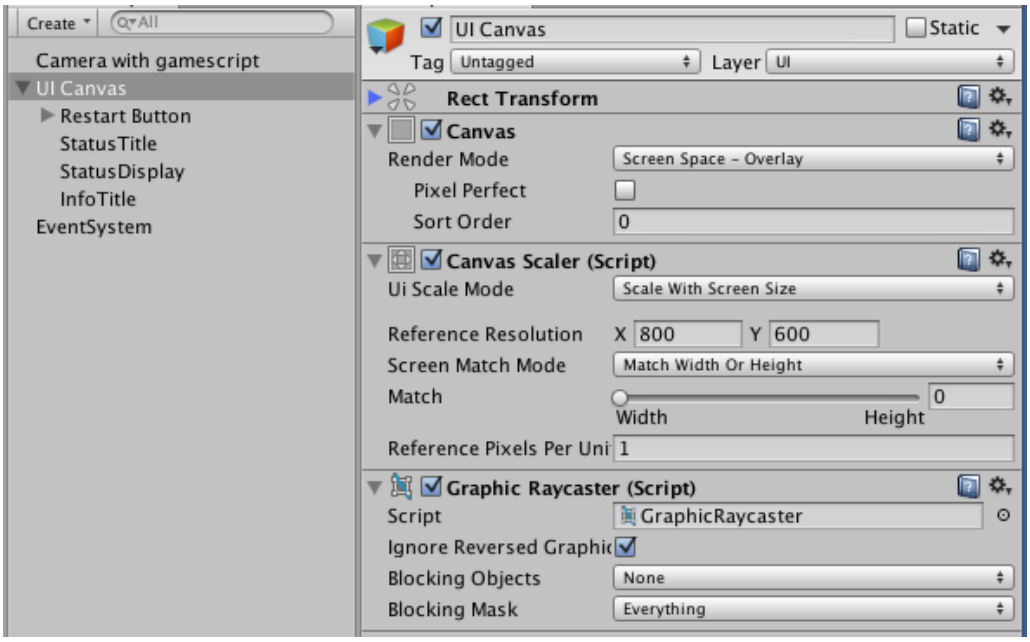
We can change the various colors of the hexagon tile base using the script.

The Game Scene

The **HexMatch** script is attached to the main camera in the game scene.



The UI Canvas element in the scene has some basic ui elements like Texts & Button.



Game Script details

HexMatch script is attached to the main camera with the following public variables which we can set.

HexCelPrefab – The HexCellPrefab
StatusText – The Text ui to show the revealed tile count
RestartButton_GO – The Restart Button ui element.
GridOffset – X and Y values to position the hexagonal grid in scene
NumMines – number of mines in the grid (need to be lower than total blank tiles)
InnerSprites – Array to keep reference to all the sprites to be shown inside a hextile. 0 is blank. 7 is bomb sprite. 1-6 are same numbered sprites
ScaleDownValue – A float value to scale the tiles down to fit inside viewable area.
TimeToRegisterHold – A float value for the seconds to wait till we determine is it is a tap or a tap+hold.

Code Logic

Hexagonal Logic – The **HexHelper** classes help with the conversion of screen coordinates to grid coordinates and vice versa. This script also helps in finding the neighbour tiles for any given tile. For this game we have a **vertically aligned & horizontally aligned odd offset** hexagonal grids. If you are eager to know the math behind the whole conversion you may refer to these hexagonal tutorials, https://gamedevelopment.tutsplus.com/tutorials/introduction-to-axial-coordinates-for-hexagonal-tile-based-games--cms-28820 & http://www.redblobgames.com/grids/hexagons/.

Please feel free to contact in case of any doubts. (juwalbose@gmail.com)

Game Logic

The level data is represented as a 2 dimensional array (levelData) which in turn forms one coordinate system used in the game – the offset coordinates. This is essentially the row & column indices of the level array. For ease of hexagonal logic we use 2 more coordinates, the axial & the cubic coordinates with the help of the HexHelper classes.

While creating the grid we take all blank tiles & randomly place mines in few of them by assigning a value of 10 to the level array. A valid grid in levelArray will have a value of 0 & a value of -1 means outside grid. When we place a mine we increment the values stored in all the neighbouring indices on the levelArray. So if a neighbour had no mines nearby would have 0 as its value & gets incremented to 1 when one mine is placed near it. Then the actual grid is created & each cell will already have assigned as mine or blank or the number of neighbour mines. On tap we will set this data by setting the inner sprite of the tapped cell. So if that was a mine, the game ends. If it is a tile with atleast one neighbouring tile, it gets revealed. If it is a blank tile then all blank tiles connected to it will also get revealed. If we tap and hold, we just set the inner sprite to flag sprite.

The core is that all tiles already know if they are mine or not. If not a mine they already have the sprite corresponding to the number of neighbouring mines. So there is no heavy lifting at runtime other than the connected reveal logic.Once all blank tiles are revealed the game is won.
The classes are commented thoroughly to help you understand.
Please feel free to contact in case of any doubts. (juwalbose@gmail.com)

Thank you for the purchase :)