Isometric Source Code   
by Geoff Howland

This is some old source code that I used to draw my isometric tiles, when I was doing it that way in the early stages of Force Recon. I was using the CDX library to handle DirectX then and there are a lot of global or outside variables that aren't documented in the routines very well simply because they were documented outside the code. I haven't cleaned this up since I first put it together and I'm not even sure the version of this code since I just grabbed it out of the middle of my old archive, I hope it helps in an algorithmic fashion though, which is how I'm intending it.

I reformatted this with tabs but I have not done anything else with it. The tabs will be lost in HTML format so download the text format to read them correctly.

**Definitions:**

mx,my = Map X/Y coordinates   
sx,sy = Isometric screen coordinates   
sax,say = Screen Actual coordinates, square tile version   
xOffset,yOffset = Pixel offset from top left corner of current screen actual tile   
mxOffset,myOffset = Offset of map tiles from the top left corner of the screen, for scrolling the map   
ax,ay = Actual screen coordinates in pixels   
sx,sy = Starting X/Y, usually for a comparison   
ex,ey = Ending X/Y, usually for a comparison

void drawTiles(int wallVal)

{

// Variables for drawing the map //

int startX, startY=5; // Starting positions for drawing tiles //

// Y=5 seems to offset it correctly //

int drawX=0, drawY=0; // Working positions for drawing tiles //

int mapX=0-screen.mxOffset, mapY=-3-screen.myOffset; // For starting map positions //

int tmapX=0, tmapY=0; // Temporary working map positions //

int val=0; // Value for tile, for more than one tile and testing //

// Draw Floor Map //

for (int y=0; y < 52; y++) { //MAP\_NUM\_Y; y++) {

if (y % 2 == 0) startX = 0; // Even Line //

else startX = 32; // Odd Line //

drawX = startX; drawY = startY; // Sync pix starts //

tmapX = mapX; tmapY = mapY; // Sync map starts //

for (int x=0; x < 25; x++) { //MAP\_NUM\_X; x++) {

val = checkMap(tmapX, tmapY);

// Draw Floor //

if (val == 1)

tempSprites.AddSprite(FloorMap,0,drawX-TILE\_X-64+screen.xOffset,drawY-TILE\_X+screen.yOffset,0,0,0,1,0);

else if (val == 0)

tempSprites.AddSprite(FloorMap,0,drawX-TILE\_X-64+screen.xOffset,drawY-TILE\_X+screen.yOffset,0,0,0,0,0);

// Draw walls where applicable //

if (val == 3)

tempSprites.AddSprite(Wall,0,drawX-WALL\_X-64+screen.xOffset,drawY-WALL\_Y+screen.yOffset,0,0,0,1,0);

else if (val == 2)

tempSprites.AddSprite(Wall,0,drawX-WALL\_X-64+screen.xOffset,drawY-WALL\_Y+screen.yOffset,0,0,0,0);

else if (val == 4)

tempSprites.AddSprite(Wall,0,drawX-WALL\_X-64+screen.xOffset,drawY-WALL\_Y+screen.yOffset,0,0,0,2,0);

else if (val == 5)

tempSprites.AddSprite(Wall,0,drawX-WALL\_X-64+screen.xOffset,drawY-WALL\_Y+screen.yOffset,0,0,0,3,0);

// Increment X for the next tile //

drawX += TILE\_X;

// Increment Map values //

tmapX+=1;

tmapY+=1;

}

// Increment Y for next line //

startY += 15; // Y for new ground (64x30)

// Adjust starting position of map for next line //

if (y%2 == 0) { // Even //

mapY+=1;

}

else { // Odd //

mapX-=1;

}

}

///// End walls draw /////

}

void scrollScreen(int x, int y)

{

// Increment Screen pixel offsets //

screen.xOffset += x;

screen.yOffset += y;

// Increment Map tile offset //

// x offset //

if (screen.xOffset > 63) {

screen.mxOffset += 1;

screen.myOffset += 1;

screen.xOffset = screen.xOffset - 64;

}

else if (screen.xOffset < 0) {

screen.mxOffset += -1;

screen.myOffset += -1;

screen.xOffset = screen.xOffset + 64;

}

// y offset //

if (screen.yOffset > 29) {

screen.mxOffset += -1;

screen.myOffset += 1;

screen.yOffset = screen.yOffset - 30;

}

else if (screen.yOffset < 0) {

screen.mxOffset += 1;

screen.myOffset += -1;

screen.yOffset = screen.yOffset + 30;

}

// End Incrementing Map tile offset //

}

// Checks tile value on coordinate of map //

int checkMap(int x, int y)

{

int val=0;

if (x < 0 || y < 0)

return (-1);

val = Map1[x+(MAP\_NUM\_X\*y)];

return (val);

}

**Mouse Routines**

void mouseConvert()

{

// Modify tmx,tmy with the tileX,tileY //

mouse.tmx += TILE\_X; mouse.tmy += TILE\_Y;

// Get Tile X,Y //

mouse.tx = mouse.tmx / 64;

mouse.ty = mouse.tmy / 30;

// Get Mouse X,Y //

mouse.mx = mouse.tmx % 64;

mouse.my = mouse.tmy % 30;

// Determine what tile from center tile //

if (mouse.my < 14) { // Mouse is on top part //

mouse.sx = 29 + (-2 \* mouse.my); // Interpolate edges //

mouse.ex = 33 + (2 \* mouse.my);

if (mouse.mx < mouse.sx) {

mouse.ax = -1; mouse.ay = -1;

}

else if (mouse.mx > mouse.ex) {

mouse.ax = 1; mouse.ay = -1;

}

else {

mouse.ax = 0; mouse.ay = 0;

}

}

else if (mouse.my > 14) { // Mouse is on bottom part //

mouse.sx = 29 + (-2 \* (29 - mouse.my)); // Interpolate edges //

mouse.ex = 33 + (2 \* (29 - mouse.my));

if (mouse.mx < mouse.sx) {

mouse.ax = -1; mouse.ay = 1;

}

else if (mouse.mx > mouse.ex) {

mouse.ax = 1; mouse.ay = 1;

}

else {

mouse.ax = 0; mouse.ay = 0;

}

}

else { // Middle line is automatic //

mouse.ax = 0; mouse.ay = 0;

}

// Decode ax and ay to figure out the screen tiles //

if (mouse.ay < 0) { // Above center tile //

if (mouse.ax < 0) { // Left of center tile //

mouse.sax = mouse.tx; mouse.say = (mouse.ty \* 2) -1;

}

if (mouse.ax > 0) { // Right of center tile //

mouse.sax = mouse.tx+1; mouse.say = (mouse.ty \* 2) -1;

}

}

else if (mouse.ay > 0) { // Below center tile //

if (mouse.ax < 0) { // Left of center tile //

mouse.sax = mouse.tx; mouse.say = (mouse.ty \* 2) +1;

}

if (mouse.ax > 0) { // Right of center tile //

mouse.sax = mouse.tx+1; mouse.say = (mouse.ty \* 2) +1;

}

}

else {

mouse.sax = mouse.tx; mouse.say = mouse.ty \*2;

}

// End ax and ay Decode //

// Decrement Screen Actual X and Y //

mouse.mx = mouse.sax-1; // Init the map coords //

mouse.my = mouse.say-1;

// Convert Screen to Map Coords //

mouse.mx = mouse.mx - (mouse.my / 2); // Dividing by two gives the number of X's to move over for 0x for my lines down //

mouse.my = mouse.my + mouse.mx; // Adding mx to my will also offset the upwards slope that each Y line has //

/\* if (mouse.mx < 0) // If mx is negative then cross the map //

mouse.mx = MAP\_NUM\_X + mouse.mx;\*/

}

[Related Posts Plugin for WordPress, Blogger...](http://www.linkwithin.com/)