**Tilemaps**

*The tileset and tilemap are taken from the C toolchain, only to provide an example.*

*Needed tools:*

* [TokenIDE](https://www.cemetech.net/programs/index.php?mode=file&id=515)
* [Tiled](http://www.mapeditor.org/)
* [ConvTile](https://github.com/mateoconlechuga/convtile)
* [ConvPNG](https://github.com/mateoconlechuga/convpng)

1. **Creating tilesets**

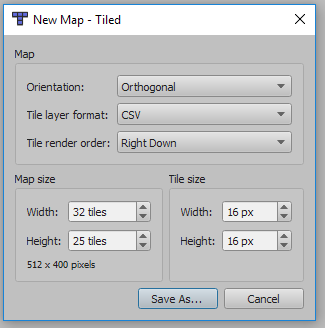
A tilemap cosist of several sprites. These sprites are combined in one big tileset, which is just a big picture with all the separate sprites in it. All the sprites should have equal size, and that size can be anything, up to 255 if you want. To create sprites, you can use Paint or [TokenIDE](https://www.cemetech.net/programs/index.php?mode=file&id=515). In both cases, be sure to save the tileset as a PNG file, otherwise you can’t use it for creating tilemaps. Paint is straightforward; TokenIDE is a bit harder, but it might be easier. When using TokenIDE, go to Tools > Hex Sprite Editor > Color Sprites, specify the tileset width and height (can be anything, as long as it fits the number of sprites you want to use), and start drawing. Be sure to select “xLIBC” instead of “BasicColors”! When you are ready, save it via File > Save As and save it as a PNG file. In this tutorial, we will use this tileset:



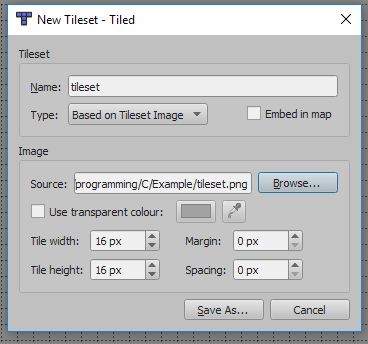
1. **Creating tilemaps**

A tilemap is basically a 2D map full of sprites (we will now use tiles instead of sprites). These can be orthogonal tiles, but isometric tiles can be used as well, for example in Age Of Empires II. In ICE you can only display orthogonal tiles, so we will stick to that part, which is luckily the easiest as well. You have 2 options to create a tilemap: do it manually, writing down all the indexes of the tiles in the tileset, or using an editor, like [Tiled](http://www.mapeditor.org/). Maybe there are other tools, but I will only explain Tiled, so in case you don’t understand how to create tilemaps, be sure to download Tiled!

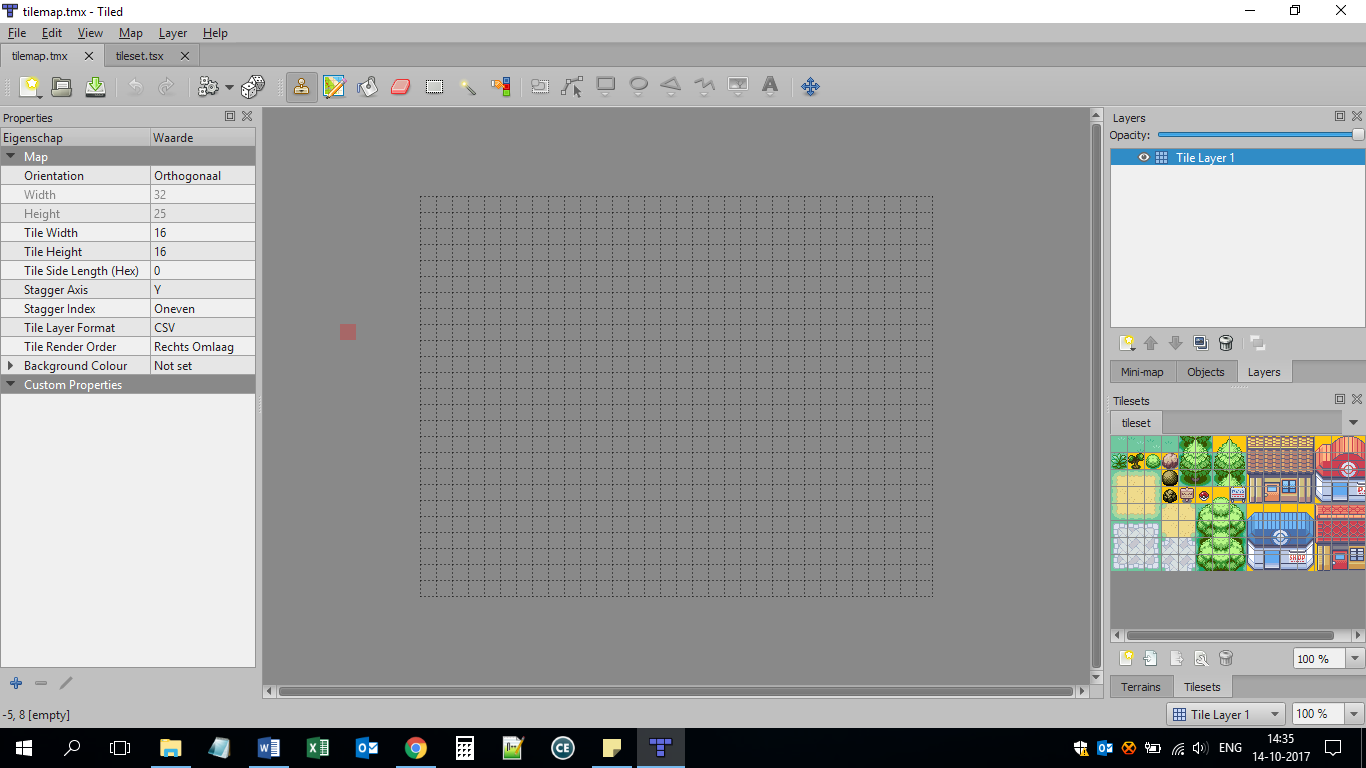
When opening Tiled, you get a nice, empty screen, with 2 buttons: “New Map” and “New Tileset”. Click on “New Map”. The first 3 options are right, but the map- and tile-format are more interesting. Be sure to select the right width and height of the map, and the right width and height of the tiles. In our example, the tilemap is 32x25, and the tiles are 16x16, so that looks like this:



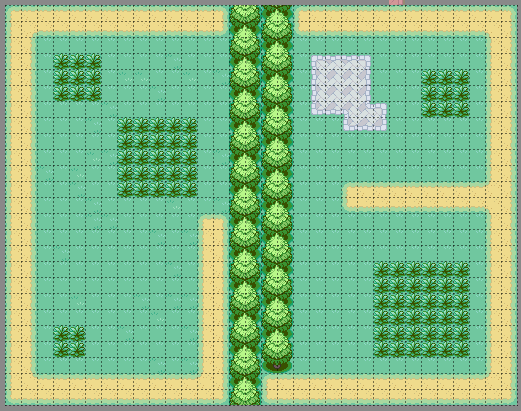
Save this tilemap in the same directory as your PNG file, the tileset. Now you see an empty tilemap, and you can’t load any tiles yet, so we first need to load the tileset. Click on File > New > New Tileset…. Browse your computer to load the tileset PNG, change the tile width and height eventually, and save this tileset in the same directory as your PNG. That looks like this:



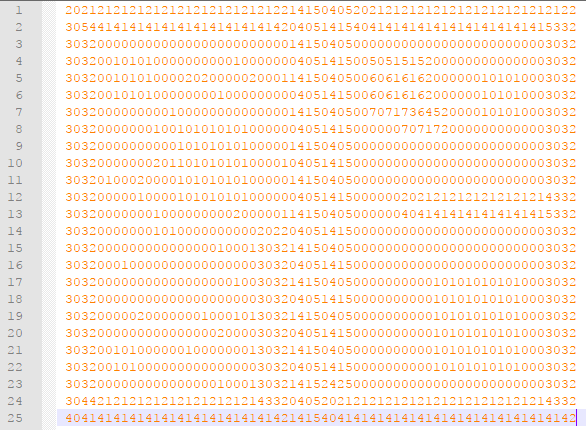
At the top bar, click on tilemap.tmx, or whatever you named it, and now you should see the tileset loaded at the bottom right!



Now you can start editing, and creating the tilemap you want. I will use this tilemap:



Cool! Now you need to export this tilemap as a CSV file. Click on File > Export As and save it as a CSV file in the same directory as your tilemap, tileset and PNG file. However; ICE still can’t use a CSV format, so we need a converter, and that is why we need [ConvTile](https://github.com/mateoconlechuga/convtile). Be sure to put convtile.exe in the same directory! Now, from the command prompt, run ‘convtile output’. If everything works right, there should be 2 files created: output.c and output.h. You only need output.c. Open this file, remove all the “0x” and the “,”, and only all the hexadecimals are left, like this:



Put the hexadecimals all on one line, and you have the code which could be used in “DefineTilemap(“.

1. **Convert to ICE format**

Now you need to use [ConvPNG](https://github.com/mateoconlechuga/convpng) to convert the tileset into an appvar. Create a new file ‘convpng.ini’ with this code in it:

#AppvarICE : TILES

#PNGImages :

#GroupICE : tileset\_gfx

#OutputPaletteArray : false

#Tilemap : 16,16,true

#Palette : xlibc

#PNGImages :