**Pixel Coordinates to Hexagonal Coordinates**   
**by** [**Amit Patel**](http://www-cs-students.stanford.edu/%7Eamitp)   
*24 May 1996*

*Taken from a posting to rec.games.programmer*

I'll post the routines I use to calculate which hex the mouse is in. First, I should explain the hexagon size and layout.

|  |
| --- |
| \_\_\_ \_\_\_  / \_\_\_/  \_\_\_/ \_\_\_/  / \_\_\_/  \_\_\_/ \_\_\_/ |

Each hexagon is 28 x 24 pixels, but since the columns overlap, the distance from the center of one hex to the center of the next column's hex is 21.

My coordinate system is offset-grid with no gaps. The lower left is (1,1); as you go up, the N coordinate increases. (I call them (M,N) instead of (X,Y) to distinguish between the hex and square coordinates.) Every other column is pushed up half a hexagon height.

First, this is the approach based on a rec.games.programmer article saved on my web pages. It is based on the view that hexagons are a projection of three dimensional cubes onto a plane. (See that web page for an explanation.)

// Note: HexCoord is a struct that just stores hex coordinates

HexCoord PointToHex( int xp, int yp )

{

// NOTE: HexCoord(0,0)'s x() and y() just define the origin

// for the coordinate system; replace with your own

// constants. (HexCoord(0,0) is the origin in the hex

// coordinate system, but it may be offset in the x/y

// system; that's why I subtract.)

double x = 1.0 \* ( xp - HexCoord(0,0).x() ) / HexXSpacing;

double y = 1.0 \* ( yp - HexCoord(0,0).y() ) / HexYSpacing;

double z = -0.5 \* x - y;

y = -0.5 \* x + y;

int ix = floor(x+0.5);

int iy = floor(y+0.5);

int iz = floor(z+0.5);

int s = ix+iy+iz;

if( s )

{

double abs\_dx = fabs(ix-x);

double abs\_dy = fabs(iy-y);

double abs\_dz = fabs(iz-z);

if( abs\_dx >= abs\_dy && abs\_dx >= abs\_dz )

ix -= s;

else if( abs\_dy >= abs\_dx && abs\_dy >= abs\_dz )

iy -= s;

else

iz -= s;

}

return HexCoord( ix, ( iy - iz + (1-ix%2) ) / 2 );

}

Now, here's another approach that I'm now using. It's not as general, but it's faster.

HexCoord PointToHex( int xp, int yp )

{

// NOTE: First we subtract the origin of the coordinate

// system; replace with your own values

xp -= X\_ORIGIN;

yp -= Y\_ORIGIN;

int row = 1 + yp / 12;

int col = 1 + xp / 21;

int diagonal[2][12] = {

{7,6,6,5,4,4,3,3,2,1,1,0},

{0,1,1,2,3,3,4,4,5,6,6,7}

};

if( diagonal[(row+col)%2][yp%12] >= xp%21 )

col--;

return HexCoord( col, (row-(col%2))/2 );

}

In this approach, I first figure out which "half row" the (x,y) lies in, and put that in `row'. Each hexagon occupies two half rows, but every other column chooses different half rows to start with.

Then I figure out which column I'm in, approximately, and put that in 'col'. (Each approximate column is 21 pixels wide.)

|  |
| --- |
| | \_\_\_\_|  | /  |/ |  | |/|  | \_\_\_\_/ 28 = hex width  | |  0 21 |

The vertical lines marks the approximate column boundary. The half row and the column number tells me whether I need to look at the / diagonal or the diagonal.

I then look at the pixel locations of the diagonal. I can use the y coordinate (modulo the half row height) as an index into the diagonal. If the x coordinate (modulo the column width) is LESS than the diagonal value, then I need to move the coordinate to the LEFT.

- Amit