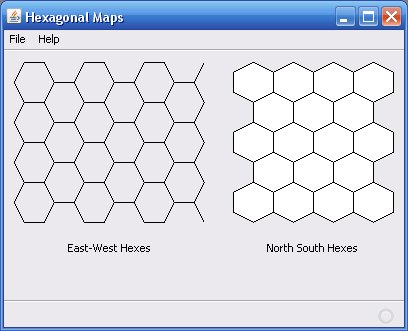
On a whim this week, and in keeping with my desire to learn about Java GUI’s I thought it’d be fun to create a basic GUI with a Hexagonal Map on it and then identify which hexagon a user has clicked on.

As there are two ways in which to draw a hexagonal map I chose to include both, mainly as the code would be almost identical for both.



Some research eventually revealed that there are two main approaches that can be used to convert a mouse pointer position into a hexagonal co-ordinate. The first (and I suspect better approach) involves splitting the map into sections and then calculating the gradient of the lines to establish which hexagon the pointer is in. As you may have guessed I have yet to get this method fully working…

The other method uses a mousemap to identify which hexagon the mouse pointer is over, both of mine are shown below and are based on Tanstaafl’s in his articles on gamedev.net. As an aside this method was apparently used for Civilization II!

http://www.hughson.co.uk/images/Blog/MouseMapHex.pnghttp://www.hughson.co.uk/images/Blog/MouseMapHex1.png

Using this method you simply divide your map into sections, transfer the mouse pointer position to the mousemap and get the colour of the pixel. I’ve included the relevant code for the east-west hexagonal map below, as there seems to be a distinct lack of example source code out there.

[?](http://www.ericbess.com/ericblog/2008/03/03/wp-codebox/#examples)[[Copy to clipboard]](javascript:;)Download [hexMapCode.txt](http://www.hughson.co.uk/wp-content/plugins/wp-codebox/wp-codebox.php?p=69&download=hexMapCode.txt)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85 | **private** **void** ewHexesMouseClicked(java.awt.event.[MouseEvent](http://www.google.com/search?hl=en&q=allinurl%3Amouseevent+java.sun.com&btnI=I%27m%20Feeling%20Lucky) evt)  {  [ImageIcon](http://www.google.com/search?hl=en&q=allinurl%3Aimageicon+java.sun.com&btnI=I%27m%20Feeling%20Lucky) source=**new** [ImageIcon](http://www.google.com/search?hl=en&q=allinurl%3Aimageicon+java.sun.com&btnI=I%27m%20Feeling%20Lucky)("c:**\\**MouseMapHex1.png");    [BufferedImage](http://www.google.com/search?hl=en&q=allinurl%3Abufferedimage+java.sun.com&btnI=I%27m%20Feeling%20Lucky) mouseMap = **new** [BufferedImage](http://www.google.com/search?hl=en&q=allinurl%3Abufferedimage+java.sun.com&btnI=I%27m%20Feeling%20Lucky)(source.getIconWidth(),  source.getIconHeight(),[BufferedImage](http://www.google.com/search?hl=en&q=allinurl%3Abufferedimage+java.sun.com&btnI=I%27m%20Feeling%20Lucky).TYPE\_INT\_RGB);    **try**  {  mouseMap = ImageIO.read(**new** [File](http://www.google.com/search?hl=en&q=allinurl%3Afile+java.sun.com&btnI=I%27m%20Feeling%20Lucky)("c:**\\**MouseMapHex1.png"));  }  **catch**([Exception](http://www.google.com/search?hl=en&q=allinurl%3Aexception+java.sun.com&btnI=I%27m%20Feeling%20Lucky) ex)  {  [System](http://www.google.com/search?hl=en&q=allinurl%3Asystem+java.sun.com&btnI=I%27m%20Feeling%20Lucky).out.println(ex);  }    **int** tileX = 0;  **int** tileY = 0;  **int** mouseMapWidth = 60;  **int** mouseMapHeight = 40;    *// Get mouse position*  **int** mouseX = evt.getX();  **int** mouseY = evt.getY();    *// Find out what region of the map the mouse is in*  **int** regionX = (**int**)(mouseX / mouseMapWidth) \* 2;  **int** regionY = (**int**)(mouseY / mouseMapHeight);    *// Find out where in the mousemap the mouse is*  **int** mouseMapX = mouseX % mouseMapWidth;  **int** mouseMapY = mouseY % mouseMapHeight;    *// Get the pixel colour in the mousemap for the mouse position*  **int** pixelColour = mouseMap.getRGB(mouseMapX, mouseMapY);    *// Find regionDX and regionDY*  **int** regionDX = 0;  **int** regionDY = 0;  **switch** ( pixelColour )  {  **case** 0xffff0000 : *// red*  {  regionDX = -1;  regionDY = -1;  statusMessageLabel.setText("Red");  **break**;  }  **case** 0xffffff00 : *// yellow*  {  regionDX = 1;  regionDY = -1;  statusMessageLabel.setText("Yellow");  **break**;  }  **case** 0xffffffff : *// white*  {  regionDX = 0;  regionDY = 0;  statusMessageLabel.setText("White");  **break**;  }  **case** 0xff00ff00 : *// green*  {  regionDX = -1;  regionDY = 0;  statusMessageLabel.setText("Green");  **break**;  }  **case** 0xff0000ff : *// blue*  {  regionDX = 1;  regionDY = 0;  statusMessageLabel.setText("Blue");  **break**;  }  }    *// Calculate the tile selected*  tileX = regionX + regionDX;  tileY = regionY + regionDY;    *// Display the hexagon co-ordinates*  statusMessageLabel.setText(tileX + ", " + tileY);  } |

I’m going to take a look at the more mathematical option, just as soon as I’ve done this weeks OU study…

Resources:

* Isometric ‘n’ Hexagonal Maps [Part 1](http://www.gamedev.net/reference/articles/article747.asp) & [Part 2](http://www.gamedev.net/reference/articles/article748.asp) by Tastaafl (Mousemaps)
* [Coordinates in Hexagon-Based](http://www.gamedev.net/reference/articles/article1800.asp) Tile Games by Thomas Jahn (Theory)
* [Isometric Cube Coordinates](http://www.gamedev.net/reference/articles/article723.asp) by Charles Fu (Mathematical Theory)
* [Hexagon Hit Test](http://blog.devstone.com/aaron/archive/2004/04/18/152.aspx) by Aaron Zupancic (C#)