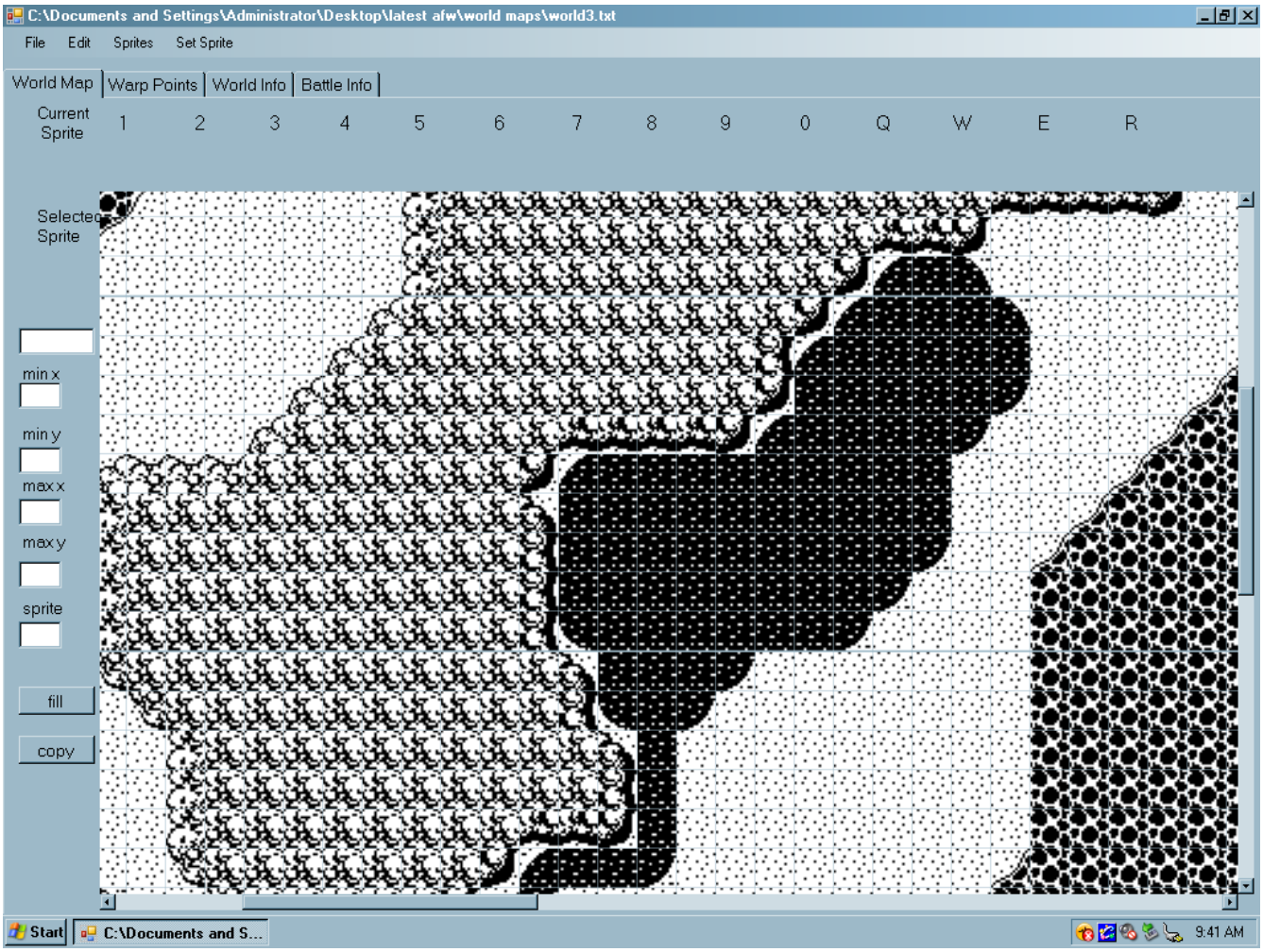
Map Creation HowTo

You will need .NET 3.5 sp1 installed in order to run the IDE on your computer.

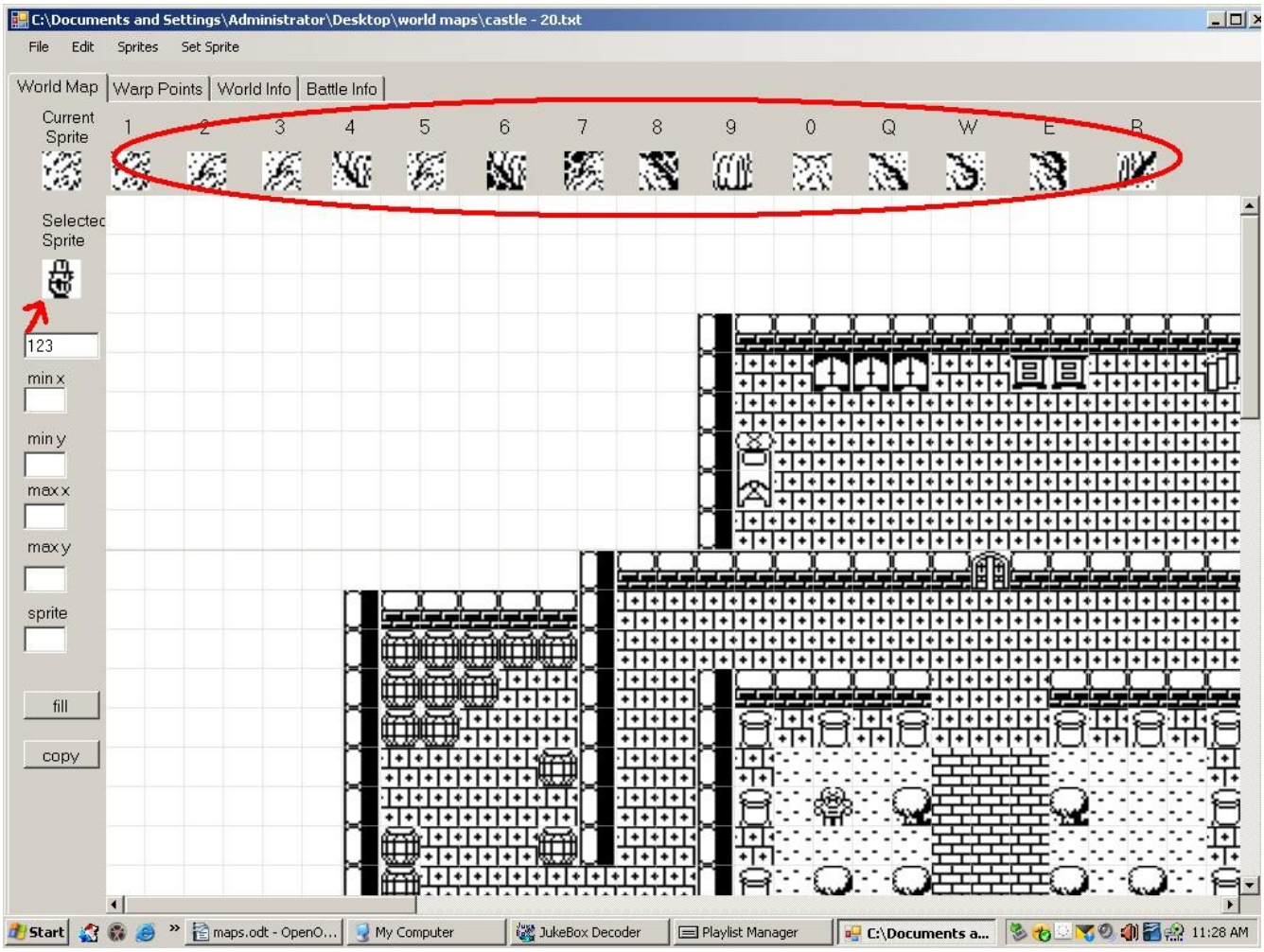
All maps are 60x109 arrays. Rows 0→56 are used for graphics, while the last 3 rows hold information about the map, such as warp points, battle info, and linked storylines. You will have viewing area of about 50x100 when it comes down to the maps. The other 7x9 are for buffer areas that prevents white space from showing up when coming to the edge of the map.

The array is an int. The map structure is fairly simple. The int stores the ID #'s for the sprites that are located in the gfx file. AFW looks at the map array (called world), and searches through it. Whatever number it finds, it searches through the gfx file and displays that sprite. In the beginning, hand coding in a map with over 5000 individual sprites was a pain. It was about a 10+ hour process for one map. I got tired of this and created an IDE. This made it so I could simply click to create a map instead of hand coding it in. Now it can take as little as 30 minutes to create a map.

The IDE:



This is the IDE. More specifically, it is the tab where you create how the map looks. A map has already been loaded, which is why you see a picture in the middle. Let's look at the basics:



The top section, which is circled in red is a set of sprite shortcuts. You can have up to 1000 sprites. The last thing you want to do is to search through a menu list of 1000 sprites every time you want to change your sprite. These shortcuts were created, so you can put your most used sprites into these sections and have easy access to them.

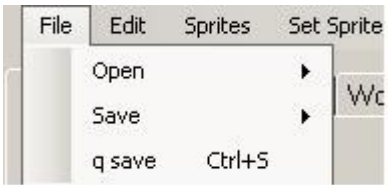
Each sprite in the sprite shortcut section is represented by a number or letter. These numbers and letters represent shortcut keys. These shortcut keys will either assign a sprite to that letter/number or set it as the "current sprite". The shortcut keys are as follows: Ctrl + <number / letter> will set the sprite. Alt+ <number / letter> will set the sprite to the "current sprite". Instead of using the ALT shortcut, you can click on the sprite and that will change it to the "current sprite."

The "current sprite" (to the left of the red circle) is the active sprite. The active sprite will be the one, which is displayed when you click on the map in the middle of the IDE. The "selected sprite" (below the current sprite" displays a sprite that can be assigned to the sprite shortcut section. Below the "selected sprite" is a text box. This text box will display the ID number of the sprite. As you can see, there is a red arrow pointing from the text box to the picture of the sprite. This shows their relation. If you wish, you can type the ID number into this box and get the desired sprite. Or you can search through the menu for the sprite you want. Only numbers can be used as ID's.

** Important Paragraph**
The ID numbers are actually names of the sprites. All of the sprites are stored in a folder called "afw_sprites" under the C drive. You must place this folder under C drive or the IDE will give you errors. You must name the sprites as a number. If you put letters into it, then it will break everything. The names are parsed and put into the world array. That is how the thing works. You must use the exact same numbers for the pictures under the afw_sprites folder as you do for the row numbers for that sprite in the gfx array. Otherwise, your map files will not come out right.

Menus:

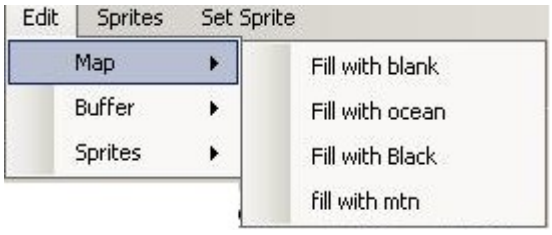
We will now venture into the menu system.



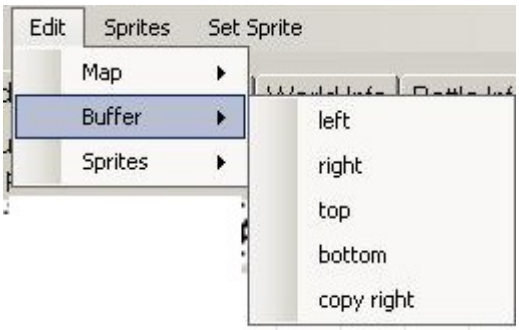
This is the File menu. You have 3 options: open, save, and q save. Open will open of a text file version of a map you saved. Save works like the "save as" option in other programs. It will ask which file you will like to save the map to. If you have not saved the map yet, then it would be good to use this option.

Qsave is a shortcut I created in order to use the Ctrl+S option. Always trying to use "save" caused the program to eventually crash. Adding in the shortcut prevented the crashing. Also, it allowed me to get happy with Ctrl+s. It is a good idea to continuously save. The IDE is not programmed very well. I made it out of a language I haven't touched since .NET 1.1. It is poorly programmed and seems to crash after making 1 or 2 maps. It is a good idea to close the IDE and reopen it every time you create a new map. It may not be the best IDE in the world, but it works and that is what mattered to me. If someone would like to remake the IDE in a good language like C++ or preferably JAVA, then please do. I will be happy to provide the source for this IDE. Just post the new IDE and send me the code.

This is the edit menu. You have 3 choices: map, buffer, and sprites.



The map option allows you to fill the map 4 different types of sprites: blank, black, ocean, and mountain. This basically creates a background for you. For world maps, you will probably want to use ocean, because most of the world would end up being ocean, anyways. Filling it is mountain (mtn) allowed me to make caves. Blank, I used for individual rooms and shops. I used "black" in deep dark dungeons.

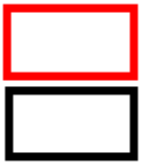


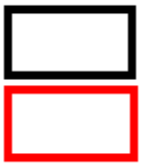
The buffer option is a special option I created to make your maps look good. Originally, I ran into problems with switching from one map to another. As I got to the end of the map, I would see blank area begin to appear. Then as I crossed over to the new map, the screen would blink and the white space would be no more. This just looked bad, so I decided to create a buffer area. If you have maps that are "joined" together in order to create a larger map, like a 200x200 world map, then you will need to buffer the ends of the maps in order to prevent the white space and to make sure the maps look right.

If you decide to make a large map that extends beyond the bounds of 50x100, then you will need to do the following: First, create your first map. When you create your 2nd map, you will need to decide which side of the first map it will be on. For example:

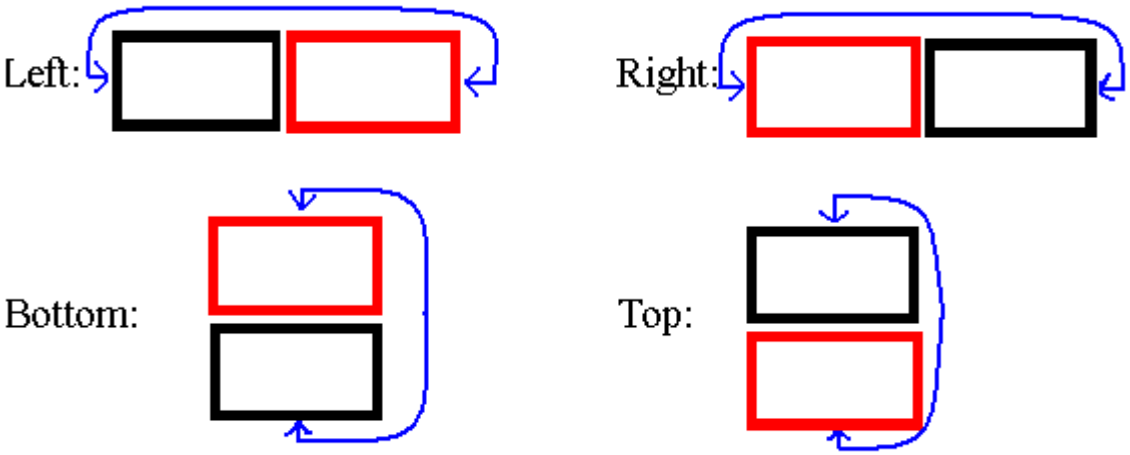
Left: 

Right: 

Bottom: 

Top: 

The above pictures give examples of map layouts, with the first map being the red box and the 2nd map being the black box. If you decide to have the map to the left, then you will have to do a "buffer right" on the 2nd map. If you do the 2nd map to the right, then you will have to do a "buffer left" on the 2nd map. If you decide to have the 2nd map on the bottom, then you will have to do a "buffer top" on the 2nd map. If you decide the top option, then you will have to do a "buffer bottom" on the 2nd map. Let's look at another example:

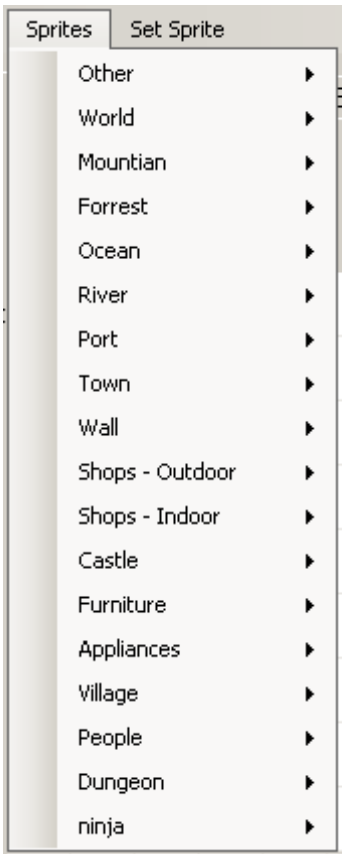


This shows that both the maps link together. For example: the “bottom” scenario shows that the first map is on top while the 2nd map is on bottom. However, the bottom of the 2nd map will go to the top of the first map. In this scenario, you have to buffer both the top and bottom of the 2nd map. You will have to do the same thing with the “top” scenario (buffer both the top and bottom of the 2nd map). In both of the “left” and “right” scenarios, you will have to buffer both left and right on the 2nd map.

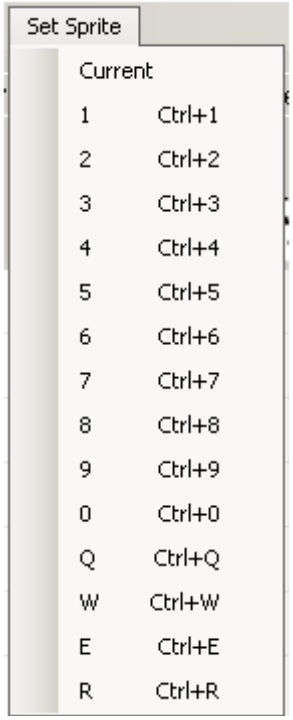
If you make it so that 4 maps are used (assuming they make a square of 100x200), you will have to use all buffers on every map except for the first one. When using the buffer option, you will be asked to load a file. You want to load the world map that will be beside it. The buffer basically gets a few lines off of the selected map and copies it into your current map. From there you can continue on with the making of your map. Play around with it until you get a good idea of how it works and what you should and should not do.

This is the “sprites” section of the edit menu. This basically fills in the “sprite shortcut” section at the top with a set of sprites. It made it easy for me, because I didn't have to search through a huge list of sprites to find the ones I would use the most. Play around with the different options to see what will display.

I have not created a feature where you can add to this menu. You will have to modify the IDE in the event you want more options.



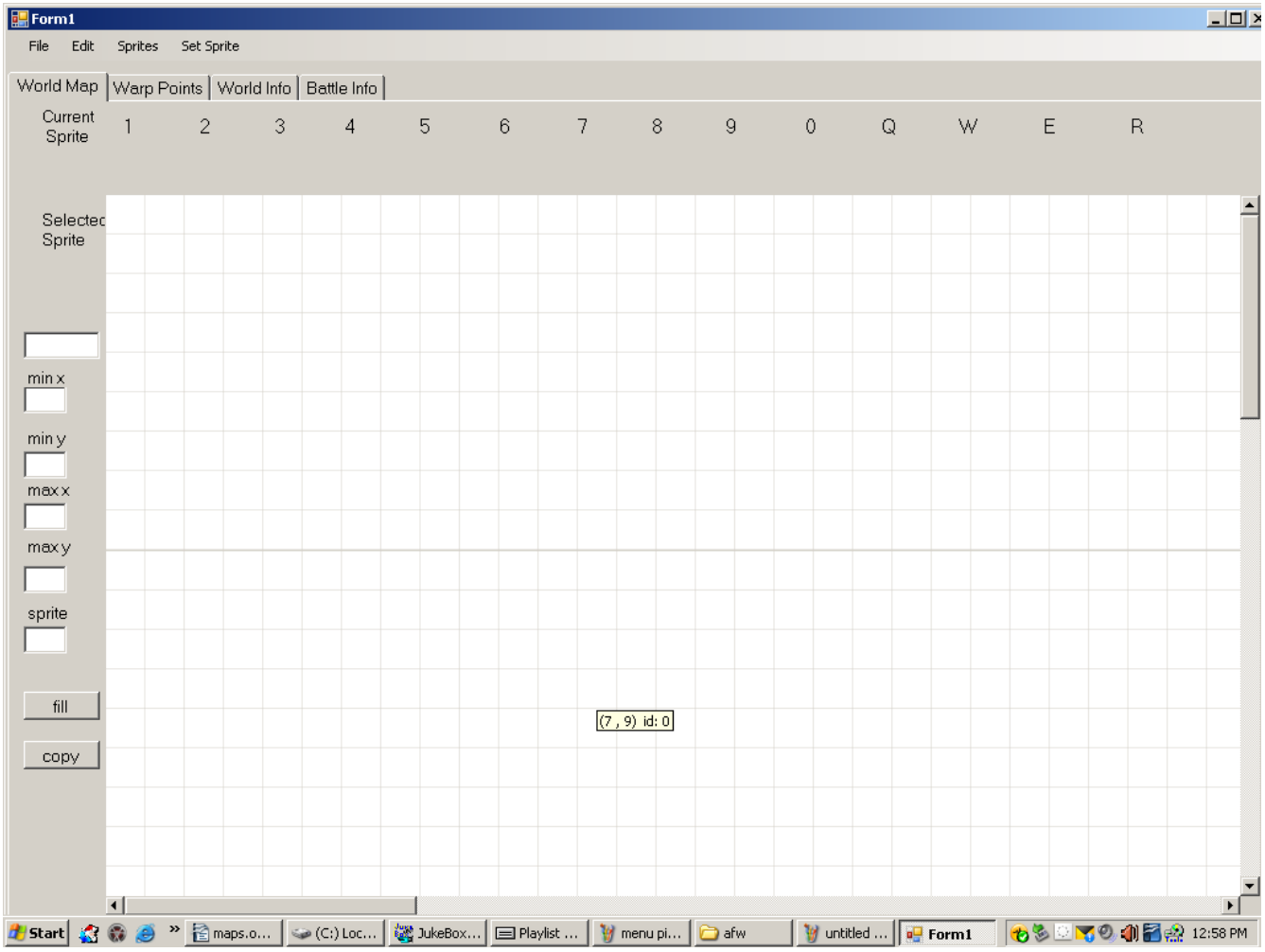
This is the Sprites menu. It holds a huge list of sprites. For some reason this menu seems to be the cause for the program to slow down so much and to cause it to randomly crash. When it gets slow to display this menu, it is a good idea to save everything, exit out of the program, and restart it. Look through the menu and see what you can find.



This is the set sprite menu. This will set a sprite to the sprite shortcut section. As you can see, it has shortcuts, so use the shortcuts.

Making the maps:

Making maps is simple. Set up your sprite shortcuts first. Then set a current sprite. If you are starting with a blank background, then you will see something similar to the following:



If you notice, you see a grid in the middle. This grid is actually a set of picture boxes (1000's of them). Each one is clickable. When you click on it, it will display the "current sprite" in that picture box. If you hover over the picture box with your mouse, you will see a little box pop up with numbers in ()'s with "Id:" next to it. This shows the coordinates at, which your main character will be standing at. This is helpful when attaining your coordinates for a town, so when you walk on top of it, you can warp to that town. The "Id:" will give the ID number of the current sprite, so if you need to get the number for it, it is there for you. Play with it and see how it works.

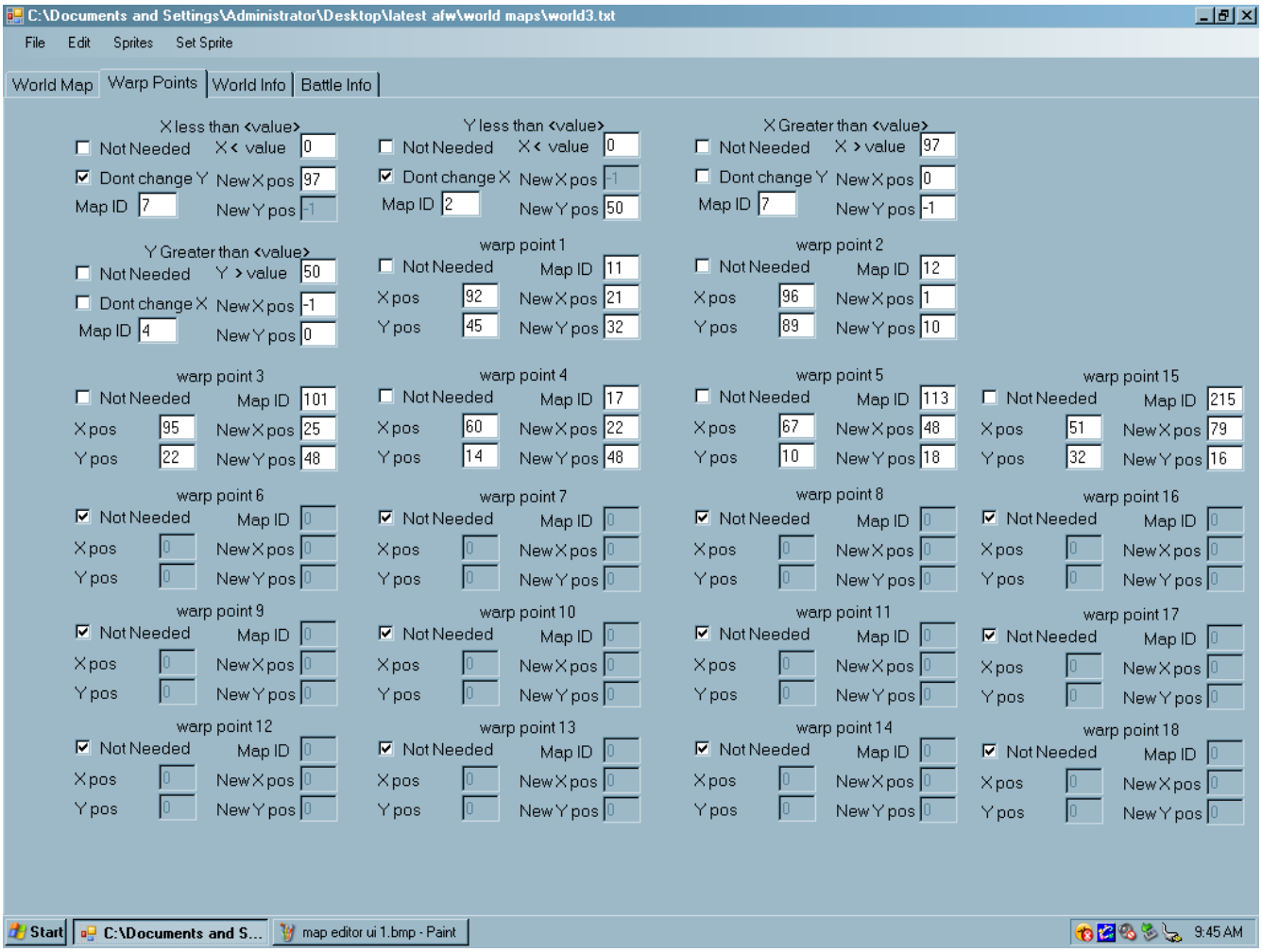
On the left, you see a min x, min y, max x, max y, sprite, and the buttons fill and copy. The fill routine is very helpful. It allows you to fill a square section with a single sprite. This is helpful in floor creation, when you have a lot of floor to cover. This is how it works: use your mouse and hover over a picture box in the map area. It will give you the coordinates. Put these coordinates in min x and min y. Then go right and hover another picture box and get those coordinates and put them into max x and max y. Type in a sprite ID # into the "sprite" text box and then press fill. Done. Play around with it.

The copy button uses the same concept as the fill x button. You specify a square area that you want to copy over. Then in the "sprite" text box, you specify a place on the x axis to copy it to. It is not the best copier in the world, but it does work. And it worked for what I wanted it to do. Play around with it and get to know it.

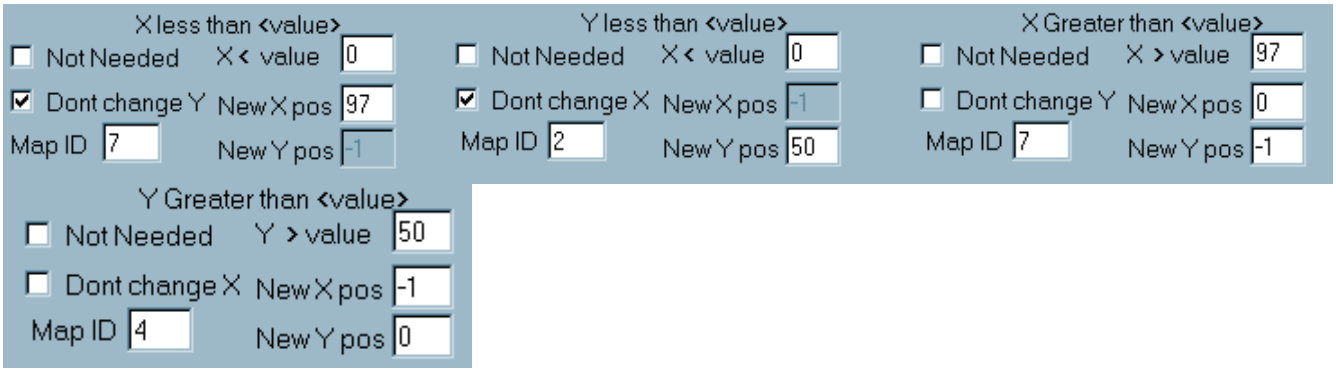
There are some sprites that have color to them. If they are blue in color, then that means that they are walkable sprites. If they are red in color then that means that you cannot walk on them. On the calculator, they will show up in black and white, but it is helpful in the map editor to know whether you can walk on a sprite or not, especially if it is a "walk through" wall.

Warp Points:

This is the warp points tab on the IDE. It will specify the coordinates and the map to go to in the event you walk on a certain (x,y) coordinate.



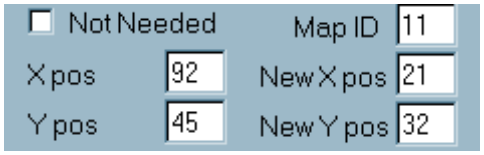
If you look at the top, you will see the following sections:



These sections will allow you to go to a different map in the event that you reach the edge of the map. 2 of them are for the y scale and 2 are for the x scale. If you look at the "X less than <value>" section, you notice that if x<0 then it will jump to map ID # 7. It will reset the Xpos to 97, which is on the right side of map 7. The option to not change Y is there so it will appear that you are still walking in the same place on the world map.

There are times when you will change Y. For example, let's say that you are entering a town or something, you will obviously change Y, because the may layout will be different. You will have to use your own discretion when it comes down to this. All of these boxes work on the same concept. There is something you need to take note of: how to properly set up the warp points to accommodate the buffer areas.

The above picture is from a properly created buffered world map. Those are the numbers you need to use, save for the Map ID numbers. You can change those. As for the others, leave them as is. For those who want it written down, here are the values: On the x scale, use x<0 and x>97. On the y scale, use y<0 and y>50. If you notice, -1 signals no change in either the x or y coordinate. Warning: do not make both new x and new y -1. If you do, you will get errors and you may not ever see your map.



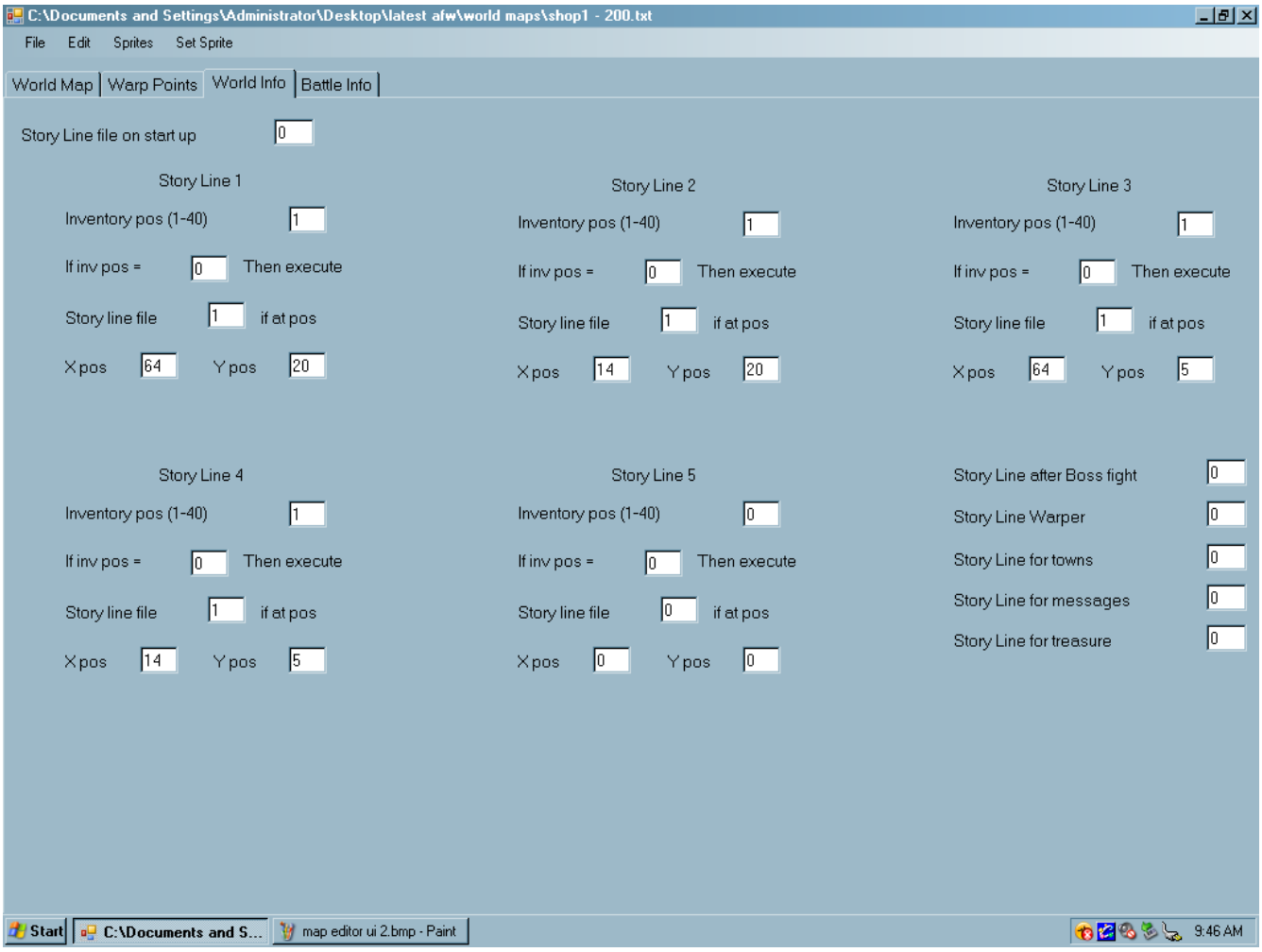
Above is a picture of a warp point section. This allows you to set a town in the middle of a map and when the character walks on it, it will immediately pull up the town map. First you need to grab the (x,y) coordinates by hovering your mouse over the desired sprite. You need to put those coordinates in to the Xpos and Ypos textboxes (this is where the numbers 92 and 45 are). Now you need to find a place where you want your character to appear on the map you want to transfer him to. Put those coordinates into the NEW Xpos and New Ypos text boxes. Now enter the map ID #. The map ID # is the number that appears on the worldXXX file in your calculator under the maps folder. For example, if you are calling world2 then use the number 2 for the map ID.

If you want to warp to some other place on the current map, then put the current map's ID # in the map ID's textbox. If you do not need the warp point, then click on the "not needed" checkbox. If you do

this and you already have info in the textboxes, then it will erase the textboxes. You do not need to have the warp points in order. You can have them in any order. AFW checks all warp points regardless. **Note: you can use these warp points for any warping functions. You will need these for towns, stairs, doors (that lead "into" houses), or other random transporters. **

World Info:

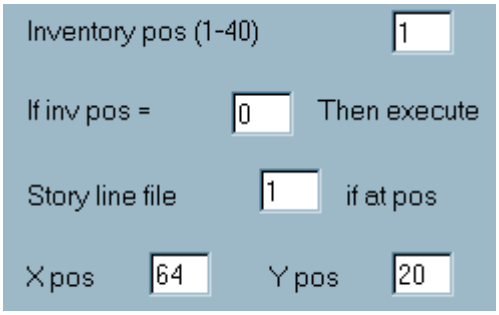
This is the section where you link your storyline files to the map. This is also how you will create shops. For info on storylines, look at the "programming in AFWSL" guide and the "AFWSL syntax" guide in the storyline editor folder. Below is what the tab looks like:



Executing a storyline file upon the loading of a map:

Storyline file for start up is a file that is executed when the map is loaded. Choose an ID number and put it into the text box and save. Now that storyline file will execute when the map is loaded.

Linking shops to the map:



If you want to add shops to the game so people can buy stuff, then you will need the section pictured above to do it. The section should be self explanatory, but for the sake of argument, I will explain it further. The first line asks for the inventory position. This is the inventory marker ID#. It ranges from 1 to 42. On the IDE it says 1 to 40, but it is actually 1 to 42. The number 1 is reserved for the shops, so you are actually going from 2 to 42 if you decide to input another storyline here.

As shown above, the number for the inventory pos is 1; therefore, it is obvious we are creating a shop. Now the checker only executes if the inv pos equals a certain number. If it equals that number, then it will check Xpos and Ypos and see if you are standing at that coordinate. If you are, then it will execute the storyline file. In this case, it is storyline file 1, which is reserved for shops.

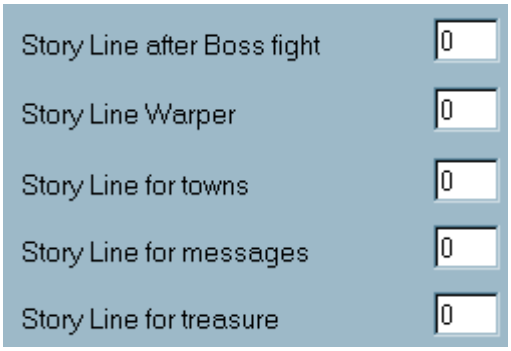
The Xpos and Ypos should probably be in front of some guy selling things. That is how you add shops to maps. This section is not only meant for shops. It was meant for shops and to help aid the plot along. Basically it allows you to execute a script if the inventory marker is equal to a certain value and if you are standing at a certain set of coordinates. You can use your imagination when it comes down to this.

If you notice, there are only 5 different spots that look like this. This means that if you decide to follow this rule to create shops, then you can only have 5 shops per map. I made separate maps that were dedicated to shops. I made the "entering of buildings" take me to one of the shop maps. You can do this if you want, but you do not have to.

Note: these storylines activate when you press the 2nd key.

Food for thought: If you wanted to, you could create a story line that would change the value of inv pos and if it equaled something different, then you could change what the shop holds (ie pull of a different storyline and force a different set of shops to load).

Other storylines:



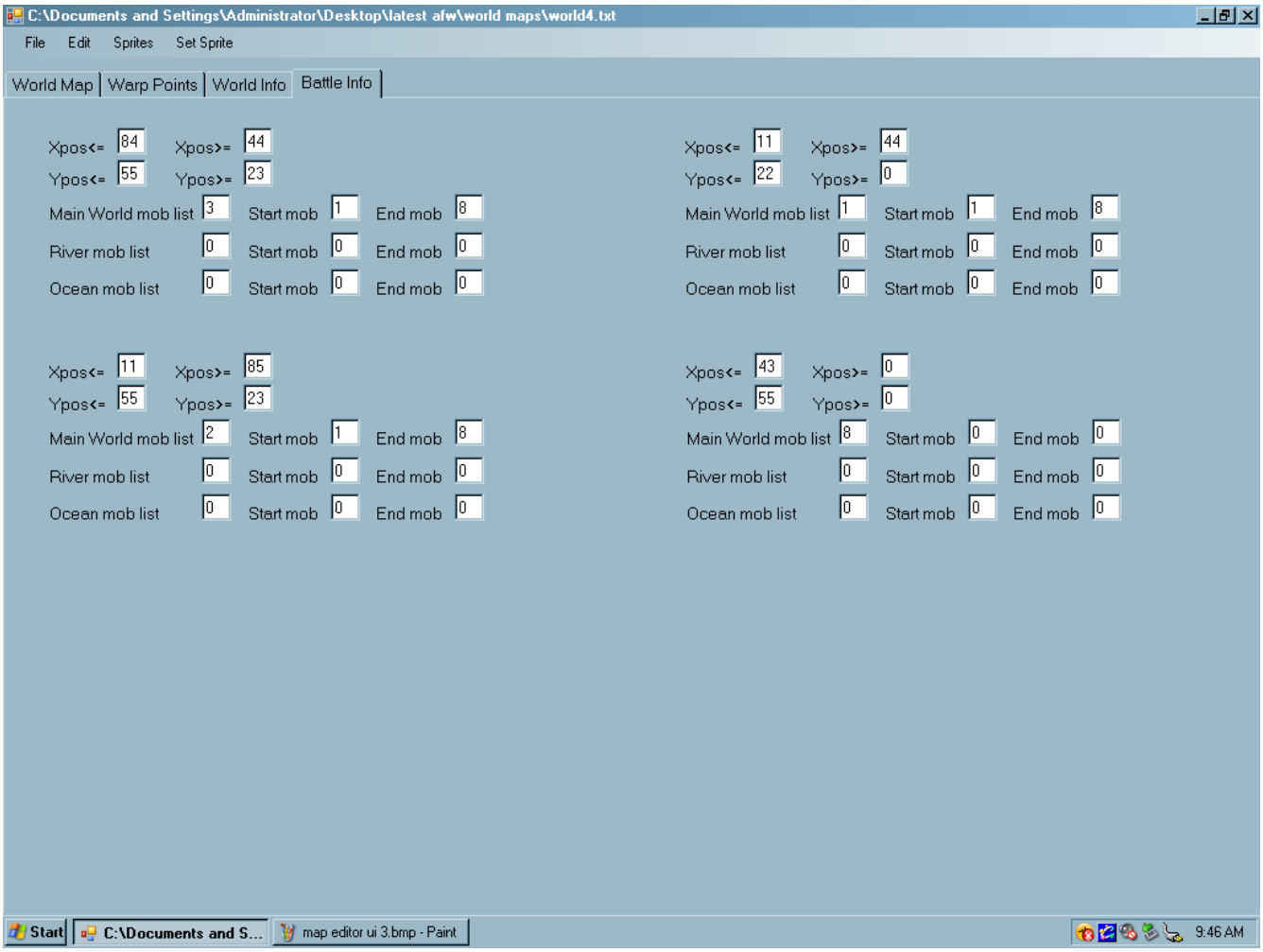
These are the other storylines you can link to a map. The storyline after boss fight is a storyline that is executed after you defeat a boss (this has not been tested). The Story line warper is to expand the warp points from 18 to about 115 (this is loaded upon loading the map). Storyline for towns is a storyline that would set up your towns people and what they say.

Storylines for messages would be a storyline that would activate messages by non-mobile towns people (it could also be for other things if you wanted it to be. It is activated when the 2nd key is pressed). The storyline for treasure is a storyline that is directly created for treasure boxes (it is activated when the 2nd key is pressed). See the programming in AFWSL guide for more information on how to make these storylines. All you have to do is simply put the storyline ID # into the corresponding textbox and then it will be linked.

Adding battles to the maps:

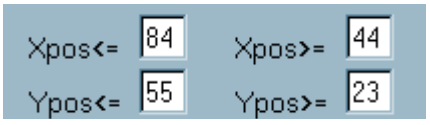
You are able to add up to 4 different battle sections to a single map. Meaning that the creatures you fight can change as you go further south, north, west, or east, if you set things up to do this. You could make the whole map have all of the same monsters, but what fun would that be? The dungeons in final fantasy 1 were maps (floors) smaller than 50x50. If you like this idea, then you can have 2 floors per dungeon on a map. But if you remember: as you went deeper into the dungeons, you fought harder and harder mobs. This battle section allows for those ideas. You can make half your map have one set of mobs and the other half have another set of mobs.

This is also set up to allow you to have different mobs for rivers, land, and ocean (just like final fantasy). Below is what the tab looks like for adding battle information to the map.

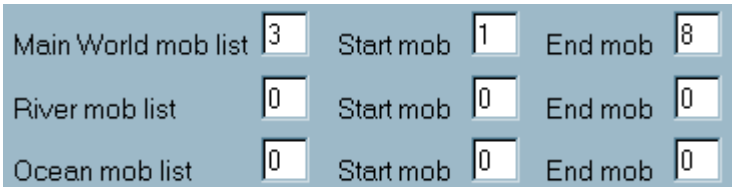


The first thing you will need to do before adding mobs is to create them. Look at the mob creation howto for information on how to do this.

Defining the layout:



The above picture is the section where you will define the range (area), which the mobs will exist in. This will create a square or rectangle area (depending on how you define it). Within this range, a certain set of mobs will exist. Since there are 4 ranges, you must be careful to make the ranges exact and right next to each other. If you do not, then you will end up with areas where you will not get attacked. On the other side of the coin, you could make secret areas, which have no mobs attacking you; thus, allowing you safe passage.



This section defines which mob files to use. If you notice, there are 3 different areas: land, river, and ocean. If you want to fight the same mobs in the ocean that you would on land, then go ahead, but I think it looks better if you fight fish in the ocean or rivers instead of wolves and such. With each area, you need the mob file ID # and you need to specify a range for which mobs to use. Your mob files can have up to 12 mobs in each one. When you specify your range, you are specifying the order, which you have your mobs listed in your file. If you do mobs 1- 4, then you will use the first 4 mobs. If you specify 3 – 9, then you will use the 3rd mob in your list through the 9th mob in your list. This should be easy enough to figure out.

This pictures shows that I am use the 3rd mob file and using a range of 1 to 8. If you run into problems, loading your mobs, then just load all of them. Then make more mob files for the other sections. Each of my areas only had 4 different mobs, but there were 2 copies of each mob. However, the copies had different strengths, levels, and held different items.

There exists a map viewer IDE. This just pulls the map so you can see it. You cannot edit it in any way. It is best if you use 1024x768 resolution if you want to see the whole map.

That is all I can think of when it comes down to the IDEs.

Map creation:

The map making process is a 2 step process: create the array using the IDE. Then use the tigcc map editor to export the file. The tigcc map editor is there to export the world maps, so afw can use them.

First create the map in the IDE. If you don't know how to use the IDE then read the 14 pages above. If that doesn't help, then play around with it until you figure it out. After you do that, you should have saved the file to some sort of text file (you might have to add the .txt to make it a text file, though). If you open up the text file in notepad, you will notice a really long array. Open up the Tigcc map editor. Copy and paste the array in your text file to a header file that you created, or into the world.h file. Then rename the array to what you want.

Go into the map_saver.c file and modify the case statement to add your array (map) to the case statement. The number you put into your case statement will be the ID number of your map, so keep this in mind. After you do this, run the editor and export the file. Finished!

Other Info:

In the original tigcc map editor I provided, there were only the ID numbers of 1 to 4. These numbers can be any number from 1 to 999. Don't use zero. Why? Because it is reserved for battles. I used a simple map ID naming convention:

1 → 10 were for the world maps
11 → 19 were for towns
20 → 29 were for castles
200 → 299 were for shops
100 → 199 were for dungeons

You do not have to use this, but it makes it a lot easier if you know what ID's store what kind of file, otherwise you will have a really confusing time of linking your maps together.

** NOTE: you can only have up to 4 different maps loaded per Tigcc map editor. Why? 64K limitation. **