The Graphics (gfx) Editor

This will simply introduce the graphics editor and tell you how to use it.

Overview:

The gfx engine uses an array of sprites for all of the graphics in the game, save for a few things like the menu system and the intro picture (the picture shown when you first start the game). The intro picture is hard coded in, but I may decide to make it external one day. The array for the gfx system is 1000 x 16 in size. All of the sprites are 16x16. The system is set up, so each row in the array is for one 16x16 sprite. This allows for 1000 sprites total.

Approximately 400 sprites are already created. So this leaves room for another 600 custom sprites that can be added. The main sprites can be removed or redone; however, some of the row numbers are hard coded in. For example: the main system is set up, so certain row numbers are walkable while others are only for the ship/canoe. The battle system display is also set up, so the background is linked to the row numbers. So if you decide to edit the included sprites instead of adding your own, you could end up with unexpected results. Best idea: don't modify.. just add to the sprites.

I originally didn't plan very far ahead in the early stages, so the special sprites are a bit random. However, here are a list of the sprites you can walk on:

Sprite row numbers: 300 to 400, 1, 5, 22, 49, 51, 52, 55, 58, 13, 70, 71, 75, 79, 124, 126, 128, 131, 155, 158, 172, 179, 184, 197, 198, 201, 202, 191, and 208

Sprite row numbers 37 to 41 is for the river. (The canoe will be activated when you walk onto it.)

Sprite row number 33 is for the ocean. (Only the ship can travel on this row number.)

** Note: I created the sprite range of 300 to 400 after I redid the gfx system. The entire range is not filled with sprites. Only a little bit of it is. If you wish to add different sprites, which you want to walk on, then add them into this range. **

Some of the character sprites are also hard coded in, so you need to be careful with these. They can be changed, but you will simply make things look weird if you don't put a characters picture there. Here is a list of defines. (The numbers correspond to the row numbers in the gfx array.)

```
#define prince 580
#define sage 576
#define robes 572
#define lady 568
#define woman 564
#define old_woman 560
#define guard 556
#define bmage 552
```

```
#define wmage 548
#define warrior 544
#define scholor 540
#define char_ninja 536
#define guy 532
#define fighter 528
#define char_rexx 524
#define char_yoki 520
#define char_toki 516
#define char_nick       500
#define char_canoo 504
#define char_ship 508
#define char_airship 512
```

If you notice, the defines are all 4 numbers apart. This is because there are 4 different sprites to each character. One for facing up, down, left, and right. The define numbers are the ones for the character previous to that character. This allows for the directional defines to be combined with it. (This will be explained in other help files.) Keep this in mind: Some of these defines cannot be changed, because they are hard coded in, but you can change the sprites to something else if you wish.

GFX editor overview:

All of the sprites are put into a header file. They are short int arrays, but in hex. It is set up for the sprite16() routine. So it will look like the following:

```
static unsigned short int grass[]={
 0x2200, 0x0020, 0x0001, 0x0808, 0x0000, 0x9080, 0x0008, 0x0000,
 0x0420, 0x8002, 0x0000, 0x0090, 0x0800, 0x0000, 0x8020, 0x0000


};

static unsigned short int grass_bottom_left[]={

 0x8000, 0x4408, 0x2040, 0x2000, 0xd220, 0xd000, 0xe802, 0xc880,
 0x1410, 0xca00, 0xe900, 0xe0c2, 0xf430, 0xf70c, 0xefb2, 0x8f7d

};
```

The above naming method is a good one to remind you what the sprite is, but it can be quite annoying later on. In the end, I found it easier to put the row number as the name, for example:

```
static unsigned short int a235[]={
  0xffff ,0xffff ,0xffff ,0x0921 ,0x628d ,0x2549 ,0x4545 ,0x654d
 ,0x610d ,0x654d ,0x644d ,0x654d ,0x4105 ,0x2549 ,0x654d ,0x0821
};
```

Of course you cant give them a number as a name, so I used aXXX as the name, where XXX is the row number. Why give them a row number name? Simple. In order to create the external gfx file, I use a switch case statement in order to fill the array with info. Then I simply export the array to a file. For example:

```
while (i<1000)
{

        switch (i)
        {
                case 0: ptr=bl; break;
                case 1: ptr=grass;      break;
                case 2: ptr=grass_bottom_left;          break;
                case 3: ptr=grass_top_left;     break;

…

                case 316: ptr= a316;break;
                case 317: ptr= a317;break;
                case 318: ptr= a318;break;
                case 319: ptr= a319;break;

}}
```

ptr is the pointer that is used for the sprite. The loop increases the variable i and the case statement, which is testing the variable i, goes in and places the current sprite into the pointer. After the case statement, the pointer is placed into the array at row i. This is how all of the sprites are placed into the array. The default sprite is called nosprite. This is so that all rows are filled with at least some type of info, so you don't run into garbage. If you make a large set of sprites, you can see that it would be a lot easier to just name the sprite as the row (case) number, so you don't always have to look back and try to figure out what the name of that sprite was.

Adding new sprites:

In order to add new sprites, make a 16x16 picture. Convert it to hex using something like Istudio. Place it in an array in the header file. Then modify the case statement, so it will add the current sprite into the array. Compile the thing in tigcc (v .96 beta 8 recommended) then run it in the calculator. I recommend using an emulator first to see if it works. If it does, simply export the file from the emulator to an external file and then upload it to your calculator. Just running the file should create the external gfx file automatically. You should not have to do anything else, except to archive it.

File List:

map_saver – this holds routine to write the gfx file.
gfx – holds the while loop that fills in the array. Also has the case statement.
mapcreator – holds the main routine.
Sprits.h – header file for most sprites
sprits2.h – header file for the newer sprites I created.
mapping_sys.h – header file for access to routines.