**By Jim Adams**

**Screen to Map Coordinates**  
**3rd Update (Jun 5, 1998)**  
**by Jim Adams**

*Copyright © 1998 by Jim Adams. All rights reserved.*

Do what you want with this information, although all text is copyright © 1998 to Jim Adams, and may not be duplicated, replicated, rejuvinated or whatever else you can think of to make a copy with the following exceptions:

1. You may post it in its entirety for public viewing in an unaltered state.
2. You MAY NOT charge money for using or viewing it.

I'm not responsible for anything you do with this information or any damage it may cause you, or anybody else, using it.

No animals were harmed during the making of this article.  
For further information about this topic, visit your local library.  
Badges!?! Badges?!?! We don't need no stinking badges!!!!

if(overdoing\_it\_a\_bit)  
exit(1);

**The Old Method**

(Refering to our Sci-Fi Isometric game CRPG, 'The Light Befallen')

Our original method to convert from screen space to map space was to 'cut' the screen into sections the size of our tiles, 64x32. Once we placed the coordinates inside one of the sections, we could then look into a 64x32 array and see where exactly it was, whether it be pointing to the map section up-left, up-right, down-left, down-right, or on the map section itself.

NOTE:  
In actuality, the tiles are 64x31, but are padded down to 32 with the bottom-most line blank so it all fits together correctly. The padding works for calculations ONLY. Don't waste the space to store the extra line in memory for graphics purposes. This is all because when we tile the screen, the two leftmost and rightmost pixels of the tile will touch, with room for the topmost and bottommost lines to fit in.

The array would represent something like the tile in memory. If the pointer fell inside the \* area, it's inside the map section. If it's inside the area up and left or \*, then its one map section to the left. You should get the idea about the rest of the sections.

..............................\*\*\*\*..............................

............................\*\*\*\*\*\*\*\*............................

..........................\*\*\*\*\*\*\*\*\*\*\*\*..........................

........................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*........................

......................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*......................

....................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*....................

..................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..................

................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*................

..............\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..............

............\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*............

..........\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..........

........\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*........

......\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*......

....\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*....

..\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

..\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..

....\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*....

......\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*......

........\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*........

..........\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..........

............\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*............

..............\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..............

................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*................

..................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..................

....................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*....................

......................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*......................

........................\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*........................

..........................\*\*\*\*\*\*\*\*\*\*\*\*..........................

............................\*\*\*\*\*\*\*\*............................

..............................\*\*\*\*..............................

................................................................

Here's how the function looked:

void pointer\_get\_map\_position(short pointx, short pointy,signed short mapxo,

signed short mapyo, short \*mapx,short \*mapy)

{

short mx, my;

short rmx, rmy;

short xo, yo;

signed char xoff[5] = { -1, 0, 0, 0, 1};

signed char yoff[5] = { 0, -1, 0, 1, 0};

char off\_map[32][64] = {

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1 },

{ 0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1 },

{ 0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1 },

{ 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2 },

{ 3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4 },

{ 3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4 },

{ 3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 }

{ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4 },

};

// divide by the base tile size, adjusting for the clipping

mx = (short)((pointx + 32 + mapxo) / 64);

my = (short)((pointy + 16 + mapyo) / 32);

// calculate a rough estimate of the map coordinate from those

rmx = mx + my;

rmy = my - mx;

// figure the position of the pointer within that rough map space

xo = (pointx + 32 + mapxo) & 63;

yo = (pointy + 16 + mapyo) & 31;

// further adjust the map positions based on the pointers exact

// position within those map coordinates

rmx += xoff[off\_map[yo][xo]];

rmy += yoff[off\_map[yo][xo]];

// store the results

\*mapx = (short)rmx;

\*mapy = (short)rmy;

}

Now that is the quick and dirty method. Although not to great, as it uses an array, it does do the job. You can look at the array and see it uses the numbers 0-4.

The function quickly finds where the pointer is located inside that array (like a mask), and gets the number. The function then uses that number in another array that says how much to offset our map coordinates to get the correct ones.

I won't get into more detail on this - it's a quick and dirty method, and it's so simply to understand.

**Out With the Old, In With the New**

After a few nights of playing, I came up with the following formulas to replace the above function. (Note that these are based off a tile size of 2:1, which means tiles are twice as wide as they are high, or in our case 64x32. You can swap all the 2's in the calculations with others to compensate if your tiles are different sizes.)

signed short xo = pointer\_xpos - (center\_x + xadjust);

signed short yo = pointer\_ypos - (center\_y + yadjust);

signed short x = yo + (xo/2);

signed short y = yo - (xo/2);

This will return a signed value in x and y that represents the difference from the center map coordinate to where you are pointing. The center map coordinate should be where you draw the 0,0 map coordinate, which for us is the top-left of the screen (0,0).

(The tiles have their own fine coordinate system, with coordinates 0,0 being the top-middle of the tile.)

Something like:

Top-Left of Screen

+------------------------------------------------

| /|\ <-- This (TOP) is coord 0,0 of the tile

y| / | \

a| / | \

d| / | \ <-- FIRST TILE TO BE DRAWN HERE

j| \ | /

u| \ | /

s| \ | /

t| \|/

| |

|

--------------

xadjust

xadjust and yadjust are fine-scrolling or adjument variables if your engine is capable.

The above might be hard to understand. If you refer to my older isometric engine article, and the smooth scrolling tile worlds article, these make more sense.

Let's just say that if you draw your tiles 16 pixels left of what we normally would, then xadjust would be -16. Draw it 20 pixels lower then we would, yadjust would be +20.

Confusing? Not really. Let's say we start drawing our screen at SCREEN coordinates 0,0. The top-middle of the tile is drawn there, so the left side of the tile is cut from the screen (as in the above pic). All calculations are based off this - if your tiles are different, use xadjust and yadjust to compensate for it.

We place our pointer at SCREEN coordinates 125, 304. We are not fine scrolling, so those can be 0.

So:

1)

xo = 125 - (0 + 0);

yo = 304 - (0 + 0);

x = yo + (xo / 2);

y = yo - (xo / 2);

2)

xo = 125;

yo = 304;

x = 304 + (125 / 2);

y = 304 - (125 / 2);

3)

x = 304 + (62);

y = 304 - (62);

4)

x = 366;

y = 242;

Those numbers now represent the map coordinates (fine). To come up with what map section is currently pointed at, just divide these values by you map section size (32x32 in our case) and voila! We have map coordinate 11, 7! (Gosh, I hope that's right :) )

What's that you ask? What is the MAP section size? Each tile in the map can be further mapped out to have its own coordinate system inside of it. For all calculations, it can be any size, but for simplicity sake, use the height of your tile (32 for us).

Now I have to warn you - if you take the fine map coordinates and try to divide it by the map section size, you WILL have a miscalculation when dealing with values in the negative range (because of roundoff errors).

To fix that, if the map coordinates are negative, subtract 31 (or your map section size - 1) from the result and divide that by your map section size.

Those with a keen eye will notice I'm off by 1 there - it should be subtract 32 (the map section size), but there's method behind my madness. It's hard to explain why - all I can say is that two sections together, the left one in the negative range, right one in positive - the edges can't share a coordinate, so you have to 'smudge' it by one.

So one last final time, I'll present the code to get the map fine coordinates and map section coordinates:

void pointer\_get\_fine\_map\_coords(short pointer\_xpos, short pointer\_ypos,

signed short xadjust, signed short yadjust,

short center\_x, short center\_y,

signed short \*mapx,signed short \*mapy)

{

signed short xo, yo;

signed short x, y;

// get the x and y positions on screen relative to scroll and center coords

xo = pointer\_xpos - (center\_x + adjhorz);

yo = pointer\_ypos - (center\_y + adjvert);

// start to figure the map coordinates

x = yo + (xo/2);

y = yo - (xo/2);

// store the results

\*mapx = (signed short)x;

\*mapy = (signed short)y;

}

void pointer\_get\_map\_coords(short pointer\_xpos, short pointer\_ypos,

signed short xadjust, signed short yadjust,

short center\_x, short center\_y,

signed short \*mapx,signed short \*mapy)

{

signed short x, y;

pointer\_get\_map\_coords(pointer\_xpos, pointer\_ypos,

xadjust, yadjust,

center\_x, center\_y,

&x, &y);

if(x < 0)

x -= 31;

if(y < 0)

y -= 31;

x /= 32;

y /= 32;

\*mapx = (signed short)x;

\*mapy = (signed short)y;

}

**Contact Me!**

Do your part to stomp out bugs! Please report any mistakes I may have made and I'll try to fix them right out. Got any suggestions for future articles or need help with a certain subject? Email me!

Jim Adams  
the Collective Mind  
Co-Designer 'The Light Befallen' - A new Sci-Fi Isometric CRPG  
nospam.tcm.lv@worldnet.att.net (remove first part (damn spam-bots))  
nospam.tcm@pobox.com (again, remove first part)  
<http://www.mackay.net.au/~dace/>