Isometric Tiles   
Revision 1.0   
by [Dino M. Gambone](mailto:dmgmbone@yahoo.com)

**What are Isometric Tiles?**

Isometric tiles are diamond shaped pictures that can be combined with other isometric tiles to form a seamless landscape for tile-based games. Due to its diamond shape, the isometric tile gives the illusion of depth.

|  |  |
| --- | --- |
| http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168318_0.gif | http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168319_1.gif |
| A single isometric tile | A map using isometric tiles |

**Isometric Tile Dimensions**

At first glance of a normal isometric tile image, you think that there are only two dimension, a width and a height. This is not totally true. Although the image does actually have only 2 dimensions (width and height) the tile image represents a 3 dimensional space and thus it needs to be described using 3 dimensions: Length, Width, and Height. A first this will be a bit confusing to remember. Don't worry. It will eventually become more natural as you work with isometric tiles.

The picture to the right shows the isometric tile with its dimensions described.  
http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168320_2.gif

* Length: The leftmost point of the tile to the rightmost point of the tile.
* Width: The farthest point of the tile from the screen to the closest point of the tile to the screen. This is probably the most confusing dimension of isometric tiles.
* Height: The 'altitude' or thickness of the tile.

**Drawing isometric tiles**

Since an isometric tile isn't a nice rectangular shape, a simple bitblt will not work. What needs to be done is a transparent bitblt needs to be performed. If the method you are using to bitblt the tile does not support transparent bitblt'ing, then a mask of the tile being bitblt is needed. The steps needed to bitblt the tile are as follows:

1. You need the picture of the tile and a picture of a mask of the tile.  
   http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168322_3.gifIn the picture to the right, the white background of the tile image is the transparent portion of the image.
2. Bitblt the mask using the OR operator. For windows, this is the SRCPAINT raster operation (ROP). This will create a sort of 'cookie cutout' where the part where the image will be is white.
3. Bitblt the actual image using the AND operator. For Windows, this is the SRCAND raster operation (ROP). This will place the image onto the cutout without damaging any surrounding images.

The actual shape of the mask is up to you. The mask shown in the picture above allows the entire image of the tile to come through.

http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168323_4.gifThings get a bit trickier if you just want the top portion of the tile to show. No longer can simply just bitblt a mask and image because the second bitblt will damage the surrounding graphics. This is one drawback to having 3 dimensions. A solution to this drawback is to have the tile sides as a seperate image. This means more bitblt'ing but more flexiblity also. If you need the top of the tile refreshed, but not the sides then you can do that if you have the tile in two images.

**Isometric Tile Layering**

In order to get a smooth transition from one type of tile/terrain to another tile/terrain, 'fringe' layers need to be applied. Fringe layers are simply flat tile images (no sides) that are drawn on top of a tile for a smooth transition effect. Bellow is an example of the application of a fringe to a border of grass and stone. Of course a better artist could make the fringe look even better by applying a little dirt on the side and better colors.

|  |  |
| --- | --- |
| http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168324_5.gif | http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168325_6.gif |
| Tiles used to draw the image. Third tile is a fringe tile. | Small example of a fringe. |

This example is not the best example but it will do. The way the the fringe layer is applied is the same way the stone tile is drawn. First the stone tile is drawn (mask and then image) and then the fringe is drawn at the same point (mask and then image). This gives the illusion that the grass is invading the cracks of the stone. (Well it's suppose to if the art work was better :) )

Layer upon layer can be applied to a tile. There is a drawback, of course. The more layers you add to the tile, the slower the drawing of the map will be since it needs to bitblt over and over again. Layering is nice and I recommend it... but don't go overboard.

**Variable Tile Heights and Tile Base**

An isometric tile occupies a three dimensional space. There is no rule that all tiles in a set of tiles (or tileset) need to be the same height. Having variable height tiles is actually a nice feature to have. Let's take our stone tile for instance. The first image of the tile has a height of 9 pixels. There is no rule that says that our grass tile needs to be the same height as the stone tile. Actually, making the tiles different heights by a pixel or two will make the landscape look even more realistic.

In order to be able to draw tiles with variable heights properly, we need to know where is the base for this tile. What's the tile base? The tile base is simply the offset from the tile top (the maximum height value) to the point in the tile where the tile is flush with the ground. The image bellow shows an example of 3 tiles with different heights and how they are drawn next to each other using the tile base as a sort of guide line.

|  |  |
| --- | --- |
| http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168326_7.gif | http://web2.cyol.com/expo/xml/star/1000/56/img/wave/2010/04/img_1270168327_8.gif |
| 3 tiles with different heights. The purple line is the tile base for that tile. | The 3 tiles drawn next to each other with the tile bases lined up. |

When drawing tiles with a tile base other than 0, simply move the tile up on the Y coordinate prior to the bitblt and then bitblt at that point. Here's a sample C/C++ code to do that:

void DrawTile(char TileID, long DestX, long DestY, char BaseOffset){

//Initializing stuff

DestY -= BaseOffset; //Move the tile up the Y axis to take into

//account the tile base.

//Actual bitblt call

}

It's actually very easy to implement. One very nice thing about variable height tiles is that it doesn't take away that much processing power to do unlike the bitblt'ing done above.

**Conclusion**

That is the quick and slightly confusing story of isometric tiles from my point-of-view. All of these ideas have been developed with a particular map engine in mind but the ideas should be abstract enough to be useful in any map engine. If you have any questions about this article, feel free to contact me (See above). I am willing to help anyone who asks nicely. If I don't respond right away, don't get mad... I'm probably just busy or away. I'll always respond to tell you I got your email.

Good Luck.