28th August 2012

Creating a hexagonal grid for games [C++ / JAVA]

This tutorial is intended for people with experience in programming.

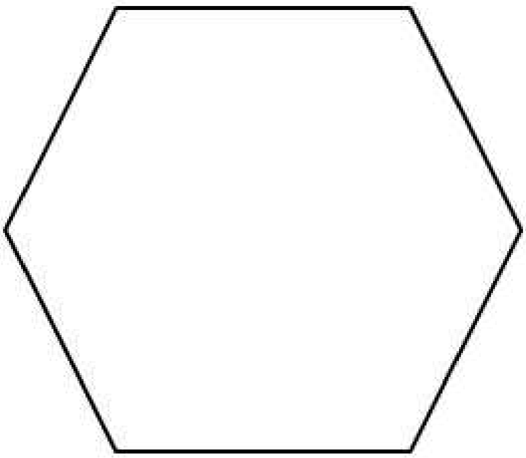
So, you want to make a hexagonal grid for your new strategy game. Well you've come to the right place.
First of all a little geometry.

A regular hexagon can be divided in 6 equilateral triangles (all angles of 60 degrees, all sides are equal). This means that the hexagon's edges will all have the same length.

[]

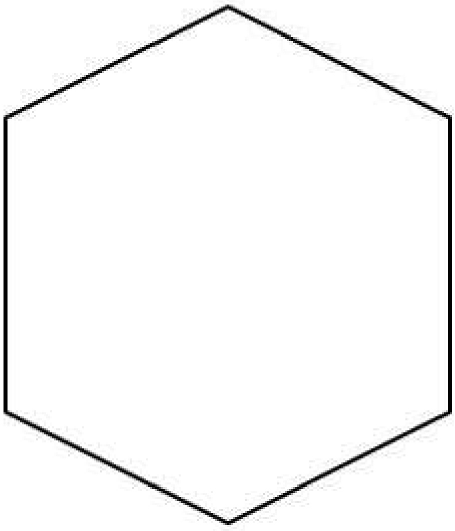
When creating flat hexagonal grids you also have to take into account the orientation of your hexagons. There are usually 2 orientations to choose from:

A flat orientation



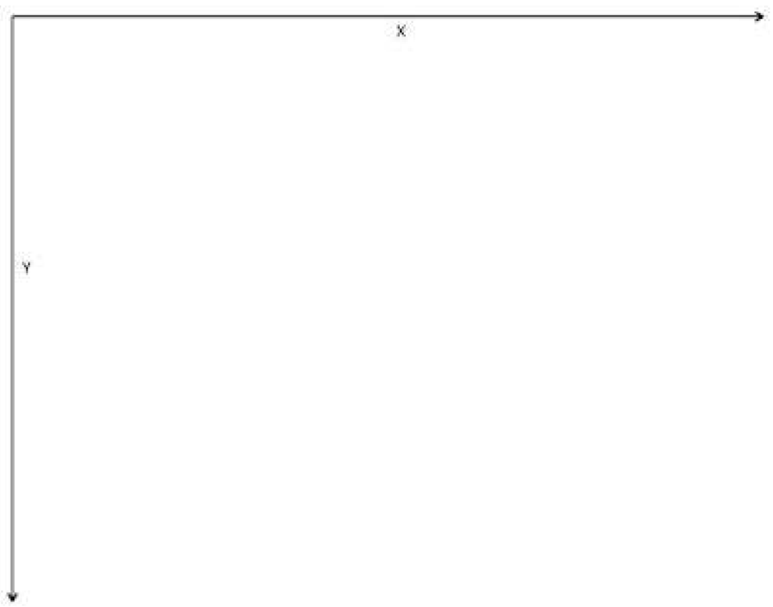
[http://4.bp.blogspot.com/-2qqf2CHCarA/UDyw16RPzAI/AAAAAAAAChQ/fSbeJgT1uRE/s1600/flat.JPG]

Or a pointy orientation:



[http://1.bp.blogspot.com/-INWERLCeox8/UDyw9ZL5z8I/AAAAAAAAChY/XfCD2HbMaGM/s1600/pointy.JPG]

Let us assume we have a coordinate system like this one (the basis is located in the top-left corner. Ox grows to the right, Oy grows to the bottom):



[http://1.bp.blogspot.com/--z9_xe4OpVw/UDyxc_zEgmI/AAAAAAAAChg/mMK4PuOVqio/s1600/coord.JPG]

Let "**s**" be the length of a hexagon side,
and let "**a**" be the distance between:
* the center point of the hexagon and the left or right edge (they are the same), for *pointy* orientation
* the center point of the hexagon and the top or bottom edge (they are the same), for *flat* orientation

a = sqrt((s^2) - ((s/2)^2))
or
a = sqrt(3)*(s/2)

(both formulas are the same, the first one is a bit more precise)

Calculating the coordinates of the 6 vertices:

Let (X, Y) be the coordinates of the center point of a hexagon
The following formulas give the coordinates of all six vertices of the hexagon:
(See the pictures at the bottom of the page. The vertex numbers go clockwise. For a pointy orientation vertex 1 (point 1) is the one on the top. For a flat orientation it is the one on the top-right)

*for *POINTY* ORIENTATION:

point 1: (X, Y - s)
point 2: (X + a, Y - (s/2))
point 3: (X + a, Y + (s/2))
point 4: (X, Y + s)
point 5: (X - a, Y + (s/2))
point 6: (X - a, Y - (s/2))

*for *FLAT* ORIENTATION:

point 1: (X + (s/2), Y - a)
point 2: (X + s, Y)
point 3: (X + (s/2), Y + a)
point 4: (X - (s/2), Y + a)

point 5: (X - s, Y)
point 6: (X - (s/2), Y - a)

Calculating the coordinates of the center points in a hexagonal grid:

Let (**x_0**, **y_0**) be the coordinates of the first hexagon in our grid (the top-left one, with index 0,0)

Let our hexagon grid be a 2 dimentional array of hexagons, called "*grid*".
It will have a maximum of "*gridMaxI*" rows, and "*gridMaxJ*" columns

The coordinates of the center point of the hexagon at position **grid[i][j]** can be calculated with the following formulas:

**flat* orientation:
$x_{i,j} = x_0 + j*((3*s)/2)$
$y_{i,j} = y_0 + (j\%2)*a + 2*i*a$

**pointy* orientation:
$x_{i,j} = x_0 + (i\%2)*a + 2*j*a$
$y_{i,j} = y_0 + i * ((3*s)/2)$

Example pseudocode (C++/JAVA):

Assume we have a class called Hexagon.

It has the following fields:

Point *centerPoint*
Point *vertices* [6]
int *i*, int *j*

...and the following methods:

Hexagon (int X, int Y, int I, int J) - constructor. Creates a new Hexagon with center point: (X, Y), and position in the grid: [I][J]
int getX() - returns the x coordinate of the center point
int getY() - returns the y coordinate of the center point
void setCenter(int x, int y) - sets the center point coordinates for our hexagon
void calculateVertices() - calculates the coordinates of the 6 vertices, based on the type of orientation (flat or pointy)
void drawHexagon() - draws the hexagon on the screen

s , **a** , **gridMaxI** and **gridMaxJ** are global variables

The following pseudocode snippet will draw a hexagonal grid with *gridMaxI* rows and *gridMaxJ* columns.
Every hexagon in the grid will have edges with length *s*:

***Flat orientation:**

```
grid[0][0] = new Hexagon(10,15,0,0);

for (int i=0; i< gridMaxI; i++)
```

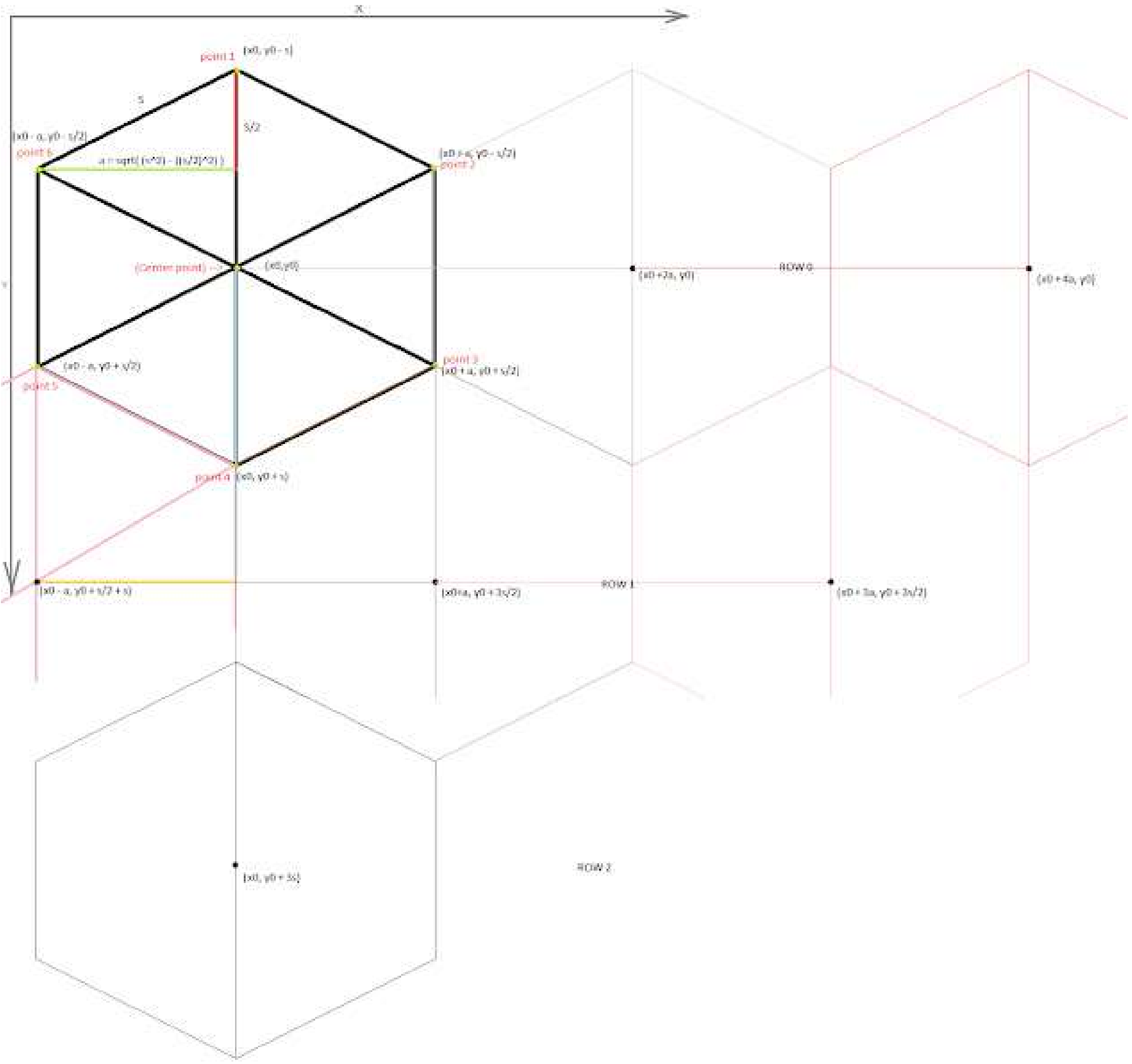
```
 for (int j=0; j< gridMaxJ; j++)
 {
  grid[i][j] = new Hexagon((grid[0][0].getX() + (j*((3*s)/2))), (grid[0][0].getY() + ((j%2)*a) + (2*i*a)), i, j);
  grid[i][j].calculateVertices();
  grid[i][j].drawHexagon();
 }
```

***Pointy orientation:**

```
grid[0][0] = new Hexagon(10,15,0,0);

for (int i=0; i<hexgridMaxI; i++)
 for (int j=0; j< hexgridMaxJ; j++)
 {
  grid[i][j].setCenter( (grid[0][0].getX() + ( i % 2 ) * a + ( j * 2 * a ) ), (grid[0][0].getY() + (i * ( (3*s) / 2 ) ) ) );
  grid[i][j].calculateVertices();
  grid[i][j].drawHexagon();
 }
```

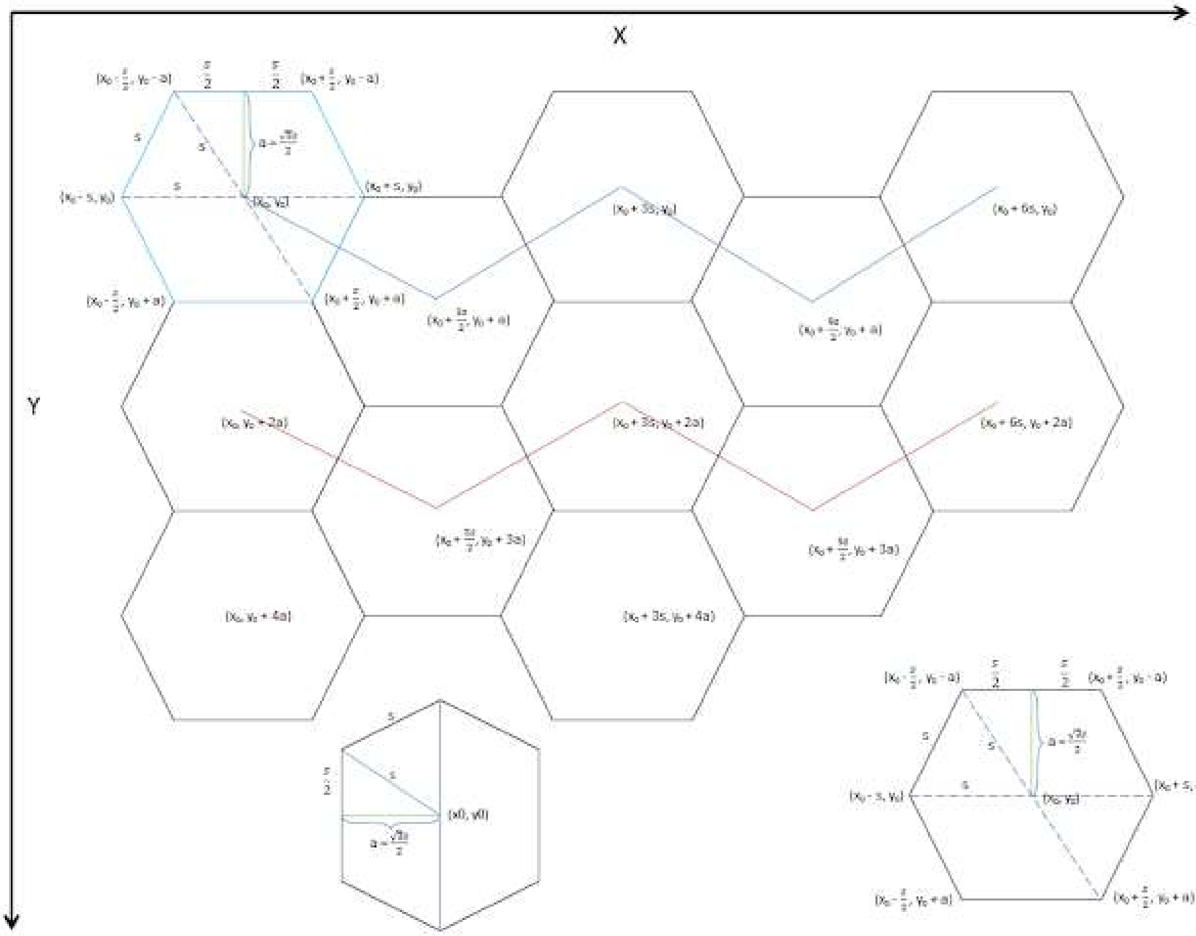
Example picture of a *pointy* oriented grid:



[http://3.bp.blogspot.com/-_hkbKBBlS9s/UDyzhkJcgUI/AAAAAAAACho/VZkDbaA6EkM/s1600/hex+grid+coords.bmp]

Every odd-numbered row in the grid (rows 1, 3, 5, 7...) has the center points of the hexagons offset to the right by **a**

Example picture of a *flat* oriented grid:


X
Y
$(x_0 - \frac{s}{2}, y_0 - a)$
$\frac{s}{2}$
$\frac{s}{2}$
$(x_0 + \frac{s}{2}, y_0 - a)$
s
s
$a = \frac{\sqrt{3}s}{2}$
$(x_0 - s, y_0)$
s
(x_0, y_0)
$(x_0 + s, y_0)$
$(x_0 + 3s, y_0)$
$(x_0 + 6s, y_0)$
$(x_0 - \frac{s}{2}, y_0 + a)$
$(x_0 + \frac{s}{2}, y_0 + a)$
$(x_0 + \frac{3s}{2}, y_0 + a)$
$(x_0 + \frac{9s}{2}, y_0 + a)$
$(x_0, y_0 + 2a)$
$(x_0 + 3s, y_0 + 2a)$
$(x_0 + 6s, y_0 + 2a)$
$(x_0 + \frac{3s}{2}, y_0 + 3a)$
$(x_0 + \frac{9s}{2}, y_0 + 3a)$
$(x_0, y_0 + 4a)$
$(x_0 + 3s, y_0 + 4a)$
s
$\frac{s}{2}$
s
(x0, y0)
$a = \frac{\sqrt{3}s}{2}$
$(x_0 - \frac{s}{2}, y_0 - a)$
$\frac{s}{2}$
$\frac{s}{2}$
$(x_0 + \frac{s}{2}, y_0 - a)$
s
s
$a = \frac{\sqrt{3}s}{2}$
$(x_0 - s, y_0)$
s
$(x_0 + s, y_0)$
(x_0, y_0)
$(x_0 - \frac{s}{2}, y_0 + a)$
$(x_0 + \frac{s}{2}, y_0 + a)$

[http://2.bp.blogspot.com/-sZfLjWz-1wY/UDy1ItdkF_I/AAAAAAAACh4/EBwCMiwdoxY/s1600/hexgrid_rotated.png]

The blue line is row 0, the red line is row 1 of the grid. Every odd-numbered column (columns 1, 3, 5, 7...) has the center point of the hexagons offset to the bottom by **a**

EDIT: after browsing the net for a while I've come across an amazing source for more elegant and advanced implementations of hexagonal grids, including drawing lines from hexagons on a grid, intersecting ranges, implementing Field of View areas on a hexagonal grid and much more. Here is the link: http://www.redblobgames.com/grids/hexagons/ [http://www.redblobgames.com/grids/hexagons/]

Posted 28th August 2012 by Anonymous

3 View comments

Anonymous March 28, 2015 at 4:22 AM

helpful

Reply

Anonymous December 5, 2016 at 11:17 PM

Hi i was just wondering, if i just wanted one singular hexagon with grids inside it. Rather then a collection of them how would i go about it?

Reply

Anonymous November 30, 2017 at 8:24 AM

ayyyyyyyyy

Reply

To leave a commen click the button below to sign in with Google.

SIGN I

Load more