Mob creation HowTo

Mobs are the creatures you can fight. Mobs were made external, so the game could be expandable and due to size limitations. Each set of mobs are stored in 2 separate files: a file for the stats of the mobs (mobsX) and a file for the graphics (gfx) for the mobs (mobpX). S stands for stats and p stands for pictures. The external files are simply arrays. The gfx array is called "mob" it is 48x16 in size. The stats array is called "stats" and is 12x41 in size.

Each mob can have up to 4 sprites associated with it. 4*12 = 48. This is why the gfx array is 48 rows long. Each row is for a single 16x16 sprite. This means that you can have up to 12 mobs per file, although I usually use 8. Each row in the stats array is for the stats of a different mob.

Stat Array overview:

The stat array is a long int, so you are not limited to 65535. The following is the stat array:

stats[X][0]= this is for the mob ID (use a low single or double digit here)
stats[X][1]= this is the mob HP
stats[X][2]= this is the skill % of the mob
stats[X][3]= this is the spell % of the mob
stats[X][4]= spell 1 the mob can cast
stats[X][5]= spell 2 the mob can cast
stats[X][6]= spell 3 the mob can cast
stats[X][7]= mob's str
stats[X][8]= mob's int
stats[X][9]= mob's dex
stats[X][10]= mob's def
stats[X][11]= mob's absorb %
stats[X][12]= mob's level/weapon damage (all mobs do H2H combat)
stats[X][13]= item 1 carried by the mob
stats[X][14]= item 2 carried by the mob
stats[X][15]= item 3 carried by the mob
stats[X][16]= spell slot 1 that the mob can be affected by.
stats[X][17]= spell slot 2 that the mob can be affected by.
stats[X][18]= spell slot 3 that the mob can be affected by.
stats[X][19]= spell slot 4 that the mob can be affected by.
stats[X][20]= spell slot 5 that the mob can be affected by.
stats[X][21]= spell slot 6 that the mob can be affected by.
stats[X][22]= whether the mob is vuln, neutral, or resistant to fire
stats[X][23]= whether the mob is vuln, neutral, or resistant to lit
stats[X][24]= whether the mob is vuln, neutral, or resistant to ice
stats[X][25]= whether the mob is vuln, neutral, or resistant to water
stats[X][26]= spell 1 that the mob is immune to
stats[X][27]= spell 2 that the mob is immune to
stats[X][28]= spell 3 that the mob is immune to
stats[X][30]= exp gained from killing the mob (make sure it is 4 or more – if 0, then divide by 0 errs)

stats[X][32]= amount of gold that the mob gives at the end of battle
stats[X][39]= this determines whether the auto feature is turned on. Use TRUE and FALSE
stats[X][40]= this will tell afw what sprite design to use. (4x4, 1x1, or 1x2)

Skill% and spell % - this is quite useful. In the early game when you have a low percentage of hitting, you are not always going to land a hit. Having the mob hit 100% of the time is deadly. In order to even out the game, I gave the mobs the same disadvantage as the players had: they had to deal with a low hit percentage. It makes early battles go slower, but makes them survivable and winnable. The spell percentage rate has the same concept. If a mob is made for early in the game, then make sure the skill % is low or comparable to a normal players skill percentage at that point in the game.

Mob's level – all mobs use H2H combat when doing damage. H2H combat is based off of that mobs level, so the higher the level, the more damage the mob will do.

Mob stats – the mob stats (str, def, dex, int) work the same way as the characters stats do. The mobs dex will be placed against the characters dex in the event of attacking, being attacked, and during the event that you try to run from battle. If the mobs have a lot higher dex than your characters, then you may never be able to run and you will simply die.

Mob absorb % - works the same way an armor absorb % does. The range is 1% to 100%, where 100% means that the mob will never take any damage.

Items carried by the mob – this is fairly simple. I allowed the mobs to have a 33% drop rate of items in the event that you kill them. Mobs can carry up to 3 items. The items can be different or the same. If the items are the same then you get that many of those items. The process is simple. All you have to do is put the item ID number into this column and the mobs will drop the item(s) upon death. The ID number can be of items, weapons, or armors, special items, etc. You could do magic, but then a magic spell would end up in your inventory and you couldn't use it. You would have to drop it or something. General rule: don't add magic or skills to the "items" that creatures can give you if you want to do this, then use AFWSL after a boss.

Spells the mob can cast – mobs can cast up to 3 different spells; however, they are not bound by mana restrictions like your characters are. They have infinite mana, which means that they can cast the hell out of the spells you give them, so be careful. In order for them to cast a spell, just put the spell ID number in that column and then they will cast it.

Spells mobs can be affected by – mobs can be affected by up to 6 different spells. Usually you will leave this blank, not unless you want the mob to have haste or sanc. These spells do not wear off the mobs, so beware. Once they are on, they do not come off not unless you dispel them. This section is usually used by the game, so you may not have to worry about it.

Vuln, neutral, and resistant to elements – there is no magic absorb on creatures; however, they have a different way of taking magical damage. If they are vuln (vulnerable) to an element, then they will take 2x the damage. If they are neutral, then they take normal damage from the element. If they are resistant (res), then they will take half the damage from the element. Assigning vulns, neutrals, and reses are done with 3 different numbers: (-1), 0, and 1. (-1) is for vuln, 0 is for neutral, 1 is for res.

Immunity to spells – mobs can be immune to up to 3 different spell types. All you have to is put the spell type into the column and it will become immune to that spell. This works only for maledictions.

Exp gained – this is the exp your party members will gain after the mob is killed. You will immediately gain the exp when the mob is killed, which means that you could level during battle. Also, the exp is divided equally between all party members. So if the mob gives 8 exp upon death and you have 4 party members, then you will have 8/4 = 2 exp per person. Keep this in mind. **Note: keep the minimum of 4 exp. If you give 0 exp, then you will get a divide by 0 error and crash the game. If you give less than 4 exp and you have 4 party members, then they will get 0 exp each. **

The auto feature – the ultimate form of laziness. Trying to make mobs was a pain, so I created a routine to do it for me. If you enable the auto feature, then it will automatically fill in all the stats for the mob. You should probably always use this feature, because it will, for the most part, keep the game balanced. Here is the thing though: all of the stats will be based off of the mob's level. You can give the mob a level yourself or let the auto feature do it for you. If you let the auto feature decide the level, then it will base the level of the mob off the average level of all of your characters; thus, making it just like final fantasy 8, where the mobs level with you. Although this may be neat, it can really be a pain in the ass for anyone who plays the game. General rule: you should always give the mobs a level when using this feature, not unless you just have a dungeon where the mobs must be difficult.

The design (column 40): This lets afw know what kind of picture to display when trying to display the mob. There are 3 different setups: a mob that is 32x32 in size (this uses 4, 16x16 sprites in a square), 16x16 in size (this uses 1, 16x16 sprite), or 16x32 in size (this uses 2, 16x16 sprites placed beside each other). There are 3 different numbers to define what gfx setup to use: 1, 2, and 4. 1 is for the 16x16. 2 is for the 16x32 and 4 is used for the 32x32 setup.

Mob creation:

First you need to create sprites for the mob. All sprites are 16x16. If you want a 32x32 mob, then you need to break the picture up into 4 separate 16x16 sprites. If you want a 16x32 mob, then you need to split it into 2 different 16x16 sprites. After you create your sprites, use a program, like Istudio, to convert them to hex. Convert that hex to a proper array that can be used by the sprite16() routine. Then put the array into the sprites.h header file.

Create your new mob header file and modify the stats array to your liking. At the end of each row of the array you will need the following routine (of course you will need to edit it to work right):

```
i=0;
while (i<16)
{
                mob[28][i]=creep_p1[i];
                mob[29][i]=creep_p2[i];
                mob[30][i]=creep_p3[i];
                mob[31][i]=creep_p4[i];
                 i++;
}
```

This while loop will loop through 16 times. It is filling the mob array with the gfx information for the sprites you created. This particular one is filling in a 32x32 picture for the mob called the creep. If you have less than 4 sprites to fill in, then simply use the bl[i] array I have already created. This will prevent garbage from happening.

Here is an example of what a 16x16 sprite input will look like:

```
i=0;
while (i<16)
{
                mob[20][i]=spike2[i];
                mob[21][i]=bl[i];
                mob[22][i]=bl[i];
                mob[23][i]=bl[i];
                i++;
}
```

This is what a 16x32 sprite input will look like:

```
i=0;
while (i<16)
{
                mob[28][i]=tails1_p1[i];
                mob[29][i]=tails1_p2[i];
                mob[30][i]=bl[i];
                mob[31][i]=bl[i];
                 i++;
}
```

You need to keep in mind that the row numbers for the mob array will be related to the row numbers of the stats array. The above row numbers are for example purposes only. If you make all your row numbers like that, then you will only end up with one mob and nothing else will display. Ie, you will have errors, so don't do things that way. Instead, look at the mob1.h through mob9.h header files and use them as examples on how to code in your mobs.

Those files are shortened and do not use the full array, because the arrays are automatically filled in, because I used the auto feature. The parts of the arrays that do exist use the certain parts that I needed to hand modify. Using this method made it easier for me, because I only had to edit certain things for all mobs; thus, making mob creation faster. You can redo the mobs or use them if you wish.

Side note: stats[x][31] is no longer used – at least I don't think it is, so don't bother with it. If you get errors, then simply set it to 1. Also columns 33 → 38 are not used. I thought ahead and allowed for expansion, just in case I wanted to add extra things for mobs. That is why those exist.

After you create your header files, you will need to include them in all 3 C files (mapper, itemedit, and map_saver). Your header files will, also, be in a routine that you can call (just like the message editor). You will need to add that routine to the case statement in the mapper.c file. Keep in mind that the number you use for the case will end up as the file ID number for the mob file. So case 88: will end up as mobs88 and mobp88.

After you do that, compile and run the program. Enter the file ID number you want to save the file to and there. You are done. Keep this in mind: the file ID number is important and it is used in order to load the correct file on your maps, when you create them. More details about this in the map creation howto.