Modding Faq for Notrium ver. 1.3

For your ease of use I have provided an Index. Some sections are broken up into smaller sections which you can easily skip to by using the Find command in many text editors with the number and letter combination following the section. For example to quickly skip to the Areas.dat data file breakdown you'd use A3 in the file find and you'd be brought to that section. You can also use the decimal and letter combination following a section to quickly skip to it as well. I hope that all our loyal modders find this FAQ useful and I apologize ahead of time for the length of it.

Index 0.0Q

Getting Started 0.3Q

What do I need to get started modding?

Well here is a grocery list of things you should have before you attempt to mod Notrium.

1. **Basic knowledge of how to open and close files and the basic operations of your computer. Being able to navigate the folders, open, close, and save files is a must.**

2. **A simple text editor. This will be required if you want to edit any of the data files. I also highly suggest Jan's File Viewer as an alternative, though be aware you still probably be using a text editor at some point.**

3. **A program to edit and create graphics. A good free 2d editor is Gimp 2.0. I don't know of any superb 3d graphics programs. You can also alternatively have someone create graphics for you.**

4. **A program to create or edit sound effects. You can also have someone create sound effects for you as well.**
5. **A program to create or edit mp3 music files. Like graphics or sounds you can probably get someone to help you with this as well. There are many unsigned and eager bands on mp3.com that may not mind having their music showcased in your mod.**

6. **Patience. You're going to make mistakes, a lot of mistakes. So go slow and don't add several things before you test what you've already added. It is very easy to bury**

yourself in problems and not be able to find what is causing them because of the simple fact you didn't test things at a reasonable pace.

7. **Someone to test your work. You're going to be biased; either thinking your work is too good or too bad. Get at least one other opinion before you add in tons of things. What seemed like a good idea may just turn out to be an entirely frustrating gameplay issue. Jan's File Viewer is fantastic for sniffing out syntax errors for you.**

Data File Listing 0.5Q

Before you plow ahead you should at least learn what each of the data files do. There is a severely larger amount of files to deal with in 1.3 than there was in 1.2. So don't think that being an expert at modding 1.2 makes you an expert at modding 1.3. So here is a breakdown for old and new modders alike. All of these files will be found in your /Data folder. If you can't find that, stop now and refer to the first ingredient of Getting Started. For your own mod you're going to want a folder within the /data folder that has the exact name of the mod you'll be creating. You'll probably want to use the Barebones mod on the Notrium site as your starting point if you're going to make a completely new mod.

AI_Tactic.dat - This file dictates how a creature friendly and unfriendly, react to you and each other. Each creature has two AI Tactics by default.

Animations.dat - This file is where you define the various comic book style cut scenes in your mod.

Areas.dat - You'll define the multitude of Areas used within your mod with this file.

Bars.dat - The various bars like health, energy and whatever else you can conceive is originally setup here.

Climate_types.dat - If you've tooled around with the Areas.dat file than you're probably going to mess with this too. This is where you define the actual climate used by an area.

Creatures.dat - All creatures are defined with this file including the player races.

Dialogs.dat - Got something witty to have your creatures say, or maybe they'll just grunt. This is the file you set it up with.
Items.dat - Inventory items are setup in this file.

Journal.dat - This file can be called whatever you like actually and you should have several, one for each player race. This is the player races journal that is shown to them as they play through the game. It is primarily used to deliver hints and storyline elements.

Light.dat - This is where you define all the lights used throughout the game.

Map.dat - This file sets up how your areas you defined in Areas.dat are arranged in the actual game.

Music.dat - This file sets up the music that is played during the game either in the main loop or that will be cued by effect.

Object_Definitions.dat - This sets up the graphics for plot objects and props along with several other options you can change to your liking, like collision detection.

Particles.dat - This is where you define all the particles used throughout your mod. An example of a particle is footprints, rain, radar dots, ect.

Player_races.dat - The characters the player actually gets to play in your mod are defined here.

Plot_objects.dat - Plot objects are basically just props or objects you can interact with. This is where you edit and create them for your mod.

Polygons.dat - This is where you define collision detection types. I seriously suggest against messing with this file as several basic collision types are already provided.

Scripts.dat - Timed events, endings and other various things are done through scripts. Chances are if you want to do something fancy you'll be using a script to do it.

Sides.dat - You get to define who hates whom in this file. What side a creature is NOT decided in the Creatures.dat file. It is decided during its placement into the game by either an effect, the Areas.dat file or in the game's Map Editor.

Sounds.dat - Want to cue a sound within the game, this is the file where you define a sound that can be cued with a special effect I'll cover further into the FAQ.

Terrain_Maps.dat - You don't touch this file other than to make changes to the name. While you can edit this file entirely I seriously suggest against it. This file is created exclusively for use by the in-game Map Editor. You should not be messing with this file other than to name your Terrain Maps.

Terrain_Types - This is where you define the various ground types used within the game.

Weapons.dat - Any weapons you plan to be used within the game are defined here. Basically anything you want to be done with the left mouse button during the game is going to be done in this file.

Data File Breakdown 0.7Q

Ok the purpose of this section is to better acquaint you with each and every file. This is extremely important if you want to know what you're doing.

AI_Tactic.dat A1

```
Tactic Name;//tactic name
```

```
0;//identifier
  begin_levels;
  20;//anger level size
    1;//action
    0;//parameter0
    1;//can hit enemies
    1;//can eat
  end_levels;
end_of_file;
```

That is as simple as the file will ever get. One entry with only one anger level. Each tactic's block begins with it's *;//tactic name* **and ends with** *end_levels;***. If you accidentally chopped off one of those you're going to have trouble getting your mod to work right.**

Each angel level begins with *;//anger level size* **and ends with** *;//can eat* **and must be placed between** *begin_levels;* **and** *end_levels;***. It's very important you know how blocks are setup within a file so when you copy and paste a section you get everything you need.**

Every file will end with *end_of_file;***, this one is no exception, if it's missing it Notrium wont be happy.**

Ok so now you understand how the blocks in this file are setup. Now I'll tell you what each part of the anger level does to the best of my knowledge.

;//anger level size **defines how long the creature will stick with this anger level. Basically a creature starts with 0 anger and as they become more aware it switches to the next anger level. The larger the level, the less likely it will move down to the next anger level or move back up to the previous anger level.**

;//action **tells the creature exactly what it should be doing. The numbers you can use are listed at the bottom of the file, but I'll break each down for you.**

0: is good for when you simply don't want the creature doing anything at all.

1: is good for if you want the creature to just randomly move around.

2: is good if you want the creature to follow creatures that are friendly to it, you'll use the parameter to set how close an enemy needs to be for them to break off and attack them

3: simply has them attack an enemy for as long as their anger level is large. The larger you made the anger level the longer they'll keep after their enemies. So to make really them fight like skirmishers you'd want to keep the anger level smaller, for really ruthless and unrelenting creatures you'd want it higher.

4: is the exact opposite of 3 as it makes them run away from enemies. The larger the level, the longer they'll run away from the enemy.

5: makes a creature guard the spot they were spawned at, parameter0 tells them how far they can go from the spot to persue an enemy.

6: simply makes the creature fire in random directions.

;//can hit enemies **tells the game if the creature can attack while it is in this anger level. If set to 0 the creature wont be able to attack.**

;//can eat **tells the game if the creature can eat its eat item. If this is set to 0 the creature will not eat its eat item defined in the Creatures.dat file.**

Well now you know how each part of the anger level block works and how the blocks are arranged. On to the next file.

Animations.dat A2

```
animation;//name---------------------------------
0;//identifier
animation.jpg;//texture
1;//frames
What you want to show in the game goes here.;
50;//text_y
0;//start_x
0;//start_y
1024;//end_x
768;//end_y
end_of_file;
```

Here you have as basic as the animations file will ever get, down to one entry. You don't actually have to use this file for anything other than to end the game since there is no other way to end the game besides through the animation effect I'll cover later. Like before you'll notice the name starts the entry and the *;//end_y* **ends the entry.**

Inside of the actual entry the frame block begins with the line right before *;//text_y***. Since it isn't labeled you'll have to remember each frame block begins with the text line that ends with** *;* **and ends with** *;//end_y***. It's important to know that the game needs as many frame blocks as you tell it are there with the** *;//frames* **line. It's also important to note that the animations are done in a comic book style where each cell of the image is displayed at a time with some text. So all you're really doing with each frame is telling it what part of the same image to display and with what text.**

As before you'll want your file itself to end with *end_of_file;* **as shown in the example above.**

Now I'll break down *;//texture* **line down to the end of the entire animation block.**

;//texture **tells the game what graphic to use for your comic book or single frame animation. The image should be 1024x768 and in jpg format. I'm not sure if this is rigid requirement but it is best to stick to it regardless for simplicities sake.**

;//frames **tells the game how many frames long your animation will be. Remember that for however many frames you put in this entry you need to provide frame blocks.**

After the frames comes whatever text you want displayed, always ending it with a *;* if you are manually inputting it and not copy and pasting. This text line should always starts a frame block.

;//text_y **tells the game where to put the text while the particular frame of the animation is showing. About 300-400 is centered but it will vary depending on how much text you're display for the current frame. The more text, the lower you'll want the number if you want it centered. It's also important to remember your text will obscure the image so you'll want to place it where it looks best, which isn't always exactly centered.**

0;//start_x
0;//start_y
1024;//end_x
768;//end_y **These four lines tell the game the box dimension of the image to display for this frame. The first two define the upper left corner of the box's dimension while the last two define the bottom right of the box's dimension. This example above is good for a single frame animation since it shows the entire image in the first frame. If you still want just only the text to change than you simply use the same box dimensions as the last frame and put in new text.**

Now you should have a firm understanding of how the animations.dat file works and how its blocks are set up.

Areas.dat A3

```
0;//-------------------------------------------------------
Area A;//area name
0;//area class
-1;//climate in climate_types.dat, -1=random climate
1;//terrain map number
-1;//wrapping, -1=no wrapping, 0=wrap horizontal, 1=wrap vertical, 2=wrap both
1;//random object density, the bigger the value, the more the area will get.
This is multiplied ingame by the area size.
begin_alien_list;
  1;//alien type in creatures.dat
    1;//number of these aliens
    3;//side
  2;//alien type in creatures.dat
    2;//number of these aliens
    1;//side
end_alien_list;
begin_excluded_plot_object_classes;
0
end_excluded_plot_object_classes;
Put the text you want to show when the player enters the area for the first
time here.;//enter area text, specify none to not show it
end_of_file;
```

Here you have a very basic area example from the Areas.dat file. There are 2 blocks within each Area block. As you can see the entry starts with *0;//* **and ends with the** *;//enter area text***. The number of each area should be sequential or 0, 1, 2, 3, ect.**

The first block you'll come across within the file is the alien list block which is 3 lines long for each entry within it. It begins with *;//alien type* **and ends with** *;//side***. The alien block itself is began with** *begin_alien_list;* **and ended respectively with** *end_alien_list;***. Any creatures you want randomly placed within the are going to end up in this block. It's important to note that there doesn't have to be anything in the alien list. You wont be using this block if you're not going to place random creatures. Creatures placed precisely are placed with the Map Editor in the game, not the Areas.dat file.**

The second block you'll deal is very simple, but references a file I don't cover till further into the FAQ so bear with me. The purpose of the excluded plot object block is to keep a particular plot object class from appearing in the area. All you do is place one number per line between *begin_excluded_plot_object_classes;* **and** *end_excluded_plot_object_classes;***. This can be done for a variety of reasons. One reason could be you've got a special set of plot objects that you want randomly placed (more than just one plot object or it's not worth it.), so you simply exclude that class from the areas you don't want it in. So you can have several areas with plot objects unique to them.**

So now that you've got a general idea of the blocks for this file I'll break down the lines for you. Since I already told you how the area number works I'll start with *;//area_class***.**

;//area_class **tells the game what class your area is. You can make this whatever you like. This is for setting up isolated areas or for setting up several randomly generated sections of the game. On the whole this is how you will group common areas.**

;//climate **tells the game what climate number to use for this area. Climates will be covered in detail further into the file. Climates will control the day and night cycles of the area, what props are placed in it and also how rain works in it.**

;//terrain map number **is the number of the terrain map the game references to build the area. Terrain maps are designed with the Map Editor. This has a strong effect on how your area looks and also the specific placement of everything from terrains(floor tiles), to creatures and plot objects. I'll cover the Map Editor in detail further into the file.**

;//wrapping **is a new feature among many. This basically just sets up if the area has any kind of wrap around. In other words if you leave one side will you come back in the opposite side. What you set this to determines what kind of wrapping it will use. You're generally aren't going to use this, so you'll want it set to -1. If you want the right and left to wrap around you'll use 0. If you want the top and bottom to wrap around you'll use 1. You'll use 2 to make all sides of the area wrap.**

;//random object density **tells the game how tightly to pack your area. If you want no random objects you set this to 0. If you want a normal amount you set it to 1. Less than 1 will give you a reduced amount naturally. Since it is a multiplier you don't want to go too high. For example setting it to 2 will make the object density 2 times the normal amount. So I wouldn't recommend using more than 2 in most cases otherwise the area will become seriously cluttered. It's also important to note that there is a balance between all the areas and increasing the random object density in one area will reduce random objects in the others and the total random objects is still set in the Plot_objects.dat file.**

begin_alien_list; **simply start the alien list block. All Aliens blocks must be after this line.**

;//alien type **tells the game what creature to use. You'll naturally be referencing the Creatures.dat file for the number of the creature you'll use. This also happens to be the start of a Alien block**

;//number of these aliens **does exactly what it sounds like. It tells the game how many of the creature type to randomly place within the area.**

;//side **determines what side the randomly place creatures will be on. You'll be referencing the Sides.dat file this line. This also happens to tbe end of the Alien Block.**

end_alien_list; **simply ends the alien list block. All Aliens blocks must be before this line.**

begin_excluded_plot_object_classes; **begins the excluded object class list. All excluded classes must go after this line.**

end_excluded_plot_object_classes; **ends the excluded object class list. All excluded classes must go before this line.**

;//enter area text **is the text that is displayed when the player first enters the area. This text is only shown the first time. Make sure to put none if you don't want anything showing. This is also the last line of an Area's entry. So either** *end_of_file;* **comes after this line if it is the last entry or a new Area's entry will.**

Now you should have a basic idea of how the Areas.dat file is setup.

Bars.dat A4

```
bar a;//name
  0;//identifier
  0;//bar type, 0=normal, 1=show body temperature, 2=show number of items
parameter0, 3=show wielded weapon ammo, 4=show armor, 5=show carry weight,
6=show seconds from game start
    0;//parameter0
  1;//visible (only applicable for player's bars)
  1;//visible on enemies
    1;//show number
    3;//anchor point (0=left top, 1=right top, 2=left bottom, 3=right bottom)
    -250;//location x, pixels from anchor point
    -20;//location y, pixels from anchor point
    bar.png;//bar picture
    none;//background picture
    0;//background picture x offset
    0;//background picture y offset
    0;//background picture width
    0;//background picture height
    1;//height
    -200;//length in pixels for maximum (negative = bar increases to left)
    1;//red color component for minimum
    0;//green color component for minimum
    0;//blue color component for minimum
```

```
    0;//red color component for maximum
    0;//green color component for maximum
    1;//blue color component for maximum
end_of_file;
```

This file is fairly important to a degree, but the bottom line is you actually don't have to use bars at all, ever. So with that in mind if you want things like a health bar, stats for your characters, or perhaps an experience bar than you are going to use this file. One thing to realize is that you are limited to 30 bars in version 1.3 so you'll want to be sure that you need a bar and can't do it another way. If you're not careful you'll find you don't have enough bars left to do the really important things. Even with a full game, we only used 15 bars, so you shouldn't really be using more than that unless you're making a complex rpg or simulation that requires you keeping track of a bunch of values.

The only block you need to worry about with this file is the start of the entry which begins with *;//name* **and the end which is** *;//blue color component for maximum***. Other than that there are no blocks within the file and you'll find this file to be one of the simplest to edit.**

So now you know how a entry begins and ends I'll break down each line for you. I'm going to group them this time for the lines that don't vary much.

;//name **is pretty self-explanatory. It simply gives the bar a name. Mind you that this name will be shown in the game when the player floats the mouse over the bar. The name will not show if the bar is not visible.**

;//bar type **tells the game what function the bar serves as. Generally you'll leave this bar at 0. 1 is used for temperature, it is only really useful if you want it to be effected by the temperature effect. If you want to keep track of how many of a certain item you've got than you'll use 2. When you want to show how much ammo a wielded weapon has left than you'll use 3. Armor is shown with bar type 4. When you want display or keep track of the carry weight you'd use bar type 5. The final bar is 6 and it keeps track of seconds from the start of the game, its good for setting up timed events.** *;//parameter0* **is only used with bar type 2 to tell it which item it is keep track of.**

;//visible **tells the game if the bar should be displayed or not. If you're only going to use it as a counter you'll want this set to 0. On the other hand for things you want the player to see always make sure it is set to 1.**

;//visible on enemies **despite its name means it's visible on any creature that isn't the player. So if you're going to use a special bar to keep track of something for timing or otherwise on a creature you can make it invisible by setting this line to 0. Generally all you want showing for a non-player creature is its health and nothing else.**

;//show number **tells the game if it should show a numerical value or not. Setting this to 0 simply shows the bar graphically, without a number overlaying it. Setting it to 1 will show a number.**

;//anchor point **is actually more important than you might initially think. One should always anchor the bar in a position closest to the corner where the bar will be shown. To anchor it**

to the top left you'll use 0. The top right anchor will be 1. Setting the anchor to the bottom left will be 2. Last but not least using 3 will anchor it from the bottom right. The bar should retain its relative position even through scaling of the window if you use the proper anchor points.

;//location x, pixels from anchor point
;//location y, pixels from anchor point **both of these adjust the bar's placement from the original anchor point. The first line will adjust the x coordinate placement of the bar from the anchor point in pixels while the second line will adjust the y placement of the bar from the anchor point in pixels. For the x coordinate a positive value will shift the bar to the right while a negative value will shift it to the left from the anchor point. For the y coordinate a positive value will shift the bar down while a negative value will shift the bar up from the anchor point.**

;//bar picture **is what graphic your bar is made up of. Basically this defines the shape and look of your bar. Since the bar is going to be colored in it's best you use a 8 bit greyscale png.**

;//background picture **is the image that underlaid beneath your bar.**

;//background picture x offset
;//background picture y offset
;//background picture width
;//background picture height **These four lines control the placement and scaling of your background image. The first two lines set the x and y offset of your image from the bar's actual placement. The last two simply control the scaling of the image. You'll probably have to toy around with these a bit till you get the desired results.**

;//height **sets how tall the bar will be graphically, 1 will use the** *;//bar picture* **at 16 pixels in height while 2 will make it 32 as the height is actually a multiplier of 16.**

;//length **sets the maximum length of the bar. If you set the length to a negative value it will increase to the left and shrink to the right. Vice versa if you set it to a positive number.**

;//red color component for minimum
;//green color component for minimum
;//blue color component for minimum
;//red color component for maximum
;//green color component for maximum
;//blue color component for maximum **These six lines all control how the bar is colored in the game. If it's being used for a creature besides the player it will always be red. It's pretty self-explanatory and all you really need to know is that its set up by RGB on the maximum and minimum ends of the bar. So if you want purple for instance you'd do red with 1 and blue with 1. The minimum colors will be the ones that show first and than as the bar grows it'll blend into the maximum color. If you just want it one color than you set them both to the same RGB values. This also the end of the entry for a bar. After this if there is no other entry tha** *end_of_file;* **should proceed it.**

Climate_Types.dat A5

```
Climate A;//name-------------------------------------------------------------
-------
0;//identifier
1;//can be a random map
200;//light oscillate time
  0;//light r phase shift
  -0.1;//light g phase shift
  -0.1;//light b phase shift
  1.0;//light r amplitude
  1.0;//light g amplitude
  1.0;//light b amplitude
  0.7;//light max r
  0.5;//light max g
  0.5;//light max b
  0;//light min r
  0;//light min g
  0.3;//light min b
-0.8;//night temperature
0.8;//day temperature
0.5;//maximum wind speed
8;//rain particle, -1=no rain
300;//rain_particle_life_min
800;//rain_particle_life_max
15000;//rain_probability (the bigger the number, the more unlikely the rain)
5000;//rain_time_min
15000;//rain_time_max
  begin_effects_block;
    It's raining!;//event text
      100;//interval
      begin_conditions;
      end_conditions;
      begin_effects;
        0;//effect number
        0.000000;//parameter1
        0.000000;//parameter2
        0.000000;//parameter3
        0.000000;//parameter4
    end_effects;
  end_effects_block;
start_of_terrains;
  1;
  2;
end_of_terrains;
start_of_night_sounds;
night.wav
end_of_night_sounds;
start_of_day_sounds;
day.wav
end_of_day_sounds;
start_of_props;
0;//object definition number        puu0
10;//number of these on a 100*100 map
end_of_props;
end_of_file;
```

Here we have a single entry version of this file. This example is slimmed down considerably and it is a definite possibility your entries will be much longer than this one. The first block your need to understand as usual is how the climate entry itself begins and ends. It will always begin with *;//name* **and end with** *end_of_props;***. One additional thing of note for**

math buffs out there, especially if someone wants to make a small program that will calculate how the light change will look is that the light amplitude and phases discussed further into this breakdown are actually calculated from sinus curves, so the amplitudes and phases are actual phase shifts and amplitudes of real mathetical sinus functions. I know that last line will be confusing for most but it may serve some greater purpose should there be any mathematicians out there reading this FAQ.

This is the first file I've covered so far to use the effect and condition blocks, so this is getting a bit more complex now. The main block is the effects block which starts with begin_effects_block; and ends with end_effects_block;. Within that block you have 3 smaller blocks. An effect block itself starts with *;//event text* **and ends with** *end_effects;***.**

Condition blocks are universal throughout the data files and any variances with the files usually comes before the condition block begins. The condition block is naturally started with *begin_conditions;* **and ends with** *end_conditions;***. Each condition is 3 lines long and they merely go one after another so long as they come after the beginning of the block and before the end of it. So for example two condition would be 6 lines long and 8 with the block beginning and ends included. All conditions are listed in the notes.txt file and will be broken down for you later in this FAQ.**

Effect blocks are also universal throughout the data files so once you become accustomed to one you can edit the rest just as easily. The effect block begins *begin_effects;* **and ends with** *end_effects;***. Each effect is 5 lines long and like conditions they go one after another. So for example two effect would be 10 lines long inside of the larger block, making it 12 lines total with the block beginning and ends added in. All effects are listed for you in the notes.txt file and will be broken down for you later in this FAQ.**

The next block you'll come across is terrain block. This block basically tells the game what terrains(floor tiles), that the climate is allowed to use for random placement when it builds the area using this climate type. It always begins with *start_of_terrains;* **and ends with** *end_of_terrains;***. Within this block simply goes one terrain number per line followed by the ; as you can see in the example above.**

I'm going to cover the next two blocks in one explanation since they both works the same. The sounds you want played during the night are placed one per line between *start_of_night_sounds;* **and** *end_of_night_sounds;***. The sounds want played during the day are placed one per line between** *start_of_day_sounds;* **and** *end_of_day_sounds;***. You can refer to the example if this still doesn't make sense to you but they are very simple blocks to work with.**

The final block I'll be covering for this file will be for placing props when this climate type is used. The prop block starts with start_of_props; and ends with end_of_props;. Each prop entry is 2 lines long and consists of *;//object definition number* **which tells it which object definition this prop will use and** *;//number of these on a 100*100 map* **tells it how many to place if the map were 100 tiles high and 100 tiles long. Remember if your map is smaller than 100*100 the number will be proportionally reduced and the same goes for if it is larger. So a map that is 80*80 will generally have 20 percent less than the number you use, while a map of 120*120 will generally use 20 more.**

Since this file like many that will come later consists largely of blocks I've already explained I'm only going to break down the lines that aren't part of a block. Before I get started it is important to know that r represents red, g represents green and b represents blue in the lines that are dealing with lighting. So *;//light r phase shift* **for example is talking about the red channel's phase shift.**

;//name **works the same as the other files but is not showed anywhere within the game so it purely for your own reference.**

;//identifier **is what the entry is referenced by when called by the Areas.dat file. This number should be sequential.**

;//can be a random map **tells the game that if and area from the Areas.dat file has -1 set as its climate type can this be used in the random pool of climates to be chosen from.**

;//light oscilate time **is basically how long the night and day phase of the lighting in this climate will last. So basically the night and day phase will be half of this value each. This is done in seconds so 1 is really one second. This is important to not as alot of timed functions done in the game are done by the thousandth of a second where 1000 is 1 second. This is one of the exceptions.**

;//light r phase shift
;//light g phase shift
;//light b phase shift **These three lines control the way each color channel appears. Setting a value to a negative makes that color channel appear later in the phase of the day. An association is to picture three color horses and the phase shift controls how fast each horse and will arrive at the finish line. Sunrise and sunsets are created using these lines. For instance if you'd want the lighting to gain a reddish hue during sunset and sunrise you would make sure the red channel was more prominent by setting the other color channels to a negative value.**

;//light r amplitude
;//light g amplitude
;//light b amplitude **These three lines controls how the light bends from night to day. Basically with each set to 1 the light bends at a normal curve. Setting the number higher one color or all of them will cause a flatter curve that will become closer to a straight line and remove the more gradual blend from one color to another, making the change very sharp. Tweaks to these 3 lines are generally not very noticable so this is really not something you have to mess with and leaving each at 1.0 will be fine for most lighting schemes. For something like flashing lights you'll want the amplitude to be higher which will make less of a graduation and blend between the two colors.**

;//light max r
;//light max g
;//light max b
;//light min r
;//light min g

;//light min b **These 6 lines are the final ones you deal with when dealing with the lighting in a climate and also the most important. The first three lines control the RGB blend of light shown during the day phase. The final 3 lines control how the RGB blend of light is shown during the night phase. Many special light effects can be done with these 6 lines but it is important to note this will be the default lighting scheme. So if you plan on changing the lighting only temporarily you'll want to use an effect instead.**

;//night temperature
;//day temperature **These 2 lines control the temperature of a climate. The first line as it clearly states controls how hot or cold it is at night time, while the second controls how hot or cold it is during the day. For no temperature you'd put 0. A positive number will make the player hot while a negative number will make them cold. This has absolutely no effect if you omit a temperature bar from your mod.**

;//maximum wind speed **controls how strongly the wind will blow at its maximum. Not only does this line effect how hard rain appears to come down but it also effects how strongly plot objects or props that have the option to blow in the wind are shifted as well. This is something you will probably have to tweak till you get the desired result with 1.0 being a normal wind speed and 0 being no wind at all. Even a value of 3 will cause very strong winds so you really shouldn't go much higher than that.**

;//rain particle
;//rain_particle_life_min
;//rain_particle_life_max
;//rain_probability
;//rain_time_min
;//rain_time_max

All of these six lines all control the rain while the block that directly proceeds it controls what happens when you are exposed to the rain. The first line tells the game what particle from the Particles.dat file to use, if you set this to -1 there will be no rain. The next two lines determine how long the particle will remain. The minimum is the least amount of time a particle will remain so the rain will be around for at least that length while the max sets the maximum amount of time the particle can remain. The fourth line in this group controls how often it will rain, settting this number higher decreases the chance of rain, while setting it very low will cause an almost constant stream of rain. The last two lines work just the same way as the particle life min and max but control how long it actually rains for. If you want to create something like a flash flood you could set the probability fairly high but make the rain time longer so each time it rains it last for a while, but is still rare, which is how flash floods generally are.

The rest has already been covered in the block explanations so you should have no problem editing this file confidently.

Creatures.dat A6

```
creature;//name-------------------------------------
  0;//identifier
  0;//class
```

```
  creature0.png;//texture
  1;//draw layer
  -1;//corpse item number in items.dat, -1=nothing, -2=whole corpse vanishes
when dead
  0;//corpse item amount
  0;//attack type, 0=shooting attack, 1=close combat
  18;//particle to show on radar, -1=none
  0;//primary AI tactic from AI_tactic.dat
  0;//secondary AI tactic from AI_tactic.dat
  3;//footstep particle from particles.dat
  0;//die after how many seconds, 0=doesn't die
  -1;//eat item number from items.dat, -1=nothing
  0.950000;//creature size
  0.950000;//creature weight (affects pushing)
  0.800000;//maximum movement speed
  0.000000;//minimum movement speed
  0.800000;//leg animation speed
  10.000000;//turn speed
  1.000000;//inertia level (acceleration/deceleration)
  1;//hide when behind walls
  0;//weapon number
  1.000000;//weapon_x
  27.000000;//weapon_y
  0;//blood particle
  1;//bars visible when player hovers mouse on top
  1.000000;//death animation speed
  1;//creature can move from area to another
  1000.000000;//AI hear range
  500.000000;//AI see range
  4.000000;//AI see angle
  begin_footstep_sounds
    step0.wav;//sample name
  end_footstep_sounds
  begin_hit_sounds
    grunt0.wav;//sample name
  end_hit_sounds
  begin_die_sounds
    grunt2.wav;//sample name
  end_die_sounds
  begin_eat_block
  end_eat_block
  begin_death_block
  end_death_block
  begin_hit_block
  end_hit_block
  begin_timed_block
  end_timed_block
  begin_specialties
  ;//description
    1;//number
    0.000000;//parameter0
    0.850000;//parameter1
    0.000000;//parameter2
    0.000000;//parameter3
  end_specialties
end_of_file;
```

Now we come to the Creatures.dat file. This file is almost the most complex file you'll be editting next to the Player_races.dat file. What will stump alot of you on this file is the multitude of blocks it uses and that each block does have a very specific setup. If you don't setup a block properly your mod will do strange things and possibly not run at all. To get you started the entry always begins with *;//name* **and ends with** *end_specialties*.

The first three blocks I'm going to cover in one explanation since they all work the same. The first is for what footstep sound this creature will make. The sounds is simply placed between *begin_footstep_sounds* **and** *end_footstep_sounds***. The entry inside of this block will be the same for all three, you'll simply put the file of the sound wave before** *;//sample name***. The next block is the sound/s that play when the creature is hit, this time you place the entry between** *begin_hit_sounds* **and** *end_hit_sounds***. In the final sound block you'll place the entry between** *begin_die_sound* **and** *end_die_sounds***. The sounds in the final sound block will play when the creature is killed. [It is important to note that the sound waves you'll be using must be in the /sounds folder or a subfolder within that folder using the same name as your mod.]**

The next block you come across after the three sound blocks is the Eat block. This block controls what happens when the creatures eats its Eat Item. The formatting for the entry inside this block will be found at the bottom of your Creatures.dat file. It contains within it a Condition Block and a Effect Block. If you wish to make several Eat Block entries than you must include the Condition Block and Effect Block for each and they must be within *begin_eat_block* **and** *end_eat_block***. Even if the blocks are left blank you must include the header and footer for their block. It is important to note that if no condition is provided that the creature will execute the effect each time it eats one of its Eat Item.**

After the Eat block comes the Death Block. This block controls what happens when the creature dies. A creature is dead when one of its bars reaches 0. Make sure you never set a bar that you're using merely as a variable to 0 when attaching it to a non-player creature because that will be considered a dead creature. The block is unique in that asides from the usual condition/effect block you'll see used alot it has a *;//death type* **proceeding it. That proceeding line tells it what type of death it must suffer for the effects to be executed regardless of if the condition block is true, 0 for any death, 1 if only by timed death and finally 2 if killed by the player. The entry for a Death Block always includes the** *;//death type* **line followed by the Condition Block and Effects Block and will always be between** *begin_death_block* **and** *end_death_block***.**

Next block you come to is the Hit Block. This block is very basic much like the Eat Block. This controls what happens when your creature is hit. All it includes is a Condition Block and Effect Block. This happens to be one of those blocks you rarely use. The entries for this block must be placed between *begin_hit_block* **and** *end_hit_block***.**

Now we come to the Timed Block. If you plan on doing anything fancy with your creature I can almost guarantee you'll be using this block. This block is basically the equivalent of a script for the player. The Timed Block's entry begins with *;//interval in milliseconds***, which will proceed the Condition Block. The first line is very important, that is how often the game which check the conditions for the entry and be able to execute the effects. Each entry should contain the** *;//interval in milliseconds* **followed by the Condition Block and Effect Block. One thing you must be aware of is that if you only want something happening once you'll have to use a toggle with a bar that changes its value and a condition to make sure the bar is a certain value, otherwise the entry will keep repeating each time the interval comes around. There are of course otherways to make sure it doesn't loop, but that is one suggested method.**

The last and a very important block is the Specialties Block. This block does three things. When the specialty is set to 0 it will add a bar to the creature. Player races do not need their bars added in the Creatures.dat file, they're added in the Player_races.dat file instead. When the specialty is set to 1 than it is a modifier for a weapon class, either making it do more, less or no damage. The specialty being set to 2 will attach a light to the creature, this is how you'll have a creature glow or even emit particles depending on what light you use. The entries for this block should always be between *begin_specialties* **and** *end_specialties*. **Each entry for this block should be lines long and start with** *;//description* **and end with** *;//parameter3*.

As before I'm going to skip explaining the block lines since I already covered them a fair amount.

;//name **is the name your creature will go by. This isn't actually ever shown in the game, at least not yet. So this is purely for your own reference. Try and make this unique to make it easier to find the specific creature you're looking for.**

;//identifier **is the number that will be associated with your creature. This is the number that other files and effects will reference when they do something with this specific creature.**

;//class **is the creature class this creature belongs to. This is something you may not ever use it all. The purpose is merely for grouping your creature with others when doing checks. There are certain effects and conditions which check a creature class rather than a creature's specific identifier. So if you want this being checked with several others than make sure their class is all the same, yet unique from the rest of the creatures.**

;//texture **is the graphic being used for your creature. The graphic used for the texture is done in cell format. I'll cover this in greater detail later in the FAQ. Like any other referenced texture the graphic must be in either the /textures folder or a subfolder within it that is the same exact name as your mod. This graphic can be several sizes but is always divided into 16 equally sized cells. The normal, low resolution size is 256x256. A medium resolution creature is 512x512 and a high resolution graphic is 1024x1024. It's very important that using medium and high resolution creature's is going to use alot more memory. So you should be picky about what creatures really need the high detail that will show better in a medium or high res creature graphic.**

;//draw layer **is what layer the creature is drawn upon. The default layer is 1, setting it to 2 will make the creature be layered ontop or appear flying.**

;//corpse item number **tells the game what item the corpse of the creature will give when the player right clicks it. If you don't want an item to be given then simply make it -1. If you want the creature to vanish completely upon its death than make it -2 which removes the corpse graphic entirely.**

;//corpse item amount **is how many of the corpse item should be given. If you're not giving a corpse item simply make this 0.**

;//attack type **tell the game how the creature should approach combat. If set to 0 it will only get as close enough to get its weapon within range. For creatures that attack up close you'll want to set it to 1 instead.**

;//particle to show on radar **is the particle that is shown when they have a bio detector type item in use. The particle will show on their minimap. This is how you give a creature its unique radar signature.**

;//primary AI Tactic **tells the game what AI Tactic from the AI_tactics .dat file to use. For non-player controlled creatures this is going to be the tactic it uses always unless it is changed with an effect.**

;//secondary AI Tactic **works the same as the above line except that this Tactic isn't used by default. This AI Tactic is generally used for Ally creatures and/or charmed creatures. This tactic can be changed with an effect.**

;//footstep particle **is the particle shown as a footprint for a creature. If the creature floats generally you wont be using a particle and you'll just set this to -1. Creatures that are supposed to be under the ground should probably also leave no footstep particle as well unless you want an effect like the ground rippling to be done with a particle. This particle can be over-ridden by a terrain map's settings.**

;//die after how many seconds **is used for creatures that have a limited lifespan. If the creature doesn't die after a set time simply set to this -1. By default you're probably not going to have you creature die after a set amount of time.**

;//eat item number **tells the game what item from the Items.dat file will be used as the Eat Item for the creature. Be aware that a creature can only have one eat item.**

;//creature size **is how much scaling will be done on your creature's graphic. If set to you 1.0 your creature will be around 64x64, if set to 2 it will be around 128x128 in size. This works in relation with the** *;//texture* **line.**

;//creature weight **is basically how much the creature is moved from weapon kick and knock back from other creature's weapons. Very light creatures will be shoved further and heavier creatures will be shoved very little. Generally you can just match the weight to the creature size unless the creature is unusually heavy like a barrier. This weight will also affect the pushing of other creatures as their relative strength to push through creatures of comparitive weight. So a heavy creature will have no trouble pushing through a group of lighter weight creatures while it will have trouble pushing through something weights more than it.**

;//maximum movement speed
;//minimum movement speed
;//leg animation speed
;//turn speed

;//inertia level **I'm going to cover these five lines in one grouping since they all closely related. The first and second lines control the maximum and minimum speeds your creature will be moving at. Even at 0.6 a creature moves decently fast. You generally don't want a creature much faster than 2. If you set the minimum speed to higher than 0 you can make the creature in perpetual motion, in other words it never stops. If you want to do a missle type creature you'll want to set the max and min to the same value, the value you want the creature to always be moving at. The third line controls how fast the leg animtion is played, you'll need to tweak this, but generally it should match the maximum speed. The fourth line is for turn speed and tells the game how fast the creature can turn. The final line in this group is interia and controls both acceleration and how quickly a creature can slow down to a complete stop or it's minimum speed. High interia will make the creature slide around like it is on ice.**

;//hide behind walls **tells the game if the creature will be hidden when behind a wall. You only option is 0 for no and 1 for yes. Creatures that are used as containers or destructable objects should have this set to no.**

;//weapon number **is the weapon the creature will be using from the Weapons.dat file. Unlike a player race, a normal creature will only ever have one weapon.**

;//weapon_x
;//weapon_y **these setup the point from where bullets come out of your creature. If set to 0 on both it'll come from the creature's center. You'll probably need to tweak this a bit if the creature doesn't shoot from it's center, otherwise all you'll usually adjust is the** *;//weapon_y* **to how much infront of the creature the bullet should appear from.**

;//blood particle **is the particle that is used when you creature is struck by an attack.**

;//bars visible when player hovers mouse on top **tells the game if the bars the creature has are shown or not. Your only option is either 0 for no or 1 for yes. An alternative is make a bar hidden only for non-player creatures which is done in the Bars.dat. This way you can have a bar that is used as a simple counter but is not shown.**

;//death animation speed **is how fast the death animation is played. Despite the default being 1 I suggust something lower like 0.9 to 0.8 or even slower since the death animations are very fast by default.**

;//creature can move from area to another **tells the game if the creature is allowed to leave the area it was spawned if. If set to 0 the creature will never leave its area, even to chase you. If set to 1 the creature can chase you into a new area to keep up the pursuit.**

;//AI hear range
;//AI see range
;//AI see angle **These 3 lines help fine tune how aware your creature is. The first line determines how far your creature can hear or if set to 0, it is deaf and cannot hear. The default we used for hear range was 1000. The second line determines how far the creature can see, generally we set this to 500 but you can set it higher. The final is the angle the creature can see at to give you a couple of quick numbers, since it is done in radians which**

is alot of fun math you'd probably rather not deal with I'll list a couple for you. For 180 degrees it's 3.14, the default is 4 so it is a little past 180, probably more like 220 degrees. The max you can set it to is 6.2 which is a full 360 degrees of sight, in other words it can see even fully behind it. Each of these line's values are measured in pixels and when a creature sees or hears a hostile creature its anger level will rise.

Well that covers this file and its numerous blocks. You will more than likely make mistakes in this file simply because of how massive each creature's entry can get. Be very patient and careful when dealing with this file.

Dialogs.dat A7

```
Incoherent garbage;//name
  0;//identifier
  Danger Will Robinson!;//text
  1000;//duration
  -1;//next dialog line (-1=none)
  1;//text color r
  1;//text color g
  1;//text color b
end_of_file;
```

This is a very easy to file to edit and deal with. First thing you'll notice is it doesn't reference any other file. You will also not be dealing with any block other than the entry itself. You may never use this file, so if you don't want dialog in your mod you can simply skip over this file's breakdown. The beggining of the entry is *;//name* **and the end if** *;//text color b***. Per usual after the final entry** *end_of_file;* **should be there.**

Now for a line break down.

;//name **is the name for the entry. The name will not show and is purely for your own reference.**

;//identifier **is the number that will be associated with this dialog entry. Make sure this is sequential per usual.**

;//text **is what will be displayed over the creature's head that uses the dialog. You should generally keep them short and easy to understand so the player can read it quickly enough before it fades.**

;//duration **is how long the dialog will be displayed in milliseconds. 1000 milliseconds is equal to 1 second. Make sure the text is displayed long enough for it to be easily read.**

;//next dialog line **tells the game what dialog this one will change to once it has vanished. To do a simple loop all you need to do is have the dialog one changes to change right back to the original.**

;//text color r
;//text color g

;//text color b **These three lines determine what color the dialog will be when displayed. The highest value is 1 and the lowest is 0. You can make any color you wish by blending the values to get the desired color. If all are set to 1 than you get pure white and all 0 is pure black.**

You should now have a firm understanding of how this file works. On to the next one.

Items.dat A8

```
Item A;//name--------------------------------------
  0;//identifier
  0;//item class
  5.000000;//weight
  The text shown in your inventory.;//short description
The text shown the first time you obtain this item.;//first pick up text
  0.350000;//size on map
  1;//show on radar 0=does not show, 1=show on scanner, 2=show even without
scanner
  20;//particle to shown on radar
  1;//visible in inventory 0=no (can only be used via quick keys), 1=yes
  diode.png;//texture name
  -1;//wielded script,-1=nothing, only applicable for wieldable items
  -1;//wielded item disabling script, -1=nothing, this script is only checked
for conditions while the item is wielded, if the conditions prove false, the
item is unwielded
  1;//show help text for conditions
  begin_wield_slots;
    0;//wield slot number
  end_wield_slots;
  begin_effects_block;
  You used it!;//event text
    none;//event failure text
    use.wav;//sound, none for nothing
    0;//vanishes after event, 0=no, 1=yes
    U;//use key
    -1;//quick key
    begin_conditions;
    end_conditions;
    begin_effects;
    end_effects;
  end_effects_block;
  begin_combinations;
    1;//discard this item after combine
    2;//combines with
    1;//discard that item after combine
    2;//combine time
    1;//can be broken up
    begin_items_given;
      3;//result of combine
      1;//amount to give
    end_items_given;
  end_combinations;
end_of_file;
```

You'll be using this file very often. This file has a fair amount of blocks and can get complicated very easily, so make sure to test your additions and edits often when changing this file. To start you off on the right foot the entry always begins with *;//name* **and ends with** *end_combinations;***.**

The first block you come across is the Wield Slots Block. This block tells the game what slots if any the item will take up when it is in use. Making an item use slots makes it toggleable, though it isn't the only way to do a toggling item. The block always begins with *begin_wield_slots;* **and ends with** *end_wield_slots;***. The entry inside this block is** *;//wield slot number***. The forementioned line is what tells it what slot/s will be used. One entry goes to each line and an item can take none, several, all, or just one slot, but each must be one slot to each line.**

The next block you come to is the Effects Block. The entries for this block should be between *begin_effects_block;* **and** *end_effects_block;***. There are several lines before the Condition/Effect Blocks begin. The first is** *;//event text* **which is what the game will show each time you use this item successfully. The next line is** *;//event failure text* **which does the exact opposite of the line above it, showing text when you fail the use the item successfully. After those two lines come** *;//sound* **which tells the game what sound to play when the item is used. Proceeding those lines is** *;//vanishes after event***, you only have two choices 0 for no, or 1 for yes. Basically the line says should the item be removed when it used. The next two lines are** *;//use key* **and** *;//quick key***. They both cause the item to be used but in different places, use key is done in inventory and quick key is done out of inventory and can be any single letter or number key you wish. After that comes the Conditions Block which should have 3 lines per entry within it and the Effect Block which should have 5 lines per entry within it. If you wish to have multiple keys you always start with** *;//event text* **and end with** *end_effects;***.**

The final block is the Combinations Block. This block controls what items the item can be combined with and what the results of the combination are. It begins with *begin_combinations;* **and ends with** *end_combinations;***. Entries for this block begin with** *;//discard this item after combine* **which tells it to discard the current item during the combination process, your options being 1 for yes and 0 for no. The next line is** *;//combines with* **which tells it what item you're going to combine it with. The third line is** *;//discard that item after combine* **tells it to discard the item listed in the line above it or not, your options being 1 for yes and 0 for no. The fourth line is** *;//combine time* **and it tells the game how long it takes in seconds to complete the combination (The player cannot move while combining). The fifth and final line before the item block begins is ;//can be broken up and tells the game if the combination can be undone or not, 0 for no and 1 for yes (If set to yes the combination is permanent and undoable).**

The final part of Combination Block's entry is the Item Given Block which begins with *begin_items_given;* **and ends with** *end_items_given;***. Within the block are two lines for each entry. The first line is** *;//result of combine* **and tells the game what item to give as a result of the combination. The second and final line of entry for the Item Given Block is** *;//amount to give* **which tells the game how much of the item denoted in the line above to give the player.**

Now that you have a fair understanding of the blocks I'll cover the remain lines.

;//name **is the name of the item. This name will be shown in the game. Try to keep it on the shorter side but distinguishable from other items especially if you're using the same**

graphic for several items, you'll want to avoid using the same name as well as it is confusing for the player.

;//identifier **is the number that is associated with this item. No other item should have the same identifier as another item. Make sure when you copy and paste entire entries that the first thing you change is the identifier.**

;//item class **is the class this item belongs to. The only real use for this is to exclude use of this item from being used by a particular player race. Other than that it serves no higher purpose.**

;//weight **is the virtual weight of the item. You do not have to use weight in your mod. If you aren't going to use weight than simply make all items 0 weight. Calculating weight for your mod is more about how much you want them to carry of a particular item and of any item, than about being realistic usually. Since most players don't really care for this you'll probably want to stay on the lower end unless your mod stress management over action. If you've got alot of heavy items that'll force players to constantly manage their inventories and it can get tiring.**

;//short description **is the description that is shown in the inventory. You'll want to keep this brief, yet informative.**

;//first pickup text **is the text that is shown in a journal type entry the first, and only the first time you get the item. The amount of text you can use here is much greater so this is where you'll want to get detailed about describing the item and perhaps several of its more obscure uses. Be mindful that most players are going to right click through this out of habit. So if there is truely vital information you're actually better off giving it in a more interactive and interesting manner.**

;//size on map **determines how large the graphic for the item will be when dropped.**

;//show on radar **tells the game if the item should show on the radar when it is dropped. 0 for no, to never show it on radar, good for things that want to be found and not detected. You'll use 1 for yes if you do want the item being detectable. Lastly you can choose 2 to have the item shown even without a detector.**

;//particle to shown on radar **is the particle that will be used for this item on the radar. You can use unique particles to make certain items easier to distinguish from others.**

;//visible in inventory **is typically only used if you're going to have an item hidden. Generally you'll set this to 1. On the other hand if you all want to is have something available for use with a keypress than you can set this to 0 and you'll only be able to access it with a key and naturally it wont be droppable either. Setting it it 0 is perfect as well to help setup threading a bunch of effects to a key for a variety of purposes. This is obviously a more advanced feature and you probably don't want to mess with it if you don't have a good grasp of how it is supposed to work.**

;//texture **is the graphic that'll be used for your item in and out of the inventory. Per usual the graphic should be in the /textures folder or in a subfolder within it of the same exact name as your mod.**

;//wielded script **is what script is automatically run when the item is being wielded. If you're not making it a wieldable item than you should just make this -1. Otherwise you'll pick the number associated with the script you want executed.**

;//wielded item disabling script **will check a script of your choosing. If the script is false than the item will be removed from being wielded. A good example for this might be a script that checks to see if you have a certain item which is used as ammunition for a gun. When you don't it'll automatically remove the weapon from use. It's very important to note that only the conditions of the script you select will be checked and the effects will not be run, but the conditions will run on the set interval set by the script you choose.**

;//show help text for conditions **tells the game if it'll give you hints as to what you need to do or have to use the item successfully. Mind you that this goes for all the conditions that the item uses since it is not included in a block, it is universal for the item itself. I actually suggest setting this to 0 by default as the pointers the game will the player are more annoying than helpful. This is just my personal opinion, but it seemed nicer before the game gave you the hints.**

Well that covers the Items.dat file. You'll probably have a problem here and there, but this is definately the file you want to get the most practice and get really good at dealing with.

Journal.dat A9

```
Stowaway's personal log. Stardate;//journal entry start text
91379;//start date in days (Calendar dates start from 1-1-1900)
0;//date type, 0=stardate, 1=calendar
Day 1 Journal.

Day 3 Journal.

Day 7 Journal.
end_of_file;
```

Journals are done as a data file. A journal can actually be called anything you like since you get to pick the file each race uses for their journal in the Player_races.dat file. You'll also probably have several journals for your different player races. The entries for a journal end when you start a new line in the file by pressing enter. You can skip days in the journal by skipping a line as you can see denoted in the example.

So now for a simple break down of the three lines that come before all your journal entries do.

;//journal entry start text **is the text that will show at the top of the race's journal.**

;//start date in days **is exactly what it sounds like. This is the starting date of the race's journal.**

;//date type **is the type of date your journal will be using. For futuristic you'll want to use stardate, otherwise you'd use calender date for modern or past eras.**

That's it with journals as they are fairly easy to deal with. You'll want to check your journal entries to make sure they fit into the journal box on a semi-regular basis. You may just have to edit them to make them fit.

Light.dat A10

```
flashlight;//name-------
  0;//identifier
  light0.png;//texture
  0;//type, 0=flashlight, 1=omni
  0;//pulse speed
  0.8;//red color component
  0.8;//green color component
  1;//blue color component
  0.35;//alpha color component
  2.0;//light intensity
  -1;//flash particle (-1=nothing)
    0;//flashing speed
    0;//particle life time
end_of_file;
```

The lighting file is one of the easier files you'll deal with. This file does not use blocks and every entry has the same amount of lines. The file will always start with *;//name* **as will each entry. The entries will end always with** *;//particle life time*.

Since there are no blocks except for the entry itself I'll get right to the breakdown.

;//name **is the name your light will go. This is never shown in the game but it is wise to name it something distinguishable regardless, for your own reference.**

;//identifier **is the number that will be associated with the light. It's important as with other indentifiers within the same file that no two identifier be the same number.**

;//texture **is the graphic that will be used for your light. The graphic must be in the /textures folder or a subfolder within with the same exact name as your mod's folder.**

;//type **determines what kind of light it will be. Your two choices are 0 for flashlight or point light and 1 for omni or omni-directional light.**

;//pulse speed **is the speed at the which the light changes from its brightest to dimmest.**

;//red color component
;//green color component
;//blue color component **These three lines determine the blend of colors that make up the light. The max a color channel can be is 1.0 and the lowest is 0.**

;//light intensity **is the actual range of the light. The more intense a light the better it will light an area.**

;//flash particle **is the particle that will be used from the Particles.dat file. This particle will be randomly sparked from the light.**

;//flashing speed **is how frequently the particles will be sparked from the light.**

;//particle life time **is the amount of time each particle sparked will last.**

That covers the Lights.dat file. You'll probably mess with this file quite a bit until you get the desired results.

Map.dat A11

```
5;//width of map below
5;//height of map below
-3;-3;-3;-3;-3
-3;-3;-2;-3;-3
-3;-2; 0;-2;-3
-3;-3;-2;-3; 3
-3;-3;-3;-3;-3
end_of_file;
```

This a very basic Map.dat file. Unlike other files there are no entries. This file is simply composed of how you want your areas to be laid out. To place an area specifically it needs to have a unique and solitary area class. If you want your start area centered generally your map needs to be an odd number height and width. As you can see the file is arranged line by line with each area separated by a ;. It is sometimes neccesary as seen with 0 to space it out, it has no actual effect but keeps the lines from becoming skewed. When you pick a number you're actually picking an Area Class as opposed to 1.2's version where the number was the area's number. If you definately want an area to be placed at a position in the map than you'll use -2. When you want an area to be placed only if there is an adjacent area than you'll use -3. To restrict an area from being placed at a tile you'll use -1. You can place your start area anywhere you like but usually you want it at the beginning of a single height map and the middle of a square or rectangle shaped map.

You're probably not going to get how this file works until you played with it a bit. Here are the only two lines you'll deal with besides the map itself.

;//width of map below **is the width in area's the map will be.**

;//height of map below **is the height in area's the map will be.**

That should make you fairly familiar with the Map.dat file.

Music.dat A12

```
0;//identifier
  ingame2.mp3
```

```
  1;//can be randomly selected
end_of_file;
```

This file is extremely basic in its structuring. There are no blocks only the 3 line entries themselves. Each entry begins with *;//identifier* **and ends with** *;//can be selected randomly***.**

Now I'll break down the lines for you.

;//identifier **is the number that will be associated with this music file. This should be unique from any other music entry.**

[mp3's name] is simply a line that has the name of the mp3 that is going to be used on it.

;//can be randomly selected **simply determines if the music will be played as a random selection during play. So if you wish the music to play in the normal loop of music you would put 1. However if the music is meant to be used only at a specific time than you should use 0.**

That covers this file and you should have little trouble figuring it out even on your own.

Object_Definitions.dat A13

```
prop a;//name--------------------------------------
  1;//identifier
  begin_animation_frames;
    prop.png;//texture name
    0.000000;//time to display frame
  end_animation_frames;
  0.450000;//size*128
  0.000000;//vary size
  1;//transparent
  0;//layer
  0;//does it sway with the wind
  0;//gets transparent when player near
  0;//provides shade for player
  0;//collision detection type, -1=none, 0=circle radius parameter0, 1=polygon
parameter0 from polygons.dat size parameter1
  0.700000;//collision detection parameter0
  0.000000;//collision detection parameter1
  0;//Block vision. If 1, enemies and player cannot see other creatures
through it. Use sparingly, this slows the AI down.
  0;//Stops bullets. Use sparingly as this can slow the game down a lot.
end_of_file;
```

I purposely started with the second entry as the first must always be the null object that is used by the Map Editor to prevent random placement near it. This file is fairly simple, but it is still important to understand what everything does as this file is used for props and plot objects, both of which will likely fill your mods. So you'll definately want to make sure these are done right.

The first and only block you'll deal with besides the entry itself is the Animation Frame Block. The entry for the Animation Frame Block is always placed between *begin_animation_frames;* **and** *end_animation_frames;***. Each entry is two lines long and is composed of the** *;//texture name* **line which tells it what graphic to use from the /textures**

folder or the subfolder within of the same exact name as the mod, and the *;//time to display frame* **line which tells it how long to show the specific graphic in milliseconds. One thing of note is that animations automatically unsync. Despite the last line it's well enough to note that any dropped plot objects within the game will sync and have their animation start at frame 0. So if you want to have several plot_objects synched you should create a spawner type plot object that will drop the plot object that has the animation you want synched.**

Now for a breakdown of the individual lines in this very important file.

;//name **is the very first line in an entry and tells the game what name to associate with the prop. This name will not be used when the definition is used for a plot object, instead the plot object's name will be used. The only place this name is important is for your own reference and of course it will be shown in the map editor. I cannot stress enough the importance of giving unique names, especially if you're going to use the same graphic for several props or plot objects with only minor differences in them.**

;//identifier **is the number that will be associated with the object. This number should be unique and no other object should share the same number.**

*;//size*128* **is the scaling of the graphic if it were a 128x128 sized graphic. You don't want to go too big on this. I believe the max is 5, after that it will cause problems. All increasing an actual image's size beyond 128x128 will do is increase the visual acuity of the graphic while it is scaled having it lose less visual depth. The graphic will always be scaled to a 128x128 sized graphic despite what it's pixel height and width is, so a value of 5 will make the graphic 640x640 pixels in the game despite what size the graphic was. Using larger sized graphics uses much more memory so I suggest breaking up larger images into pieces and assembling them with the Map Editor.**

;//vary size **determines if and how much random scaling will be done to each individual plot object or prop that uses this object. If you want no random scaling than you simply set this to 0. The highest this will probably ever be useful for you is a scaling of 2. This is a great way to give your mod a more organic look so not everything is the same exact size.**

;//transparent **tells the game if this object will be using the alpha channel within the graphic. If a graphic has no alpha channel for whatever reason, perhaps because it uses up the entire size of the image than you can set this to 0 and it'll save some memory. By default this should always be set to 1 unless as I stated, the graphic fills every pixel of the image and does not have any transparency. In short this disables alpha masking.**

;//layer **determines what graphical layer this object defintion will be used on. There are three layers the first being 0 which is what most objects belong to. Layer 0 is below the player and most other creatures. The next layer is 1 which is on the same layer as the player and most other creatures. The final layer is 2 which is above the player and most other creatures.**

;//does it sway with the wind **is more or less self explanatory, as it simply determines if the object will shift from its original position like a tree swaying in the wind. If your object is meant to be static than you'll want to set this to 0. For things that are rooted by are**

assumed to be able to shift around on their base like flowers, plants, trees, bushes, ect. than you should probably use 1 to make the environment seem more lively and less static.

;//gets transparent when player near **tells the game if the object should become see through when the player is near it. This is perfect for objects that are placed above the player like a tree canopy or roof of a building, which allows you to still see what is underneath. So if you want to have some kind of canopy you'll generally want to set this to 1 so the player can still see where they're going. If the player is passing beneath the object or otherwise does not need the object to become transparent than you'll use 0.**

;//provides shade for the player **is generally just used for the temperature bar. You've only got two options, 0 for no or 1 for yes. If you're in the shade you cool down. However there is a condition that does check if you are in the shade and allows you to assign a multitude of effects whenever you're in the shade. So this becomes a fairly important feature to play with, even if you don't intend to have temperature you can use shade type objects as a class type that sets off specific effects. In other words this should allow you a bit of creative freedom.**

;//collision detection type **is the collision detection if any that will be used by this object. You've got 3 options the first being no collision detection by using -1. Your second choice is the original and quickest collision, circular collision detection using 0. The third and most complex is 1 and will be used for objects that require a specific polygon shape for collision detection, like a square or rectangle(This is box collision detection so many people asked for). If you chose 1 than you'll also need to pick which polygon to use.**

;//collision detection parameter0
;//collision detection parameter1 **These two lines do different things depending on which collision detection type you chose. If you chose -1 for no collision detection than you should make both lines 0. If you chose 0 for circular collision detection than parameter0 is all you'll use which tells it how large the radius should be, the default being 1.0 but you'll probably have to tool around with it to give it the fit you want. If you chose 1 for polygon collision detection the first line tells it what polygon to use from the Polygons.dat file and the second line tells it how big the polygon should be scaled, again the default size is 1.0.**

;//Block vision **determines if the object will block the vision of the player as well as creatures. As stated in the file itself this will slow down the game to a degree so it should be used sparingly. This is a nice feature for mods which have some stealth element to them but can be used for other purposes as well. This feature is only enabled if you chose polygonal as collision type not circular collision detection.**

;//Stops bullets **does exactly what it sounds like. It stops bullets from going through the object. This feature is implemented, yet I cannot say it is 100 percent effective. So if you use this feature don't expect it to be perfect, it wont be. Like the line above it also slows down the game to some degree, so use this only when you need it. This feature is only enabled if you chose polygonal as collision type not circular collision detection.**

Like other files I've covered you should now have a good understanding of how the file works.

Particles.dat A14

```
Particle A;//name
  0;//identifier
  particle.png;//texture
  4;//size
  1;//red color component
  0;//green color component
  1;//blue color component
  1;//alpha component
  1;//shrink size
  0;//can be rotated (faster if not)
  -1;//change into another particle when hits ground
  0;//blending type (0=normal, 1=light)
end_of_file;
```

This file, much like the Lights.dat file serves to create a greater graphical appeal. Particles are used throughout the game for various things like rain, footsteps, radar dots. While all that is true there is a chance you will not mess with particles at all or simply use the default ones provided with the main game. Either way you should have a good understanding of how particles work so you can use them to spruce up your mod. Each entry begins with *;//name* **and ends with** *;//blending type***.**

There are no blocks in this file so I will simply get right to the line by line breakdown.

;//name **is the name the particle will go by but it is not referenced, not even by the Map Editor. You should name this for your own reference, it is never seen in the game.**

;//identifier **is the number that will be associated with this particle. Per usual you'll want this number unique amongst all other particles within the file.**

;//size **is the pixel size of the particle within the game.**

;//red color component
;//green color component
;//blue color component **These three lines control the color tinting of the particle. The first line controls the red component, while second is green and third is blue. You mix the colors by setting their value from 0 to 1, 1 being 100 percent of that color channel being used.**

;//alpha component **is how see through the particle will be. The maximum is 1 for normal alpha masking used by the particle and 0 is a completely invisible particle. You'll never use 0, rather you never should since it negates the point of using the particle in the first place. A setting of 0.5 is half transparency.**

;//shrink soze **determines if the particle will shrink or grow over time. This basically just makes it shrink till it vanishes or grow to its normal size from nothing. To make a particle shrink to nothing you'll use 1. To make a particle grow from nothing to its normal size set by** *;//size***, you'll set this to -1. If you don't want the size of the particle changing than you'll simply set this to 0. In addition the shrink size whether negative or positive can be a decimal which will control the speed of the shrinking or growing.**

;//can be rotated **tells the game if the particle can be rotated or not naturally. For more organic type particles like fire, electricity, ect. you'll probably want to toggle this on with 1 to get a less static appeal from the particle. For non-organic and stiff particles you may just want to save some memory and set this to 0 to turn it off.**

;//change into another particle when hits the ground **does exactly what it sounds like. It lets you choose which particle if any to change the particle into after it fades away itself. If you don't want it changing into another particle you simply put -1 for none. On the other hand you'll want to put the number of the particle you want it to change into if you do want to use this modifier. The most base use of this particle is to create either a particle trail of sorts, or a splatter type effect when it changes into a much large particle that puddles on the ground than fades.**

;//blending type **is the type of graphical blending that will be done with the particle. Type 0 or normal will blend it as an overlay and will not be tinted by other graphics. The second type, which is 1 or light, will be tinted by other graphics.**

That covers the particles. This file is quite a bit of fun to mess with and I'm sure you'll be using it just add that extra bit of polish to your mod.

Player_races.dat A15

```
Best for beginners.;//easy difficulty level description
Medium difficulty.;//medium difficulty level description
A good challenge.;//Hard difficulty level description
Player A;//name-----------------------------
0;//identifier
1;//visible in start menu
0;//side
0;//start area
journal.dat;//journal text file name
Your first choice as a race.;//description in start menu
1;//creature number in creatures.dat
0;//start animation number in animation.dat
200;//day length
100;//maximum carry weight
0.0;//climatic temperature change multiplier
player.png;//rag doll texture
computer.png;//interface texture
begin_weapon_frames;//must contain all weapon classes in order starting from 0
  0;//weapon class number
    1;//can use the weapon class (0=cannot, 1=can)
    creature.png;texture name
    2;//row that contains the 4 weapon frames
    1.000000;//weapon_x
    27.000000;//weapon_y
end_weapon_frames;
begin_specialties;
  health bar;//description
  5;//specialty
    0;//difficulty level
    none;//message
    0.000000;//parameter0
    0.000000;//parameter1
    100.000000;//parameter2
```

```
    100.000000;//parameter3
end_specialties;
begin_disabled_endings;
  1;
end_disabled_endings;
begin_disabled_item_classes;
  1;
  You cannot use this item!;//message
end_disabled_item_classes;
end_of_file;
```

This file can easily become overwhelming, especially when you multiple weapon classes and bar types being used. This is the main file you'll use when editing any of the playable races within the game. As you may have noticed with the example there are many blocks used within the file. In actuality the whole entry is almost entirely made up of blocks. The beginning of each entry starts with *;//name* **and ends with** *end_disabled_item_classes;***. Before entries come three lines, the difficulty level description lines. Each line is self explanatory and is where you'll put what text will be displayed for each corresponding difficulty in your mod.**

The first block you'll encounter in the entry is the Weapon Frames Block. This block not only determines what weapon classes the race can and cannot use, but it also determines what file and line the graphic should be used from. As noted in the file itself, it is not optional, you must list every weapon class used in the game. Each entry is placed between *begin_weapon_frames;* **and** *end_weapon_frames;***. Each entry contains six lines beginning with** *;//weapon class number* **and ending with** *;//weapon_y***. The first line** *;//weapon class number* **determines what weapon class the entry is referring to. The second line** *;//can use the weapon class* **is a toggle type line where 0 equals no and 1 equals yes, this is how you disable a weapon class, by picking 0. The third line is** *;//texture name***, it determines what graphic to take the 4 animation cells for the attack animation for the weapon class from. The fourth line is** *;//row that contains the 4 weapon frames* **and is used to tell it what row to use, 0 being the first row and 3 being the last. The fifth line is** *;//weapon_x* **and tells the game the x coordinate offset of the weapon from the player race's center. The sixth and final line is** *;//weapon_y* **and tells the game the y coordinate offset of the weapon from the player race's center.**

The second block you'll see in the entry is the Specialties Block. The Specialties Block does a multitude of things like giving a race a bar and setting the rate it drains at, what items the race starts with and even which ones it cannot drop. Each entry lies between the lines *begin_specialties;* **and** *end_specialties;***. Each Specialty Block's entry is 8 lines long starting with** *;//description* **and ending with** *;//parameter3***. The** *;//description* **line in the entry is actually not used by the game but should be used by you for reference. The** *;//specialty* **line is very important, as it chooses what specialty the player race will be using (Your choices being 0 through 7, excluding 1 which has no specialty attached to it). The third line** *;//difficulty level* **determines at what difficulty levels the specialty will be used at, they're listed at the bottom of the file (Your choices are -2 to 2). The fourth line in the entry will be** *;//message* **and it will tell the game what line of text to display when the specialty is in use. The final four lines,** *;//parameter0* **through** *;//parameter3* **are variable depending on what** *;//specialty* **you chose to use and will finalize your entry.**

The third block you'll encounter is the Disabled Endings Block and is not actually used anymore, it is not read by the game at all. The block begins with *begin_disabled_endings;* **and ends with** *end_disabled_endings;***. Since endings are no longer controlled through the ending.dat file, since it doesn't exist, this block no longer has a use. So simply keep this block empty with only the header and footer line and you should be fine.**

The final block you'll come across is the Disabled Item Classes Block. This block tells the game what item classes the race cannot use. Each entry is two lines and are placed between *begin_disabled_item_classes;* **and** *end_disabled_item_classes;***. The first line of the entry is the number of the item class you wish to exclude followed by ;. The second and final line of the entry is** *;//message* **and will display whatever text you wish on the screen when the player tries to use the excluded item class.**

I'm going to start at the beginning of the actual file since there are three lines which proceed any entries.

;//easy difficulty **is the text that is shown in the menu for your mod when the player chooses Easy difficulty. Typically you want to be as brief as possible while covering all the major differences.**

;//normal difficulty **works the same way as the line above except it is shown when Normal difficulty. Once again list the things that are exclusive to this difficulty in as brief a manner as possible.**

;//hard difficulty **is the final line before any actual entries begin and it works the same as the above lines, except it is shown when Hard difficulty is chosen. Be brief, but still make sure you cover the main differences between this difficulty and the others.**

Now for a breakdown of the lines in an entry that are not blocks.

;//name **is the name that is associated with this race. This is what is shown in the menu when choosing a race.**

;//identifier **is the number that will be associated with this entry. This number should not be the same as any other and it should be one greater than the entry before it and one less than the entry after it. It is critical that the identifiers within this file be in proper order.**

;//visible in start menu **lets you choose if you want it to be a selectable race from the menu or not. By default this will be set to 1, making it selectable, however if you wish the race to be obtained within the game only than make sure to set this to 0.**

;//side **determines what side the player race will be on. This is fairly crucial in how other creatures will react to you. This is how you setup a race's unique side if you wish. If all the races are good/bad, ect. than you should simply make them all the same side for simplicities sake.**

;//start area **is what area the player race will spawn. This allows you the freedom to make each race spawn in a different area if you so wish. Make sure the area actually exists.**

Generally speaking you don't want to throw them into an area that has a high difficulty of survival, so they can at least get used to the controls before they die.

;//journal text file name **is the file that will be used for the race's journal. As I mentioned in the journal breakdown there is no definite name for a journal file. However if a file is a journal it must follow the outline stated in the breakdown. You can give all the races the same journal or their own unique journal.**

;//description in start menu **is the information the player will be given when they select this race from the menu. The amount of text you can use is fairly limited, so like the difficulty text, be brief, yet try to cover the main points of interest.**

;//creature number **is the number of the identifier from the Creatures.dat file that this race will be using.**

;//start animation **is the animation from the Animations.dat file that will be shown upon selecting the race and beginning the game with it. This is basically your intro for the race.**

;//day length **is simply how often you will get a journal entry. The day length will also affect the day/night sounds of the area the race is in along with temperature cycle.**

;//maximum carry weight **is the amount of combined weight the race can carry before being slowed down and eventually stopped completely. It's easiest to just use 100 as your average weight being carried and than calculate all your item weights from that. This way if you want a race to be weaker or simply have less carrying space you make it less than 100 to hamper them. You can however set this up however you like and are encouraged to.**

;//climatic temperature change modifier **is the amount the temperature is multiplied by for the race. This only controls things that directly effect the temperature bar. If you're not going to use a temperature bar you can simply set this to 0. A positive value will obviously increase the effects while a negative will reduce them.**

;//rag doll texture **is the graphic that will be used for the race's representation both in the inventory and in the menu selection for the race.**

;//interface texture **is the graphic that is used for the radar in your mod. You can use the same graphic or a unique one for each race.**

Well that should give you at least a fair idea of how the Player_races.dat file works.

Plot_objects.dat A16

```
begin_plot_object_classes;
  object class a;//0
  object class b;//1
end_plot_object_classes;
Object A;//name-------------------------------------
  0;//identifier
  -1;//map type to place to -1=any map 0=> type from areas.dat
  0;//plot_object class
```

```
  0;//object definition number
  1;//number of objects to be placed
  0;//location type, 0=random, 1=distance min(parameter1) max(parameter2) from
player start, 2=distance min(parameter1) max(parameter2) from object
parameter0, 3=coordinates [parameter0, parameter1] (0=min, 1=max)  vary
location=parameter2
  0.000000;//location parameter0
  0.000000;//location parameter1
  0.000000;//location parameter2
  1;//show on radar 0=does not show, 1=show on scanner, 2=show even without
scanner
  20;//particle to shown on radar
  none;//radiate sound
  -1.000000;//time to delete object in milliseconds, -1=don't delete
  3;//trigger event by 0=player click, 1=player near, 2=near or click, 3=map
creation (try to avoid any creature specific effects when using this, also
does not check conditions), 4=interval parameter1
  0;//trigger event parameter1
  1;//show help text for conditions
  begin_effects_block;
  none;//event text
    pick_up.wav;//sound, none for nothing
    1;//vanishes after event, 0=no, but cannot be reused, 1=vanishes, 2=no,
and can be reused
    begin_conditions;
    8;//condition
      100.000000;//condition parameter0
      0.000000;//condition parameter1
    end_conditions;
    begin_effects;
      0;//effect number
      0.000000;//parameter1
      0.000000;//parameter2
      0.000000;//parameter3
      0.000000;//parameter4
    end_effects;
  end_effects_block;
end_of_file;
```

The Plot_objects.dat file is one of the most important files you'll deal with. Plot objects are the main you do things within an area. Each entry will always begin with *;//name* **and end with** *end_effects_block;***. There are only three blocks used within this file, the first not even being in an entry. Even with its blocks you'll find the Plot_objects.dat file to be one of the more comfortable files to edit as it rarely stacks up the way other files can. It's a good thing to remember that plot objects can spawn various effects from their locations so they could easily be used to add dramatic flare to mod by careful placement using the Map Editor.**

The first block I'm going to cover proceeds any entries, the Plot Object Classes block. This block should be included despite if you are going to use Plot Object Classes or not. Each entry is just one line and should be between *begin_plot_object_classes;* **and** *end_plot_object_classes;***. Each line is numbered as shown in the example, starting with 0. The wording that actually comes before the ; is what the game will call a Plot Object Class within the game itself. Plot Object Classes are how you group together Plot Objects for being checked by a condition as well as be excluded by an Area.**

The first block for an entry is the Effects Block, not be confused with the Effects Block that is contained within it as well. Entries will lie between *begin_effects_block;* **and**

end_effects_block;. **Each entry has three lines followed by the Condition Block and Effect Block. An entry begins with** *;//event text* **and ends with** *end_effects;*. **The first line in the block is** *;//event text*, **this tells the game what text to show when the effects within the block are successfully completed. The second line is** *;//sound*, **which tells the game what wave file to play when the effects are successfully completed. The final line before the two proceeding blocks is** *;//vanishes after event*, **this line determines if the Plot Object will be destroyed or not, using 0 does not destroy it, but the object cannot be reused, using 1 makes the object vanish and using 2 makes the object not vanish and reusable.**

The second block is the Conditions Block. This block works exactly the same way as it did for the Items.dat file. Each entry will go between *begin_conditions;* **and** *end_conditions;*. **Each entry is 3 lines long, beginning with** *;//condition* **and ending with** *;//parameter1*. **You can have as many entries as you wish, but remember it checks them all, not just one condition, so make sure you use the proper combination for what you're trying to check. Each condition will be covered in detail later in this FAQ.**

The third and final block for an entry is the Effects Block. This block like the conditions works exactly the same way as it did in the Items.dat file. Each entry will go between *begin_effects_block;* **and** *end_effects_block;*. **Each entry is 5 lines long, beginning with** *;//effect number* **and ending with** *;//parameter4*. **You can stack effects just like you can conditions, each one listed will be executed. Each effect will covered in detail later in this FAQ.**

Now for a breakdown of each line that isn't in a block. Some lines will be grouped together.

;//name **is the name that will show in the game when the plot object is highlighted. It's important to note that plot objects set to be set off by player is near will not show their name.**

;//identifier **is the number that will be associated with the plot object. This number should be unique and not used by any other plot objects. Make sure when you copy a plot object by hand that you change the identifier right away so you don't forget.**

;//map type to place to **will tell it what area it can be placed in. If set to -1 than this plot object will be placed in any area with the exceptions of those that exclude it by plot object class in the Areas.dat file.**

;//plot object class **is the group that this plot object will be in. The plot object classes themselves are defined at the top of the file before any entries.**

;//object definition number **is the object definition that this plot object will use. Make sure you've already got your object definition made before you make a plot object, as each plot object needs a object definition. The object definition will define the size, looks and other attributes of the graphic itself. This can also be set to 0 for objects that don't require a graphic for whatever reason.**

;//number of objects to be placed **sets the amount of this plot object to be distributed among the areas that can have it within them. Plot objects that are placed specifically through the**

Map Editor do not count against this number. Generally speaking larger areas will get a larger percentage of the number you choose than smaller ones will.

;//location type **is how the plot object will be placed. For random placement you'll use 0. To place an object from the start position of the player, parameter1 is used for the minimum distance while parameter2 is used for the max distance. To place the plot object a distance from another object than you'd use 2, parameter1 for the minimum and parameter2 for the minimum(Works best if there aren't more than one of the plot object you're placing it from.), and parameter0 for the plot object it is being placed from. To place a plot object precisely use 3, parameter0 is the x coordinate, you can use 0 to 1 (0 being the far left and 1 being the far right), parameter1 is the y coordinate you can use 0 to 1 (0 being the very bottom and 1 being the very top), parameter2 is how much to vary the location by. I really don't suggest using option 3 anymore, it's a relic so to speak, instead save yourself the trouble and use the Map Editor to place a plot object precisely where you want.**

;//location parameter0
;//location parameter1
;//location parameter2 **Reference the above break down to see the use of these parameters. They vary depending upon which option you chose for** *;//location type***.**

;//shows on radar **determines if the plot object will show on the radar or not. If you don't want it being shown by the scanner than you'll set this to 0. If you do want it showing with the scanner than you'll use 1. If you want it to show even if they don't have a scanner than you'll use 2.**

;//particle to show on radar **tells the game what particle to display if one is shown from the Particles.dat file. This is a newer feature as you may notice and it will allow you to finally give plot objects a unique radar signature, so put it to good use.**

;//radiate sound **is what wave file will be played when the player is near the plot object. You can create some ambient noises with this, just be careful you don't overuse it.**

;//time to delete object in milliseconds **is the amount of time before the plot object is deleted. To not have a plot object deleted simply set this to -1, which is what it should be by default. This doesn't seem to work with** *;//trigger event by* **options 3 or 4.**

;//trigger event by **will determine just what sets off the effects you have chosen for the plot object to execute. If you set it to 0 than the player must right click the object. If set to 1 than the player must be near the object. If set to 2 than the player can be near or click, basically whichever is done first, they don't need to do both. Setting this to 3 makes it a map creation plot object, this is unique in that it will ignore any conditions and automatically execute the effects once. Option 3 can be used for a variety of things, the main being to place an item. To make the effects of the plot object automatically trigger at set intervals in milliseconds you'd use 4. I've found option 4 to be quite messy, however it is the only way I've found to spawn something with a condition since 3 will ignore conditions. [That means that if you want something appearing only on a certain difficulty or only if they pick a certain race you'll use 4, you'll need to set the interval to 100 to make sure it only does the effects once, otherwise chances are it will do them several times. (The above**

mentioned about using option 4 to create a conditionally spawned item or object should be considered a tweak or exploit as it is not readily supported otherwise.)] ***If *;//trigger event by* **is set to 3 it is very important to not use any effects that refer to creatures since they haven't been placed yet and it can cause serious problems including crashing the game. Any effect or condition with creature or player in their description should be avoided.*****

;//trigger event parameter1 **is only used with option 4 for** *;//trigger event by***, otherwise it should be set to 0.**

;//show help text for conditions **simply tells the game whether or not it should display a hint on why the conditions are not satisfied within the game. Personally I've never liked the hints the game gave. So I suggest setting this to 0 for no, to allow the player to find out through hints you provide. However if you do want the game giving them the hint automatically than set it to 1.**

You should now be fairly well versed in how the Plot_objects.dat file works.

Polygons.dat A17

```
rectangle;//name-------------------------------------
  1;//identifier
  start_points;
    -0.5;//x
    -0.5;//y
    0.5;//x
    -0.5;//y
    0.5;//x
    0.5;//y
    -0.5;//x
    0.5;//y
    -0.5;//x
    -0.5;//y
  end_points;
end_of_file;
```

This file is unique in that its only purpose is to set up collision detection polygons to be used in the Object_definitions.dat file. You should at least copy over the rectangle polygon since it will allow you to box type collision detection along with the circular collision detection type. Remember that collision polygons take processing time and are collision detection can be said to be imperfect at best. So if your mod will depend heavily on collision detection you might want to rethink it and do it another way, since you will not get perfect results no matter how much you tweak it. Each entry begins with *;//name* **and ends with** *end_points;***.**

The only block you'll deal with in this file is the Points Block. The single purpose of this block is to define the shape of the polygon point by point. Each entry should lie between *begin_points;* **and** *end_points;***. Each entry is only 2 lines and defines a single point in the polygon. The two lines are** *;//x* **for the x coordinate of the polygon and** *;//y* **for the y coordinate of the polygon. The minimum value for a polygon's point is 0 and the maximum is 1. Another rule of thumb is however many sides your polygon will have you should have 1 more entry. A rectangle for example will have 5 entries as seen above in the example.**

There are only two lines that are not part of a block.

;//name **is the name that will be associated with the polygon. This is referenced by nothing, not even the Map Editor. This is purely for your own reference and since all each entry is, is a bunch of coordinates it would be wise to make sure you give each a unique and memorable name.**

;//identifier **is the number that will be associated with the polygon. This number should be unique to all other polygon's identifiers. The only file that will reference this number will be the Object_definitions.dat file.**

While you certainly don't need to use anything beyond the default polygons that come with Notrium this information should provide useful when you decide to make some custom objects that require custom collision polygons.

Scripts.dat A18

```
Play Animation;//name
  0;//identifier
  1;//run without calling (set to 0 if you use the script for example as a
wield script)
  0;//call position, 0=player location, 1=mouse location
  0;//calling creature 0=player, 1=creature nearest to the mouse, 2=player
controlled creature
  200;//run every n milliseconds
  none;//message
    0;//message type, 0=quick message, 1=journal
  none;//sound, none for nothing
  0;//delay script for n seconds after conditions met
  1;//disabled after first use, 0=no, 1=yes
  begin_conditions;
    35;//condition
      1.000000;//condition parameter0
      50.000000;//condition parameter1
  end_conditions;
  begin_effects;
    25;//effect number
      1.000000;//parameter1
      0.000000;//parameter2
      0.000000;//parameter3
      0.000000;//parameter4
  end_effects;
```

This Scripts.dat file will be the main way you deal with complex events. There are only two blocks in this file and they work the same as they do in the Items.dat file and the Plot Objects.dat file. Each entry for the file will begin with *;//name* **and end with** *end_effects;***. You can do such things as ending the game because health has run out or giving an item once specific conditions have been fullfilled. You'll want to be fairly familiar with this file if you plan on doing anything fancy.**

The first block is the Conditions Block. This block works exactly the same way as it did for the Items.dat file. Each entry will go between *begin_conditions;* **and** *end_conditions;***. Each entry is 3 lines long, beginning with** *;//condition* **and ending with** *;//parameter1***. You can have as many entries as you wish, but remember it checks them all, not just one condition,**

so make sure you use the proper combination for what you're trying to check. Each condition will be covered in detail later in this FAQ.

The second block for an entry is the Effects Block. This block like the conditions works exactly the same way as it did in the Items.dat file. Each entry will go between *begin_effects_block;* **and** *end_effects_block;***. Each entry is 5 lines long, beginning with** *;//effect number* **and ending with** *;//parameter4***. You can stack effects just like you can conditions, each one listed will be executed. Each effect will covered in detail later in this FAQ.**

Now that blocks are out of the way I'll break down the non block lines.

;//name **is the name that will be associated with the script. This is only used for your own reference but it should be as descriptive as possible to help you remember exactly what each script does.**

;//identifier **is the number that will be associated with the script. No other script should have the same number.**

;//run without calling **simply determines if the script should be checked without being called by an effect. The default for this is 1 which makes the script checked as often as it can be executed. If you're going to use it for an item, like a wield script for example, than you'll want to set it to 0 so it only runs when it is called by the item being wielded or an effect.**

;//call position **is the point where something may spawn if it is created. If set to 0 than it will use the player's position. If set to 1 it will use the position of the mouse instead.**

;//calling creature **determines what creature has called the script. If set to 0 than the player is the one who calls the script. If set to 1 the creature nearest to the mouse has called the script. If set to 2 the player controlled creature has called the script.**

;//run every milliseconds **tells the game how often to check this script's conditions and execute the effects if they prove true. Generally a time of 300 is good for fair precision and a time of 1000 or greater is fine for scripts that don't need to be precise. You don't want a bunch of scripts running very frequently as it may slow the game down.**

;//message **is the message that is shown when the conditions prove true for the script and the effects are executed. The length this line can be varies depending on if you pick 0 or 1 for the following line. If you pick 0 for the following line it should be kept brief. If you pick 1 for the following line it can be as long as a journal entry. Type in none if you don't want any message showing.**

;//message type **tells the game how the text you chose for the previous line will be displayed. If you select 0 it will show at the upper left briefly than vanish. If you select 1 than it will pop up as a journal entry.**

;//sound **will be the wave file that is played when the script is executed. This sound will be run each time the script is run much like a use sound is played each time an item is used, so keep than in mind.**

;//delay script for n seconds after conditions met **is a time delay in seconds till the effects are executed. This particular option has very little accuracy to it so use it with caution, especially if you intend to use large amounts of time. This is however a great way to make sure this script isn't run at the same time as another. So if you've got several scripts running at the same time, this is the perfect way to make them run in the order you wish. To make scripts run in the order you wish simply have the scripts each use an increasing delay, the lowest delay obviously being the first to run, while the highest will be the last.**

;//disabled after first use **determines if this script will run only once or everytime until its conditions are no longer true. If set to 0 the script will run everytime it's conditions are met and remain enabled. If set to 1 the script will disable after it's conditions are met the first time. You can also enable or disable a script with an effect so a disabled or enabled script does not always have to remain that way.**

Simply because of the nature of the Scripts.dat file you will probably have to tweak your scripts several times before they work the way you want. So expect to check this file often if things don't work as you planned. Less experienced modders should likewise avoid using this file as much as possible until you're familiar with the effects and conditions.

Sides.dat A19

```
Side A;//name
  0;//identifier
  start_hostile_sides
    1;
    2;
  end_hostile_sides
end_of_file;
```

Now we come to the Sides.dat file. This file is basically how you will setup what does and does not like each other. Simply put if a side is not listed than it automatically likes that side and will not be violent towards it. However, it is important to realize that only creatures on the same side cannot hit each other, so even two friendly sides can indeed hit each other and possibly even kill each other, though they will not purposely try to do so. This file is fairly straight forward and only has one block, so you'll just want to make sure you don't chop off a line and you should have no problems. Each entry begins with *;//name* **and ends with** *end_hostile_sides;***.**

The first and only block you'll use in this file is the Hostile Sides Block. Each single line entry will define a side that the side is hostile towards, if it is not hostile towards a particular side you simply don't list it. Each entry should lie between *start_hostile_sides;* **and** *end_hostile_sides;***. Each entry composes of the number of the hostile side followed by a ;. Remember that even if creatures are friendly, if they're not on the same side, they can hurt each other.**

Now for a breakdown of the two lines that are not in a block.

;//name **is the name of the side. This is only used by the Map Editor and is not seen anywhere else. The side should be descriptive enough so that you can easily distinguish it from the others.**

;//identifier **is the number that will be associated with the side. As with any other identifier this should be unique and no other entry should have the same identifier number as another.**

It's a pretty simple file so you should little trouble dealing with the Sides.dat.

Sounds.dat A20

```
Sound;//name
  0;//identifier
  sounda.wav;//filename
end_of_file;
```

This is truely as basic as files will ever get. The entry is only 3 lines long and contains no blocks and no complexities. This file is simply to setup any cued sounds you want played within the game. Each entry will begin with *;//name* **and end with** *;//filename***.**

Now for a quick breakdown of this file.

;//name **is the name that will be associated with the sound. This is not referenced in the game or even in the Map Editor, so just put something that is easy to distinguish for you.**

;//identifier **is the number that will be associated with the sounds. This should be unique to any other sound's identifier.**

;//filename **is the file that will be played when the entry is cued. The sound should be in wave format and in the /sounds folder or the subfolder within of the same exact name as your mod.**

That covers the very simple Sounds.dat file. You need not ever use this file anyways, it is merely an optional way to play a sound.

Terrain_Maps.dat A21

```
Terrain A;//name-------------------------------------
  0;//identifier
  21;//width
  21;//height
  begin_map_grid;

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0
```

```
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0

0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0
  end_map_grid;
  begin_objects;
  end_objects;
```

This file is very complex. I completely and utterly suggest against messing with it beyond changing the *;//name* **entry to whatever you want. This file is extremely easy to mess up and that you'll have to delete the entry entirely and start over usually. Do not under any circumstances edit the map grid. The main problem with editing this file is that while you**

can edit the Objects Block entry you cannot be sure if you editted the proper one. So as much as this might annoy some of you I am not going to cover this file and skip past it. Asides from changing the name of a terrain map, everything you can do in this file can be done much easier within the Map Editor itself.

Terrain_Types.dat A22

```
Random Terrain;//name--------------------------------------
  0;//identifier
  0;//AI avoids this terrain
  0;//do not place on map edges on random selection
  0;//do not place random objects on this terrain
  0.000000;//base light r
  0.000000;//base light g
  0.000000;//base light b
  20000.000000;//footstep particle live time, -1=no footsteps
  -1;//override footstep particle
  none;//override footstep sound
  begin_terrain_frames;
    random.jpg;//terrain texture
    0.000000;//time to display frame
  end_terrain_frames;
  begin_effects_block;
  end_effects_block;
No random objects;//name-----------------------------------
  1;//identifier
  0;//AI avoids this terrain
  0;//do not place on map edges on random selection
  0;//do not place random objects on this terrain
  0.000000;//base light r
  0.000000;//base light g
  0.000000;//base light b
  20000.000000;//footstep particle live time, -1=no footsteps
  -1;//override footstep particle
  none;//override footstep sound
  begin_terrain_frames;
    norandom.png;//terrain texture
    0.000000;//time to display frame
  end_terrain_frames;
  begin_effects_block;
  end_effects_block;
Terrain A;//name-------------------------------------
  2;//identifier
  0;//AI avoids this terrain
  0;//do not place on map edges on random selection
  0;//do not place random objects on this terrain
  0.000000;//base light r
  0.000000;//base light g
  0.000000;//base light b
  20000.000000;//footstep particle live time, -1=no footsteps
  -1;//override footstep particle
  none;//override footstep sound
  begin_terrain_frames;
    terrain.jpg;//terrain texture
    0.000000;//time to display frame
  end_terrain_frames;
  begin_effects_block;
  end_effects_block;
end_of_file;
```

This file's name is very deceptive and the first time you see it you'll no doubt wonder what a terrain type is. Terrain types are simply put, floor tiles used in the game. They can either be mapped specifically to a certain spot using the Map Editor or can be used as a randomly placed tile in an Area. The first thing you'll want to note and is that the first two terrain types, Random Terrain and No Random Objects must be the first two entries. I've included those two entries so you can get a good idea of how the file would look with just one new entry in it. Each entry will begin with *;/name* **and end with** *end_effects_block;***.**

The first block you'll encounter is the Terrain Frames Block. This block will determine what frames make up the terrain's animation or still frame if that be the case. Each two line entry should be between *begin_terrain_frames;* **and** *end_terrain_frames;***.The first line of the entry is** *;//terrain texture***, it tells the game what texture to use (The texture should be 128x128). The second line of the block's entry is** *;//time to display frame***, is how long the particular frame texture will show in milliseconds. It's important to not that animations will automatically unsync, so if you want your animations to blend into each other you'll have to keep the more drastic changes towards the center and the minor changes at the boundaries of the texture, otherwise it will not look good when two of the same texture are placed beside one another.**

The second block you'll encounter is the 'main' Effects Block. Each entry in this block should contain within it the Condition Block and also, not confuse you, the Effects Block as well. The entries should be between *begin_effects_block;* **and** *end_effects_block;***. Unless you plan on having two different versions that depend solely on conditions than you should only have one entry. If the terrain is simply to serve as a graphic you don't need to include the inner blocks at all like the example above that has no Condition or Effects Block within the 'main' Effects Block.**

The first block within the 'main' Effects Block is the Conditions Block. This block works exactly the same way as it did for the Items.dat file. Each entry will go between *begin_conditions;* **and** *end_conditions;***. Each entry is 3 lines long, beginning with** *;//condition* **and ending with** *;//parameter1***. You can have as many entries as you wish, but remember it checks them all, not just one condition, so make sure you use the proper combination for what you're trying to check. Each condition will be covered in detail later in this FAQ.**

The second block in the 'main' Effects Block is the Effects Block. This block like the conditions works exactly the same way as it did in the Items.dat file. Each entry will go between *begin_effects_block;* **and** *end_effects_block;***. Each entry is 5 lines long, beginning with** *;//effect number* **and ending with** *;//parameter4***. You can stack effects just like you can conditions, each one listed will be executed. Each effect will covered in detail later in this FAQ.**

Now for a breakdown of the non-block lines.

;//name **is the name of the terrain. This is only seen in the Map Editor, despite that fact make sure it is descriptive enough. If your terrain will look the same as another and just have condition/effect variations than you definately want to make sure it has a unique description to avoid confusion of using the wrong terrain.**

;//identifier **is the number that will be associated with this terrain. Each terrain should have its own unique identifier.**

;//AI avoids this terrain **determines if the AI will try its best to not move onto the terrain. For terrains with harmful effects you might want to toggle this on with 1. If the terrain is normal or you simply don't want them avoiding it than set this to 0.**

;//do not place on map edges on random selection **tells the game whether or not to place the terrain on the edges of the map or not when it is randomly placed. If you don't mind where the terrain is placed you'll set this to 0. If however you wish to keep it from being placed at the edge, perhaps because it is harmful, than you'll set this to 1.**

;//do not place random objects on this terrain **keeps random objects from being placed on this terrain. To keep random objects from being placed on the terrain than you'll set this to 1, perhaps because it is harmful. If you don't mind random objects being placed on the terrain than you can set this to 0. It's important to note that this is simply one of those features I don't have the upmost confidence in, meaning there is no guarantee a random object went up on the terrain even with this set to 1.**

;//base light r
;//base light g
;//base light b **these three lines decide what light addition a terrain will have. Each line has a light addition of a specific color channel and they blend together. This will create an effect like glowing lava by making the base light red. The base light's color will also tint everything on the terrain's tile. If you're not going to use a light addition on a terrain simply set each of these three lines to 0.**

;//footstep particle live time **is how long a footstep particle for a creature will stay around on this terrain. This wont effect any other particles, just the footstep particle that you define for each creature in the Creatures.dat file. The time is in milliseconds, so for 1 second you'd put 1000. You can also nullify footsteps entirely on the terrain by setting it to -1.**

;//overide footstep particle **is the subtitute particle that will be used as a footstep on this particular terrain. If you're not going to override the particle than you simply set this to -1.**

;//overide footstep sound **is the sound that will replace the default footstep sound for the creature for this terrain. Perhaps the terrain is snow or some other surface that would normally change the sound of a footstep, well this is how you change the sound for it. Remember like any other wave file it must be in the /sounds folder or a subfolder within of the same exact name as the mod.**

Well there you go, now you should have a fairly good understanding of how the Terrain_types.dat file works.

Weapons.dat A23

```
0=Class A, 1=Class B, 2=Class C;//weapon class names
No Weapon - do not remove!;//name-------------------------------------
  0;//identifier
```

```
  0;//class
  bullet0.png;//bullet texture
  1;//bullets at one shot
  0;//stop at hit 0=stop where hit, 1=don't stop
  0.000000;//bullet size
  0.000000;//spread
  1.000000;//bullet speed
  1.000000;//time until bullet disappears
  1000000;//fire rate
  0;//trace 0=no trace, 1=blue beam, 2=bullet trace, 3=red beam, 4=green beam,
5=yellow beam
  -1;//hit visual effect colour, -1=nothing 0=?, 1=red, 2=green, 3=blue
  0;//hit visual effect particle, -1=nothing, 0=>particle from particles.dat
  none;//hit effect sound
  0.000000;//push target back
  0.000000;//push shooter back
  none;//fire sound
  none;//hit sound
  0.000000;//AI hear volume, the amount of anger the AI gets when it hears the
weapon
  begin_wield_conditions;
  end_wield_conditions;
  begin_fire_effects;
  end_fire_effects;
  begin_hit_effects;
  end_hit_effects;
end_of_file;
```

The Weapons.dat file is basically how you have any effect cued with the left mouse button. Obviously you use this file for actual weapons, but it isn't limited to that if you're creative. There are a few blocks, but its their function that makes them unique rather than what is in them. Each entry begins with *;//name* **and ends with** *end_hit_effects;***. Before any entries there is the** *;//weapon class names* **line which lets you define all the weapon classes there will be. Each weapon class should be separated by a comma like in the example above. Each individual weapon can be its own class if you like, just be wary that it will make for work for you in the Player_races.dat file.**

The first block is the Wield Conditions Block. This block is basically your standard Conditions Block except that it defines whether the weapon can shoot or not. So if the conditions for prove false then the weapon cannot fire. Each entry will go between *begin_wield_conditions;* **and** *end_wield_conditions;***. Each entry is 3 lines long, beginning with** *;//condition* **and ending with** *;//parameter1***. You can have as many entries as you wish, but remember it checks them all, not just one condition, so make sure you use the proper combination for what you're trying to check. Each condition will be covered in detail later in this FAQ.**

The second block is the Fire Effects Block. This block is basically your standard Effects Block except that if the wield conditions prove that whatever is in this block will happen to the creature firing the weapon. This is basically the block where you'll subtract energy, remove an item that acts as ammo or anything else you want happening when the weapon is firing, even muzzle flashes using a lighting effect. Each entry will go between *begin_fire_effects;* **and** *end_fire_effects;***. Each entry is 5 lines long, beginning with** *;//effect number* **and ending with** *;//parameter4***. You can stack effects just like you can conditions, each one listed will be executed. Each effect will covered in detail later in this FAQ.**

The third block in the Hit Effects Block is the Effects Block. This block is basically your standard Effects Block except that if the wield conditions prove that whatever is in this block will happen to the creature the weapon's bullets hit. This is the block where you'll put effects that will hurt the creature's that are going to be shot. Each entry will go between *begin_hit_effects;* **and** *end_hit_effects;***. Each entry is 5 lines long, beginning with** *;//effect number* **and ending with** *;//parameter4***. You can stack effects just like you can conditions, each one listed will be executed. Each effect will covered in detail later in this FAQ.**

Now that the blocks are covered I'll get into the individual lines that are not blocks.

;//name **is the name that will be associated with the weapon. This is what will show when the weapon is selected.**

;//identifier **is the number that will be associated with the weapon. No other weapon should use the same number for its identifier.**

;//class **is the weapon class that the weapon falls under. Weapon classes are not only how you group together weapons that a creature can be strong or weak against, but also what player races can use it.**

;//bullet texture **is the graphic that will be used as the bullet of the weapon.**

;//bullets in one shot **is just what it sounds like, how many bullets will be shot each time the weapon is shot. This will work together with spread, more bullets giving a cluster shot type effect, while just one bullet with a spread will create a machine gun type effect.**

;//stop at hit **determines if the bullet will act like a railgun or not. If set to 0 than the bullet will be destroyed when it hits a creature. If set to 1 the bullet will keep going until it reaches its maximum distance, acting like a railgun.**

;//bullet size **is the size of the graphic used for the bullet.**

;//spread **is the amount of random separation between each bullet shot. If multiple bullets are shot with a spread than you'll get a shotgun type effect. If there is a single bullet than this will determine how inaccurate the spread is. A spread of 0 means no spread. For very little spread you don't want to go higher than 0.2. For a full circle 3.0 works fairly well. This works in unison with bullets in one shot.**

;//bullet speed **is how fast the bullet will move across the screen. A bullet moving slower than 1 will sway more from the movement of the creature shooting it. This goes a way in determining the maximum distance the weapon's bullet will go.**

;//time until bullet disappears **is how long it takes for the bullet to disappear in milliseconds.**

;//fire rate **is how often in milliseconds the weapon can fire.**

;//trace **is the line that is traced from the bullet spawn point to the bullet itself. If set to 0 there will be no trace. If set to 1 than the trace will be blue. If set to 2 than the trace will be a semi-transparent bullet trace typical of bullet-time effects, if set to 3 the trace will be red, 4 will make the trace green, and finally setting it to 5 will make the trace yellow.**

;//hit visual color **is the color that the creature will be tinted for the duration of the weapon's effects on it. If set to -1 than no color tint is used. If set to 1 than a red tint is used. If set to 2 then the tint is green. If set to 3 than the tint is blue.**

;//hit visual effect particle **is what particle from the Particles.dat file is used for the duration of the weapon's hit effects. If you don't wish to have any particle spark from the creature hit than use -1. There are also several effects which use this particle line to decide which particle they use as well.**

;//hit effect sound **is the sound that is played while the effects from the weapon are active on the creature.**

;//push target back **is how far the creature hit will be shoved back. The weight of the creature comes into play, so lighter creatures will always be shoved further. You can set this to a negative to pull them closer instead of further away.**

;//push shooter back **is how far the creature that shot the weapon will be pushed back. This is essential the kick that the weapon has. You can make the weapon move the creature forward instead of back by using a negative number.**

;//fire sound **is the wave sound that is played when the weapon is fired.**

;//hit sound **is the wave sound that is played when the bullet from the weapon collides with a creature.**

;//AI hear volume **is essentially how loud a weapon is. Normal loudness being 1, anything above 1 being very loud. Anything below 1 is considered muffled.**

Well that covers the Weapons.dat file and I do hope some creative uses for this file are put to good use.

Effects/Condition Breakdown 0.9Q

Effects B1

You'll use Effects for various things. Basically it is the consequence to a Condition being true. If no Conditions are used in conjunction than an Effect will execute as often as it can. It isn't uncommon to use several Effects together to create a combination of Effects for more complex events. Below you'll see a model Effect Block's entry.

```
0;//effect number
  0.000000;//parameter1
  0.000000;//parameter2
```

```
0.000000;//parameter3
0.000000;//parameter4
```

0=nothing
As it states this effect does nothing. This is ok to use if you just want to have a filler for when you make it an actual effect. You could also simply use this like a toggle so your other parameters are saved and just switch the effect number to 0, effectively disabling the effect.

1=multiply creature speed by parameter3 for time parameter1 with creature visual effect from weapon parameter2, parameter4=disable speed change when bullet hits (0=no/1=yes)
This effect encompasses several things. The first thing is for stunning, this is the effect used with the stun tazor in the game by setting parameter 3 to 0. The second use is to slow a creature down by using a number less than 1 for parameter3. The third use is to speed a creature up, setting parameter 3 to a number greater than 1 will cause that. The fourth use is for whatever reason you want to restore a creature's speed than you would set parameter3 to 1. The time measured with parameter1 is done in milliseconds, so 1 second is 1000. Giving a visual cue of the effect being in use is what parameter2 does, simply pick a weapon number or use -1 if you don't want any visual effect. The last parameter you'll deal with allows you to decide if the effect can be disabled when a bullet strikes it, as listed 0 means it wont disable until the time runs out, if set to 1 however it can be disabled by either being hit or time running out on the effect.

2=start alien attack
This will make all the creatures in the current area active and fully aggressive.

3=drop item parameter1, amount parameter2, random area size parameter3 pixels, maximum amount of similar items in area parameter4 (0=infinite)
This basically just drops an item. The item you're going to drop is parameter1. The amount of the item you're going to drop is parameter2. The random area size is actually the distance from whatever caused the effect that the item will be dropped, if you want it dropped exactly than use 0, a short distance is 50. I've never actually found parameter4 to work that well, so don't depend on it highly to keep things in check, this is more effective I believe the larger the amount of the area size is ,but the parameter none the less should limit the max amount of that item.

4=increase creature's bar parameter3 with parameter1 (if parameter2=1, don't increase over maximum or decrease below minimum)
This effect can be used for either the player or any other creature, however it's primarily used for non-player characters or NPCs. The purpose of this effect is to manipulate a creature's bar. The bar that will be changed is selected with parameter3. The amount the bar is changed by is set by parameter1, a positive number adds to the bar while a negative number subtracts from it. To make sure the bar does go over the max or minimum you'll need to set parameter2 to 1, if you don't want the max or min values being taken into account than simply have parameter2 set to 0.

5=activate scanner with distance parameter1
This effect turns on the scanner on your radar that will show items or plot objects that are detectable by radar. The distance in pixels is set by parameter1.

6=set targeting beam, type parameter1 (0=disabled, 1=normal, 2=turns green when hits enemy), length = parameter2 + weapon length * parameter3
This effect is used to extend a line from the player using the range of the weapon currently equipped. The first setting you'll deal with is parameter to decide what type of targetting beam will be used, 0 for none, 1 for normal red line and 2 for a red line that turns green when it collides with a creature. To extend the distance in pixels you'll use parameter2. You'll use parameter3 to multiply the distance of the targetting beam.

7=set light parameter1 size parameter2 to creature (if parameter1=-1, disable light), attached to (parameter3, 0=hands, 1=legs, 2=head)
Using this effect you can attach a light to a creature. First you decide what light you'll be using from the Lights.dat file with parameter1. Next you decide how large the light will be with parameter2. Lastly you'll choose where you want the light to attach to with parameter3, the hands/middle layer using 0, the legs/bottom layer with 1, or the head/top layer using 2. Depending on where it is attach will effect which direction the light points based on the facing direction of the body part.

8=set light level addition, parameter1=type (0=map tiles, 1=items/plot_objects, 2=props, 3=creatures), parameter2=r, parameter3=g, parameter4=b
This make an addition of light to select things within the game. The first thing you decide is what you will use light addition on with parameter1, your options being 0 for terrain/map tiles, 1 for items/plot_objects, 2 for props, or 3 for creatures. Once you've chosen what you're going to add light to than you need to set up the red, green and blue values with parameter2, parameter3 and parameter4.

9=select gun parameter1
This effect basically just selects a weapon. With paremeter1 you will choose which weapon from the Weapons.dat file to use.

10=drop creature number parameter1 side parameter2 (-1=same side) tactic parameter3 (-1=default) tactic2 parameter4 (-1=default). Player can switch between a friendly creature's tactics by right clicking the creature. Don't forget to set parameter3 to -1 if you're not planning on changing the default tactic specified in creatures.dat.
This effect places a creature from whatever called the effect. The first thing you decide is what creature from the Creatures.dat file to drop with parameter1. The next thing you do is pick what side the creature is on from the Sides.dat file with parameter2. Once you've chose what creature and side than you'll pick the first tactic from the AI_tactics.dat file with parameter3, you can leave it at the default with -1. Finally you'll chose the second tactic from the AI_tactics.dat file with parameter4, like before you can leave it at the default with -1.

11=change maximum bar parameter1 amount by parameter2
This effect allows you to shrink or grow the maximum length of a bar. The thing you do is choose which bar from the Bars.dat file you'll be changing the maximum amount of. Now you'll choose how much to add or subtract from the bar with parameter2, a positive number will add to the bar will a negative number will subtract from the bar.

12=change armor level to parameter1
Using this effect you can change the armor level of the player race. You simply put whatever you want the armor level set to with parameter1. This is not accumulative, it will replace the old value with the new one, not add to it in other words.

13=enable creature detector with distance parameter1
This effect allows you to see particles representing creatures that can be detected on the radar. All you need to do with this effect is set the distance in pixels the detector can reach with parameter1.

14=play sound parameter1 (from sounds.dat) with volume parameter2
When you want to play a sound within the game this is one way of doing it. First thing you'll do is pick which sound from the Sounds.dat file to use with parameter1. Next you're going to choose what volume the sound should be played at with parameter2.

15=change creature into creature number parameter1 for time parameter2 (-1 for infinite) with creature visual effect from weapon parameter3
This allows you to change a non player creature into another for either a limited time or forever. With parameter1 you're going to choose which creature from the Creatures.dat file to change the creature calling the effect to. Now you've gotten a creature chosen you need to decide how long they will remain as the creature in milliseconds with parameter2, 1000 milliseconds being 1 second and -1 to make it remain as the new creature. Finally you decide if you want a visual effect from a weapon in the Weapons.dat file to spark from the creature for the duration of its change, you'll use -1 to disable this.

16=give item parameter1 amount parameter2, parameter3=inventory number
This effect will give the player an item. With parameter1 you'll decide which item from the Items.dat file to give. You've got an item picked, so now you choose the amount to give with parameter2. Finally you get to choose which inventory to place the item into with parameter3. Using a negative value for parameter2 you can also take away items using this effect.

17=set creature's bar parameter3 to parameter1
This effect unlike many others that effect bars is not cumulative, this will change it exactly to the value you choose for the bar. You'll first need to choose which bar from the Bars.dat file with parameter3. You've got your bar picked out so now you need to decide what value to set it to with parameter1. This effect is great for bars that are merely used to store a variable or for quickly resetting a bar to a specific value.

18=increase player's body temperature by parameter1
This effect only effects the temperature and it does nothing if one does not exist, so basically it's just a shortcut from having to chose whichever bar serves as the temperature bar. The only thing you worry about with this bar is how much you're increasing, using a positive value, or decreasing with a negative value, the temperature bar with parameter1.

19=drop plot_object parameter1, object size parameter2, random area size parameter3 pixels, maximum amount of similar items in area parameter4 (0=infinite)

This allows you to drop a plot object. The plot object you're going to drop is parameter1. The size of the plot object you're going to drop is parameter2. The random area size is actually the distance from whatever caused the effect that the item will be dropped, if you want it dropped exactly than use 0, a short distance is 50. I've never actually found parameter4 to work that well, so don't depend on it highly to keep things in check, this is more effective I believe the larger the amount of the area size is, but the parameter none the less should limit the max amount of that plot object.

20=enable large map
This effect will simply enable the player to right click the radar to get a zoomed out view of several adjacent areas. There are no parameters to this effect so you can simply leave them all at 0.

21=teleport to area parameter1 (if -1, game finds the right area) plot_object parameter2, parameter3: 0=don't transfer other creatures 1=transfer other creatures
Using this effect you teleport the player to another area or just a unique plot object. The first thing you do is choose which area from the Areas.dat file to teleport the player to with parameter1, or set it to -1 to make it find the area with the unique plot object in it. Whether you've chosen to use a specific area or not you need to choose which unique plot object from the Plot_objects.dat file to use with parameter2, it should be unique, in other words there should only be one of them in the game for this to work best. The final setup for this effect is to determine whether other creatures can teleport with you if near enough, using 0 for no and 1 for yes.

22=change side to parameter1 target for time parameter2 with creature visual effect from weapon parameter3
You'll be able to change the side a creature is on for a limited time with this effect. You'll select which side from the Sides.dat file the creature will temporarily be on with parameter1. Using parameter2 you'll setup how long it will remain on that side for in milliseconds. Lastly you'll decide what weapon visual effect from the Weapons.dat file to use with parameter3, using -1 for no visual effect.

23=continuously increase bar parameter3 by parameter1 for time parameter4 with creature visual effect from weapon parameter2
This effect is basically a new version of the old poison effect that used to be in the Weapons.dat file. You'll need to chose which bar from the Bars.dat to continually increase or decrease with parameter3. Now you'll chose the amount to reduce the bar by if it's a negative number, or to increase the bar with if its a positive number with parameter1. Now you need to decide how long the drain or increase will last in milliseconds with parameter4. Finally you'll choose if you want to use a visual effect from a weapon in the Weapons.dat file with parameter2 or simply disable it with -1.

24=fire weapon parameter1, times parameter2, (parameter3, 0=normal direction, 1=towards nearest enemy, 2=random direction, 3=shoot in direction parameter4), parameter4=shoot direction (only applicable if parameter3=3)
This effect will allow you to shoot a weapon from a creature or otherwise. Your first choice is which weapon from the Weapons.dat will be shot with parameter1. Next you'll choose

how many times the weapon will be fired with parameter2. Now you've got your weapon and how many times it fires, so you choose the direction or directions it will shoot with parameter3, 0 to just shoot in the normal direction which is typically forward, to shoot towards the nearest enemy you'll use 1, to shoot in a random direction you'll use 2, and finally if you want to shoot in a specific direction you'll use 3. Now the final parameter is only used if you chose 3 for parameter3, it uses radians, so I'll decode it for you (Top = 0, Upper Right = 0.79, Right = 1.57, Lower Right = 2.36, Bottom = 3.13, Lower Left = -2.36, Left = -1.57, Upper Left = -0.79). If you want to make small degree on a direction you can adjust the number up or down, however those are the basic 8 directions.

25=show animation parameter1, and if (parameter2, 0=continue game, 1=end game)
The purpose of this effect is to display an animation. Firstly you'll need to choose which animation number from the Animations.dat file with parameter1. Lastly you'll choose if the game with continue using 0, or end using 1 with parameter2, this will happen once the animation has finished. If it is an ending you'll use want to end the game and have parameter2 set to 1. Though if it is just a mid-game animation make sure parameter2 is set to 0 so you don't prematurely end the game.

26=fire particle parameter1 times parameter2 with parameters (spread, speed, time) taken from weapon number parameter3
Majority of the nicer looking weapons will use this effect since it can replace bullets with nicer looking particles instead. You first choice is what particle from the Particles.dat file you'll be using with parameter1. Your second choice is how many times the particle will be fired with parameter2. Your final choice will have a drastic effect on how the particles show by choosing what weapon from the Weapons.dat file to model the spread, speed, and time of the particle after with parameter3.

27=make light number parameter1 for time parameter2 (-1=infinite) size min parameter3, max parameter4
This effect will create a light from whatever spawned it. First you will choose which light from the Lights.dat file with parameter1. The next choice you make is how long the light's life will be in milliseconds with parameter2, you can make it unlimited by setting this to -1. Now you'll choose what the minimum size of the light will be with parameter3. The final choice is to set the maximum size the light will be with parameter4.

28=bar parameter1 increased by bar parameter2 * parameter3
The basic use of this effect is to use one bar to affect another. You'll first need to choose which bar will be manipulated with parameter1. Next you choose which bar from the Bars.dat file to use to change the first bar with parameter2. Last but not least you choose what multiple of the modifying bar will used with parameter3.

29=run script parameter1, parameter2=check conditions (0=no, 1=yes)
This effect will automatically run a script for you. The first thing is to pick which script from the Scripts.dat file to use with parameter1. The last and final thing you do is decide whether it should check the conditions of the script with 1 for yes, or ignore with 0 with parameter2.

30=set creature's eat item amount to parameter1

This effect is primarily used to reset a creature's eat item amount or perhaps set it to a specific value. You choose what the eat item for the creature will be with parameter1.

31=kill creature (set all bars to minimum)
The single purpose of this effect is to kill a creature. There are no parameters for this, it will just set all bars to 0 effectively killing the creature it is used on.

32=change creature's anger level to parameter1 (between 0 and 1)
You'll use this effect when you want to adjust a creature's anger. The one and only value you need to worry about is the range between 0 and 1 you will set parameter1 to. To roughly figure out what value equals what level you divide 1 by the number of total anger levels in that the creature's AI tactic uses.

33=stagger mouse by parameter1, speed parameter2 for time parameter3
Stagger mouse effect is used to make it harder to be accurate with the mouse. The first thing you choose is how many pixels the mouse will be distorted with parameter1. Next you'll need to choose the speed at which the mouse will twitch from its last position with parameter2. Last you'll decide how long the effect will last in milliseconds with parameter3.

34=change player race into parameter1
This effect simply allows you to change the player race into another player race. You'll need to decide what player race from the Player_races.dat file to change the current player race into with parameter1. You can only change a player race into another player race.

35=activate/disable script parameter1, (parameter2: 0=disable, 1=activate)
This effect will allow you disable or enable a script. Your first thing to pick is the script from the Scripts.dat file to enable or disable with parameter1. Now you can choose whether to disable the script with 0 or enable with 1, using parameter2.

36=shake screen power parameter1 time parameter2 milliseconds
Using this effect allows you to shake the screen. The amount of pixels the screen is shifted is set with parameter1. The duration of the effect is set with parameter2.

37=start rain for time parameter1 (0=stop rain)
This effect will allow you stop or start rain in an area. To start it raining for a set amount of milliseconds you'll use parameter1. Adversely you can set parameter1 to 0 and make the rain stop.

38=change game speed to parameter1 (don't set it to negative)
This effect will allow to slow down the actual speed the game runs at. You will set the speed of the game with parameter1. To set the game to half speed you would use a value of 0.5. A speed of 1 will restore the game speed to normal and anything higher than 1 will speed it up. Use this effect with great care.

39=show bar parameter1, (parameter2, 1=show, 0=hide)
You'll use this effect to hide or show a bar in the game. First you'll need to chose a bar from the Bars.dat file with parameter1. Now you can choose whether to show it with 1, or hide it with 0, using parameter2.

40=destroy plot_objects parameter1 from area size parameter2 pixels (not entirely accurate, use with care)
This is the main way you can destroy a plot object, but it isn't perfect so be careful when using it. You'll use parameter1 to choose which plot object from the Plot_objects.dat file to destroy. Now you can use parameter2 to set the range in pixels to destroy the plot object of the type you choose. A good and accurate range is usually between 50 to 100.

41=destroy items parameter1 from area size parameter2 pixels (not entirely accurate, use with care), parameter3=maximum amount of items to delete (0=all)
This is how you will destroy items that are laying on the ground in the game. The item you wish to destroy will be set with parameter1. The area size of the item to be deleted in pixels will be set with parameter2. The maximum amount of the item to be destroyed is set with parameter3 or simply set it to 0 to delete all of them. A good distance for accuracy is 50-100.

42=increase maximum carry weight by parameter1
This simply increases or descreases the maximum weight a player race can carry. If number is a positive number you will increase the maximum weight, if it is negative it will decrease the maximum weight by whatever number you used for parameter1.

43=return creature to previous frame position
This effect is primary used to keep a creature away from something, be it a creature, plot object or terrain/map tile. This should be checked every frame meaning that a script's interval using this effect should be set to 0 to be accurate and not cause weird things to happen.

44=increase player's bar parameter3 with parameter1 (if parameter2=1, don't increase over maximum or decrease below minimum)
This effect will increase or decrease a player's bar despite what called it. The bar to be changed from the Bars.dat file will be chosen with parameter3. The amount to modify the bar is set with parameter1, negative values will decrease the bar while positive values will increase the bar. To cap the bar or not you'll use parameter2, setting it to 1 if you want it to not go over or below the maximum or minimum values, if you don't need it restricted than set it to 0.

45=prevent creature from using weapon parameter1 (-1=all weapons) for time parameter2, with creature visual effect from weapon parameter3, parameter4: 0=individual weapons 1=weapon classes
Using this effect you can disable a creature's weapons. The first thing you should do for this effect is actually start with parameter4 and decide if you're going to exclude entire weapon classes with 1 or just an individual weapon from the Weapons.dat file with 0. Now back to parameter1 where you will choose which weapon or weapon class from the Weapons.dat file to exclude, this depends highly on what you set parameter4 to. Now you

can decide how long the weapon will be disabled in milliseconds with parameter2. Finally you can add some pizazz to the effect by having the creature use a weapon visual effect from the Weapons.dat file with parameter3.

46=change creature's AI tactic parameter1 (0=primary, 1=secondary) to parameter2
Using this effect you can alter a creature's primary or secondary AI Tactic. Before anything else you'll need to choose if you're going to change the primary using 0 or secondary using 1 with parameter1. To finish off the effect you need to choose which tactic from the AI_Tactics.dat file you'll change the tactic to.

47=start dialog parameter1 (from dialogs.dat)
This effect simply cues a dialog from the creature that calls it. Using parameter1 you'll select which dialog from the Dialogs.dat file to show in the game.

48=pick up nearest creature of (parameter1, 0=type, 1=class) parameter2, maximum distance parameter3, side parameter4 (-1=same, -2=any)
When you want to be able to carry a creature this is the effect you'll use. First thing you'll do is choose whether you'll pick from a creature class using 1 or a specific creature type using 0 from the Creatures.dat file with parameter1. Now you can pick a creature class or specific number from the Creatures.dat file with parameter2. For the third step you'll need to set the maximum distance the creature can be to be able to be picked up with parameter3. Last but not least you'll use parameter4 to decide what side the creature has to be on, you can also use -1 for the same side as the player or -2 for any side.

49=attach camera to (parameter1, -1=player, 0=this creature [distance parameter2 from head], 1=this position), for time parameter3 (-1=infinite)
This effect allows you to detach the camera from the player. First thing is to set up whether the camera will attach to the player with 1, or to the creature that called the effect with 0 with parameter1. The next thing you'll set up is how many pixels away from the head the camera will be placed with parameter2. The last thin you'll set up is how long the camera will remain attached in milliseconds with parameter3, or remain there forever with -1.

50=player controls nearest creature of (parameter1, 0=type, 1=class) parameter2, maximum distance parameter3, side parameter4 (-1=same, -2=any)
You can take over another creature with this effect. You'll decide if it'll be a creature class or a specific creature type from the Creatures.dat file with parameter1. The specific creature or creature class will be set with parameter2 depending on what you used for parameter1. You'll set the maximum distance the creature can be with parameter3. Lastly you'll choose the side the creature must be on with parameter4, you can also use -1 for the same side or -2 for any side.

51=nearest creature of (parameter1, 0=type, 1=class) parameter2, maximum distance parameter3, side parameter4 (-1=same, -2=any) picks up this creature
This effect makes the creature pick up the one that called the effect, in other words making the creature mountable. First thing you'll do is choose whether you'll pick from a creature class using 1 or a specific creature type using 0 from the Creatures.dat file with parameter1. Now you can pick a creature class or specific number from the Creatures.dat file with parameter2. For the third step you'll need to set the maximum distance the

creature can be to be able to be picked up with parameter3. Last but not least you'll use parameter4 to decide what side the creature has to be on, you can also use -1 for the same side as the player or -2 for any side. Make sure to use this in conjunction with Effect 50 if you want to actual be able to control the mountable creature.

52=active inventory=parameter1 (0 to 10)
This effect will allow you to switch between inventories. Simply set parameter1 to the inventory number 0 through to 10 to choose the inventory that will be active. While an inventory is active items picked up will go into that inventory.

53=continuously increase bar parameter3 by parameter1 for time parameter4 with creature visual effect from weapon parameter2, cap at maximum
This will continually increase a bar by the amount, automatically stopping at the maximum of the bar. Which bar from the Bars.dat file you're going to modify will be selected with parameter3. The amount you're going to increase the bar by will be set with parameter1. The time the increase will last in milliseconds will be set with parameter4. Lastly you can select a weapon visual effect from the Weapons.dat file with parameter2, or simply set it -1 to disable it.

54=make creature vanish for time parameter1 milliseconds
Using this effect allows you to make a creature vanish entire for a set amount of time. The time the creature will vanish for in milliseconds will be setup with parameter1. This effect is great for making a creature appear to blink in and out.

55=make mouse (parameter1, 0=invisible, 1=visible)
With this effect you can make the mouse invisible or visible again. You'll choose whether the mouse will vanish with 0 or appear with 1 using parameter1.

56=override control type (parameter1, -1=default, 0=absolute, 1=relative, 2=vehicle)
You'll be able to force the use of a specific control type with this effect. What you set the control to be is -1 for default, 0 for absolute control, 1 for relative, or even 2 for vehicle type control will be done with parameter1.

57=skip to day of journal (parameter1, 0=next day, 1 and above=skip to day number) This effect doesn't affect the time of the day.
This effect allows you to skip ahead in a journal. You set what the journal day is with parameter1, 0 being for the next day and 1 or higher will make it skip to that day number. This is mainly good for testing out how journal entries look, but it can have some uses in a mod if you're creative.

58=player (parameter1, 0=cannot, 1=can) view the inventory
You can stop or allow the player access to the inventory with this effect. You've got two choices with parameter1, they can use the inventory by making it 1, or they are denied access with 0.

59=use player's item parameter1 if there is one
In the event you want to automatically use a player's item this is the effect for it. What item from the Items.dat file will be used is chosen with parameter1.

60=player (parameter1, 0=cannot, 1=can) drop items
Depending on what you set this effect to you can deny or allow the player from dropping any items. Deny the player from dropping items by setting parameter1 to 0, or set it to 1 to enable them to drop items.

61=play song number parameter1 from music.dat
When you want to play a specific song in the game this is what effect you'll use. You'll select the song from the Music.dat file with parameter1 to be played.

62=(parameter1, 0=close, 1=open) inventory
The sole function of this effect is to force open or closed the inventory. You can set parameter1 to 0 to force the inventory closed or force it open with 1.

Conditions B2

Conditions are your if's or and's to keep an Effect from executing. All Conditions for an entry must be satisfied for any Effects to be completed. You'll often use combinations of Conditions to be more specific in what you're checking. Below is a model of a Condition Block's entry.

```
0;//condition
  0.000000;//condition parameter0
  0.000000;//condition parameter1
```

0=must have item parameter0 amount parameter1
You'll use this condition to check if a certain item is in the player's inventory. You choose which item from the Items.dat file to check with parameter0. One the item is chosen you decide how many of the item they must have with parameter1.

1=must be distance parameter1 + object size from plot_object parameter0
In simplest terms this checks if the creature is a set distance from a plot object. You'll decide which plot object from the Plot_objects.dat file to check with parameter0. Now you can set the distance the creature must be from the plot object with parameter1.

2=must be distance parameter1 + object size from plot_object class parameter0
This condition checks if the creature is a set distance from a plot object class. You'll decide which plot object class from the Plot_objects.dat file to check with parameter0. Now you can set the distance the creature must be from the plot object class with parameter1.

3=creature's bar parameter0 is greater or equal to parameter1
You'll use this to check a creature's bar to see if it is greater. What bar will be checked is set with parameter0. The number you'll compare the bar against to see if it is greater or equal is setup with parameter1. This condition is well suited as a condition for non player creatures.

4=killed all creatures parameter0 from area parameter1 (-1=current area)

When you want to check if all the creature type in an area are dead you can use this condition. Using parameter0 you'll choose which creature from the Creatures.dat file to check. With parameter1 you'll decide what area to check or use -1 for the current area.

5=creature's bar parameter0 is smaller than parameter1
This condition is well suited to check if a creature's bar is less than a certain value. The bar from the Bars.dat file you'll be checking is set with parameter0. The value the bar must be less than is configured with parameter1.

6=player is race parameter0
You'll use this condition to make sure the player is a specific race. The race from the Player_races.dat file the player must be is set with parameter0.

7=player is not race parameter0
You'll use this condition to make sure the player is not a specific race. The race from the Player_races.dat file the player must not be is set with parameter0.

8=random integer between 0 and parameter0 is 0
This is your basic random condition, this is good for keeping things from happening one hundred percent of the time. You'll set the number to check between with parameter0. Simply because this number seems to change so frequently you'll want to make sure your number is high like 100 if you want it to only happen rarely and even higher if their is only a minute chance it should happen.

9=player is in shade
This condition will check if a creature is near a plot object that is set to provide shade in the Object_defintions.dat file. This condition has no parameters to be set and both can be left at 0.

10=creature has eaten parameter0 eat items
Using this condition you'll check how many eat items the creature calling the condition has eaten. The amount to check for is set with parameter0.

11=item parameter0 is (parameter1=0=wielded, parameter1=1=not wielded)
The purpose of this condition is to check if specific item is wielded or not. The item from the Items.dat file to check is selected with parameter0. To check if the item is wielded you'll use 0 for parameter1, or 1 to check if it isn't wielded.

12=game difficulty is (parameter0, 0=higher than, 1=lower than, 2=equal to) parameter1 (0=easy, 1=normal, 2=hard)
You'll check what difficulty the player is using with this condition. To check if the difficulty is higher you'll use 0 with parameter0, for lower than you'll use 1 and for equal to you'll use 2. The difficulty that will be checked to either be higher, lower, or equal to what parameter0 was set to is checked with parameter1.

13=area is parameter0

This condition merely checks what area the creature is in. You'll choose which area the creature needs to be in with parameter0.

14=creature nearer than distance parameter1 + creature size pixels from creature parameter0
This is how you'll check the distance between two creatures. First thing you'll do is choose which creature from the Creatures.dat file is being detected with parameter0. Once you've got the source creature you need to set the distance with parameter1 that it will need to be at to make the condition true.

15=player is (parameter1, 0=nearer, 1=farther) than parameter0 pixels
When you want to detect the distance of the player race. Working a bit in reverse you'll need to choose if you're detecting nearer with 0 or 1 for farther using parameter1. How far or near the player is in pixels is determined with parameter0. This is a good condition when you want to have something happen when a player approaches within a certain distance or maybe even gets far enough away.

16=player has parameter0 free weight
This condition will simply check how much free weight is left from maximum the player race has. The weight the player must have to make the condition true is set with parameter0. This condition isn't useful if you don't use weight in your mod.

17=is on terrain parameter0
This will check what terrain the creature is on. The terrain from the Terrain_types.dat file is chosen with parameter0.

18=creature's bar parameter0's maximum amount is bigger than or equal to parameter1
This maximum value of a creature's bar is checked with this condition, ideal for non player creatures. The bar from the Bars.dat file that will be checked for the creature is setup with parameter0. The value that the bar must be equal or greater to for the condition to be true is set with parameter1.

19=creature's AI tactic parameter0 (0=primary, 1=secondary) is parameter1
Using this condition you can check what tactic the creature is using. Which tactic you'll be checking the, 0 for primary and 1 for secondary will be configured with parameter0. The number of the tactic from the AI_tactics.dat file will be set with parameter1.

20=creature's anger level is (parameter0: 0=bigger than or equal to, 1=smaller than) parameter1
You'll use the condition to check a creature's anger level. First decide if you want to check if the anger level for the current tactic being use from the AI_tactics.dat file is bigger/equal to with 0, or smaller than with 1 using parameter0. Next you'll choose what you're comparing against with parameter1.

21=creature is of side parameter0
This condition is used for checking the side a creature is on. The side you're checking from the Creatures.dat file is parameter0.

22=creature is of (parameter1: 0=type, 1=class) parameter0

This checks if a creature is of a specific type, meaning its exact identifier number or a class. Firstly you choose if you're going to check the exact identifier of the creature from the Creatures.dat file with 0 or the creature class with 1 using parameter1. The type or class number is than specified with parameter0.

23=creature (parameter0: 0=is, 1=is not) player
This simply checks if the creature is the player or not. You've got only two choices the condition is true if the creature is the player with 0, or is not the player with 1 for parameter0.

24=player's bar parameter0 is greater or equal to parameter1
This will only check if the player's bar is greater or equal to a value regardless of what calls it. The bar from the Bars.dat file to be checked is set with parameter0. The amount it must surpass or be equal to is determined with parameter1.

25=player's bar parameter0 is smaller than parameter1
This will only check if the player's bar is smaller to a value regardless of what calls it. The bar from the Bars.dat file to be checked is set with parameter0. The amount it must be less than is determined with parameter1.

26=creature is not of (parameter1: 0=type, 1=class) parameter0
This checks if a creature is not of a specific type, meaning its exact identifier number or a class. Firstly you choose if you're going to check the exact identifier of the creature from the Creatures.dat file with 0 or the creature class with 1 using parameter1. The type or class number is than specified with parameter0.

27=creature's last dialog (parameter1: 0=was, 1=was not) parameter0
You'll check the last dialog that popped up over the creature with this condition. The dialog from the Dialogs.dat file to be checked is selected with parameter0. Whether the condition will be true if it was the dialog number using 0, or was not with 1 is configured with parameter1.

28=it (parameter0: 0=is, 1=is not) raining
This will simply check if it is or isn't raining. Set parameter0 to 0 if the condition proves true when it is raining or use 1 if it'll be true when it isn't raining. This is a great way to only make things happen when it is raining, like lighting flashes.

29=amount of parameter0 (-1=all) creatures in area is higher than or equal to parameter1
You'll use this condition the amount of a creature type being higher than a number. Choose the creature from the Creature.dat file with with parameter0 or use -1 for all creatures. The number of creatures must be greater or equal to parameter1 for the condition to prove true.

30=amount of parameter0 (-1=all) creatures in area is lower than parameter1
You'll use this condition the amount of a creature type being lower than a number. Choose the creature from the Creature.dat file with with parameter0 or use -1 for all creatures. The number of creatures must be lower than parameter1 for the condition to prove true.

31=amount of parameter0 class creatures in area is higher than or equal to parameter1
You'll use this condition the amount of a creature class being higher or equal to a number. Choose the creature class from the Creature.dat file with with parameter0. The number of creatures in the creature's class must be greater or equal to parameter1 for the condition to prove true.

32=amount of parameter0 class creatures in area is lower than parameter1
You'll use this condition the amount of a creature class being lower than a number. Choose the creature class from the Creature.dat file with with parameter0. The number of creatures in the creature's class must be lower than parameter1 for the condition to prove true.

33=creature (parameter0, 0=is, 1=is not) being carried by another creature
Using this condition you'll check if the creature is being carried by another creature. You have to choose whether the condition will be true if it is being carried with 0, or is not being carried with 1 for parameter0.

34=creature (parameter0, 0=is, 1=is not) carrying another creature
Using this condition you'll check if the creature is carrying by another creature. You have to choose whether the condition will be true if it is carrying another creature with 0, or it is not carrying another creature with 1 for parameter0.

35=must be distance parameter1 from item parameter0
This checks the distance of an item. The item to check from the Items.dat file is configured with parameter0. The distance in pixel the item must be is set with with parameter1. A good distance is usually between 50-100 for good accuracy.

Map Editor 1.0Q

In this section I'm going to cover the Map Editor that used within Notrium. The Map Editor is your level editor for your mods. Using the Map Editor can great simplify the level design process so be sure you are proficient with it before you decide to get heavy into modding. The file that the Map Editor outputs to is the Terrain_maps.dat. You'll only have to edit the name of a Terrain Map if you want, everything else can be done in the Map Editor.

Getting Started C1

1. The first step is enabling the Map Editor which wont be enabled by default. You'll go into the /data folder and find the file called setup.dat and change this line `1;//create game initialization debug file` **so that it show the 1 instead of a 0. That will enable the Map Editor.**

2. The second step is running Notrium and selecting the mod you'll be editing from the menu by clicking the name under Mod.

3. Once you've chosen the mod you want to edit you'll select Start Editor rather than Start Game.

Navigating the Map Editor C2

At this point you should be in the Map Editor itself. The first thing you'll notice is that when you move the mouse the screen scrolls. You'll be using the mouse to move around the current terrain map. To make the map zoom in and out you'll be using the scroll bar on your mouse. Zooming in and out will allow you see things in more detail or get a better overall look at the current terrain map you're working on. If you have multiple terrain maps you'll move between them with the Z and X keys. Lastly but certainly most important of all you can press F1 at any time to bring up overlaying text help incase you forget any keys.

Resizing the Terrain Map C3

You'll probably not be happy with the scaling of your terrain map. It's a good idea to zoom out when you're scaling your map so you can see the proportions. When you want to make the terrain map taller you'll use the End key. To do the opposite and make it shorter you'll use Home. When you want the terrain map longer you'll use the Page Down key. To make the terrain map thinner you'll use the Delete key. It's important to note that changing the scaling after you've edited it can cause things to be cut off and lost or out of position. So expect to fix things if you change the scale post editing it. So make sure to take serious note on this and try to figure out the scale of your map first before you do any editing.

Using the Object Palettes C4

There are six different object palettes you'll choose from when editing your terrain map. When I say object I mean anything you can place, not just an item or a plot object since terrains/floor tiles can be placed individually as well. The next thing you'll need to note is the identifier for each object is not shown but they go from left to right, top to bottom, so identifier 0 for the palette will be the in the most upper left. Once you've chosen an object which will replace the mouse cursor, you will place them by clicking the left mouse button. To delete the selected object you'll use the right mouse button. You can exit out of a palette at any time without having to select an object by pressing Esc. Below I'll cover the six palettes with the key that brings up the palette.

F2 key = Terrain Palette, this will be used for placing terrains or floor tiles to put it simply.

F3 key = Item Palette, you'll use this to place items on the map. It's very important that you remember which item is which if you're using the same graphic and name for several different items.

F4 key = Plot Object Pallete, you'll use this palette to place plot objects on the map. Like before you'll want to make sure if you've got different plot objects with the same graphic

and name that you are able to distinguish between them, I highly suggest using unique names to prevent this problem.

F5 key = Creature Palette, this is how you'll place creature's on the map. It is possible like before to have many creatures sharing the same name and even graphic yet serving different purposes, so make sure you can distinguish or give them a unique name.

F6 key = Light Palette, this is how you will place lighting on a map. It's important to note that color tints of a light will not show in the palette, only its original base color will show. So you will have to test the map out to see how the lighting actually looks within the game.

F7 key = Props Palette, you'll use this to place props on a map. There is no reason why each and every prop can't have its own unique name, this should keep you from getting confused which prop is which even if they have the same graphic.

Manipulating Objects C5

So by now you should know how to move around a map and even get to the various palettes. This section will cover what you do with objects once they are selected from a palette or even selected once they are already placed. First thing I'll cover is the individual object types and how to manipulate each.

Terrain/Floor Tile - These are actually pretty basic, you just left click to place it and than right click when you're done. There is not further manipulation beyond that. To delete a terrain you simple replace it with another terrain. The No Random Object terrain is special and should be deleted by once again picking that terrain and than using the left mouse button to delete the already placed No Random Object terrain that is there. You can flood fill a terrain by using the F key while it is replacing the mouse, causing it to replace any tiles similar to the one you flood filled over unless they are sectioned off.

Items - You'll place items with the left mouse button just like terrains. You can also rotate it clockwise with the W key or counter-clockwise with the A key. To fine tune the rotation you can use Left Control to rotate it at exact 45 degree angles or the Left Shift key to move it at exact degrees. You'll use the W key to increase the amount of items placed and S to reduce the amount, the amount should at least be 1. You'll use the Right Mouse Button to pick up an item once it is placed and the Right Mouse Button again to delete it, or the Left Mouse Button to paste down a copy of it. Lastly you can move a item to the front by floating your mouse over it till it is highlighted and pressing the F key.

Plot Objects - You'll place plot objects with the left mouse button. You can also rotate it clockwise with the W key or counter-clockwise with the A key. To fine tune the rotation you can use Left Control to rotate it at exact 45 degree angles or the Left Shift key to move it at exact degrees. You'll use the W key to increase the size of the plot object placed and S to reduce the size. You'll use the Right Mouse Button to pick up a plot object once it is placed and the Right Mouse Button again to delete it, or the Left Mouse Button to paste down a copy of it. Lastly you can move a plot object to the front by floating your mouse over it till it is highlighted and pressing the F key.

Creatures - You'll place the creature with the left mouse button. You can also rotate it clockwise with the W key or counter-clockwise with the A key. To fine tune the rotation you can use Left Control to rotate it at exact 45 degree angles or the Left Shift key to move it at exact degrees. You'll use the W and S keys to adjust the side the selected creature is on. You'll use the Right Mouse Button to pick up a creature once it is placed and the Right Mouse Button again to delete it, or the Left Mouse Button to paste down a copy of it. Lastly you can move a creature to the front by floating your mouse over it till it is highlighted and pressing the F key.

Lights - You'll place lights with the left mouse button. You can also rotate it clockwise with the W key or counter-clockwise with the A key. To fine tune the rotation you can use Left Control to rotate it at exact 45 degree angles or the Left Shift key to move it at exact degrees. You'll use the W key to increase the size of the light placed and S to reduce the size. You'll use the Right Mouse Button to pick up the light once it is placed and the Right Mouse Button again to delete it, or the Left Mouse Button to paste down a copy of it. Lastly you can move the light to the front by floating your mouse over it till it is highlighted and pressing the F key.

Props - You'll place props with the left mouse button. You can also rotate it clockwise with the W key or counter-clockwise with the A key. To fine tune the rotation you can use Left Control to rotate it at exact 45 degree angles or the Left Shift key to move it at exact degrees. You'll use the W key to increase the size of the prop placed and S to reduce the size. You'll use the Right Mouse Button to pick up a prop once it is placed and the Right Mouse Button again to delete it, or the Left Mouse Button to paste down a copy of it. Lastly you can move a prop to the front by floating your mouse over it till it is highlighted and pressing the F key.

The last thing left to discuss is how to mass select and what it is used for. Mass selection is done by click and holding the left mouse button and move outward till you've got a box around what you want selected. It's important to note that terrains cannot be mass selected and must be edited individually. Once you release the left mouse button all the selected objects will float. With the selected objects floated you have two choices, you can delete them all using the Delete key or you can place them at their new destination by pressing the left or right mouse button. You'll generally use mass selection to move structures you've built to new locations. You cannot rotate, scale or edit a mass selection, only move, place or delete it.

Creating a new Terrain Map C6

There are actually two main ways you can create a new map. The first and most basic is to simply press the F11 key which will create a single tile map that you can expand to whatever size you wish. The second method is a bit more complex but not much, you'll simply choice a map you want to duplicate and with it as your active terrain map you'll press the F8 key, this will make an exact duplicate of that map as the new map. Cloning with F8 is obviously a preferred method if you have already created a template terrain map to work from, saving you quite a bit of time in the long run. I do suggest using cloning as well when you want to try something unique and aren't sure how it will affect the overall area, so simply clone it and than edit the new map so that your old one is preserved.

Deleting a Terrain Map C7

Before you decide to completely delete a map you can also wipe it entirely clean and down to one tile with the F9 key. However if you do wish to entirely delete a terrain map you will need to go into the Terrain_maps.dat file and delete the entry for it. You should refer to the breakdown for that file for assistance and to make sure you delete the entry properly.

Naming a Terrain Map C8

Naming a terrain map can actually not be done within the editor itself. You're going to have to open up the Terrain_maps.dat file and find the entry that matches the name of the map you want to change the name to. Once you've found the entry simply replace the word before *;//name* **with whatever you want the terrain map to be called.**

Saving your work C9

To save your work even before exiting you can press the F12 key. Be aware that if you've done something that is hard to reverse that saving your work will cause problems. For the reason previously stated try and clone a map if you want to test something out, that way if you don't like the changes you can simply delete the cloned map and not have to worry about getting the original back to what it was before since you will still have the original.

Exiting the Map Editor C10

To exit the Map Editor you simply need to press Esc until it prompts you to save with F12 or exit without saving your changes with F10. If you were not happy with your changes and wish to revert back to your last saved version than you should choose F10. However if you are satisfied with the changes and/or are going to test them out you need to use F12. Unsaved changes cannot be tested as there is no test mode for the editor, you actually need to run the mod to see the changes you've made.

Advanced Modding 1.1Q

In this section I'll be covering advanced modding that you can use to create fancy effects like those used in the Notrium game itself. I'll also be covering specific things I didn't cover in the FAQ already that while not essential will aid you in making a more complete mod.

1. How do I make my own graphics for Notrium ?
This is an especially good question and definately one that comes up quite often. There are several different graphics used in Notrium and they have different specifications about what can be used with them.

Terrain graphics - The first and most basic is a terrain graphic or floor tile. It can be a JPG or BMP and should be 128x128. The color depth should be 16 or 24 bit. You may be able to use other formats but this is the suggested format for terrain graphics.

Animation Frame graphics - This is another unique graphic you'll deal with, however it is quite simple. This graphic should be in JPG format preferably and in 16 or 24 bit color depth. The size of the image should be 1024x768.

Items/Plot Objects/Props graphics - All three of these use the same graphic type. You can either use a TGA or a PNG, both of which support alpha masking and are used by these three graphic types. The graphic size should be a multiple of 64x64 and generally no larger than 256x256. For low detail or small graphics like items you should use 64x64. For medium detail you should use 128x128, be aware this use more memory. Finally for high resolution graphics you can use 256x256, this will use alot of memory so it should not be used in abundance. If you use a 16 bit image the alpha channel will only be 2 colors, meaning the mask will either make something completely visible or invisible, but but nothing in between. If you need multiple shades you should use a 32 bit image instead which uses a 256 color alpha channel, allowing for smoother blending of what is visible and what isn't.

Particles/Light graphics - These two graphics work very similar so I'm going to mesh them together. you can use either TGA or PNG. As for the graphic size there is no set graphic size for lights or particles, however the particles should be on the smaller side since you will be using them alot. You'll more than likely want to use 32 bit rather than 16 bit as particles and lights do not look good as 16 bit due to the lack of depth in the alpha channel with a 16 bit image. Particles and lights should also generally not have a color and should generally be in the grayscale, allowing you to tint them whatever color you wish easily.

Bullet graphics - These graphics are similar to particles in that should be on the smaller side but graphically when shown in the game they aren't as clean as a particle is. They can be a TGA or a PNG format graphic. Generally they should be around 16x16 but they can be larger if you like, you should keep them at a multiple of 16x16 to make scaling look better on the graphic. Most bullets wont use complex alpha maps so it is very possible to get away with a 16 bit bullet graphic and save yourself some memory in your mod.

Non-Player Creature graphics - This is a graphic that is not used for a player race in the game. The graphic can either be a TGA or a PNG. The color depth can be 16 bit or 32. Unlike other graphics explained so far a Non-Player Creature graphic uses cells to form animations within the game. The graphic for each creature is divided into 16 cells or 4x4 cells. There are three sizes the graphic can be, the first being 256x256 for a low resolution creature which is generally what you should use since it uses the least amount of memory. The medium resolution is 512x512 and will provide some decent detail, the cost coming at memory. The highest resolution is 1024x1024, it will provide the best detail but will cost a good chunk of memory so it should only be used sparingly. Below is a diagram of what each cell is for, each cell being numbered for your convenience.

1	2	3	4
5	6	7	8

| 9 |10 |11 |12 |
|13 |14 |15 |16 |

[Frame 1 = Head, Frame 2 = Upper Body, Frames 3-5 = Attack Animation, Frames 6-10 = Walking Animation/Lower Body, Frames 11-16 = Death Animation.]

Player Creature graphics - This is a graphic that is used for a player race in the game. The graphic can either be a TGA or a PNG. The color depth can be 16 bit or 32. Unlike other graphics explained so far a Non-Player Creature graphic uses cells to form animations within the game. The graphic for each creature is divided into 16 cells or 4x4 cells. There are three sizes the graphic can be, the first being 256x256 for a low resolution creature which is generally what you should use since it uses the least amount of memory. The medium resolution is 512x512 and will provide some decent detail, the cost coming at memory. The highest resolution is 1024x1024, it will provide the best detail but will cost a good chunk of memory so it should only be used sparingly. Below is a diagram of what each cell is for, each cell being numbered for your convenience.

| 1 | 2 | 3 | 4 | Row 0
| 5 | 6 | 7 | 8 | Row 1
| 9 |10 |11 |12 | Row 2
|13 |14 |15 |16 | Row 3

[Frame 1 = Head, Frame 2-6 = Lower Body, Frames 7 = Upper Body/Unarmed, Frames 8 = Empty Frame, Frames 9-12 = Attack Animation1, Frames 13-16 = Attack Animation2.] Additional Attack animations can be used in another graphics file for the same Player Race creature and can be assigned to a specific weapon class for it. You'll notice I numbered the rows this time, that is to help you when assigning an attack animation with a weapon class for a creature. Make sure to reference player races in the actual game if you are still confused.

2. How do I make my own sounds for Notrium ?
While this question doesn't come up nearly as much it is something good to know. There is the sound format you work with while modding Notrium, this is the wav format for the sounds. For the waves you're pretty much free to use whatever type you like as you'll notice the various wave files used in the game are of no specific bit depth. In the event a wave you made doesn't work simply try a different format of wave. These are some that have been used 8 bit unsigned mono, 16 bit signed mono, and 16 bit signed stereo.

3. How do I make a mountable creature like the hoverbike ? (You can use Item 185 as a reference.)
You'll need to understand how it was done first obviously. The first step is creating a creature, graphic and all that will serve as the mountable creature. By now you should have a fully working creature. Now you should create an item, though there are other ways to trigger the actual mounting I'm just going to cover the item method. The next thing you'll need is a disabling script which is how the vehicle is dismounted and things like controls are returned to normal. The script should have a condition to check if the player is still being carried with Condition 33 and any additions conditions like a check on an item that will serve as fuel for the mounted creature, it will have no effects. Now you've got your

disabling script you want to create the item that will allow you to mount the creature, assigning the script you just made as the *;//wielded item disabling script***. So now you've got a way to disable you should assign it some slots or the wielding item disabling script wont have much use. The final steps are to add three effects, the first to have the player picked up by the mountable creature being Effect 51, second effect is Effect 50 which will allow the player to take control of the creature it has mounted, and the final being Effect 56 which is used to change the control type to a vehicles controls. With all the effects in place you need only to test it out and make sure everything works and than you should have a mountable creature much like the hoverbike you get in the main game.**

4. How do I change a bar based on another with a script instead of using specialty in the Player_races.dat file ?
I didn't cover a particular effect in very much depth before, but you'll use Effect 28 to quickly affect change in one bar using another. Alternatively you can also check with a condition that a particular bar is a certain value and than change the bar of your choosing by however much based on what the bar was. The alternative is alot more rigid and will require many different scripts, so you are encouraged to use Effect 28 to maintain efficiency and keep your scripts to a minimum.

Questions Answered 1.2Q

I've specially set aside this question for actual forum users from the board to get important modding questions answered. The originator of each question will be credited with the questions they give with their name before each question. You should already have a good understanding of how to mod since I generally am not going step by step, instead I'll merely explain a concept. So while this may not be as easy a read as many would like, understand that these answers are to serve only as a template.

(Casanova) 1. How to properly use the "call position" "calling creature" options in the scripts. dat?

This is actually a good question and one I'm sure that is on several people's minds. Generally one should keep in mind that scripts are meant to affect the player above all. As for properly using either, it is simply a matter of what you're trying to do, if you want something spawning from the mouse's position than *call position* **can come in handy in accomplishing that. As for** *calling creature* **it is more based upon what you want the script to affect, depending on what you choose is what the effects will happen to.**

(Mageking17) 2. How can you effect a creature that is carrying the player (such as hoverbike) without knowing exactly what that creature will be?

To answer this question you need to realize that there are certain things that will be universal or rather should be, like a health bar. So basically to answer your question if your effects are going to be general and not specific to any creature you'll need to make sure that what you check every creature has. You probably wont have everything be mountable like the hoverbike however. So to make your life easier simply make all the mountable creatures all the same creature class and than simple check for that creature class.

(Mageking17) 3. Can a creature other than the player change weapons?

Yes actually, but it isn't natively supported. My suggested method is to simply change the creature with Effect 15. The new creature should use a different weapon than the original one but look exactly the same. Any bars the old creature had will carry over to the health condition will remain the same.

(Mageking17) 4. Can a creature being controlled by the player fire a ranged weapon without the player specifically telling it to?

Yes you can actually use a specific effect to achieve what you describe. You'll use Effect 50 to achieve the means of remotely controlling another creature.

(Mageking17) 5. Can you have weapon shoot a random creature instead of a random direction?

This is actually slightly tricky. The main problem you're face with is that while items, plot objects and scripts will allow for multiple condition/effect sets, a weapon will not. To start you off you're going to need a bar that changes to different numbers randomly, you'll set it up with a script. Next thing you'll do is create as many scripts as there are going to be random creatures. Each script will fire a specific creature, but here is the catch, only if the bar is set to a specific value, meaning you'll have to do below value and above value check to make it entirely specific. Now you should have scripts setup all you do is that when the weapon is fired it triggers all of the scripts, but it checks their condition. What you will hopefully get is a random creature being shot depending on what value the bar was at when the weapon was fired.

(ZeXLR8er!!) 6. How to set up a weapon clip ?

The first step to creating a weapon clip is having a bar that will represent the clip itself. Once you've got the bar made you want to make sure the player races that will use the weapon clip actually have the bar as well. To keep things simple you can use a flat clip much like was used in System Shock 2, where there was one main energy source. Using a flat clip saves you the trouble of having to define the clip size for each individual weapon. Now to make sure a weapon doesn't fire with an empty clip you need to put a condition to make sure the bar isn't empty. Next you'll need an effect that drains the clip bar when the weapon is fired. To refill the bar itself you'll need a script that refills the bar when it is empty either fully or in chunks, that will provide the illusion or reloading. If you want to get fancy you can have several different reload scripts based on what item is being wielded that reload the bar at different rates. Even though it isn't as simple I do suggest using different scripts for the different reload times. So now you should a bar that drains and refills and a weapon that wont fire if the bar is empty. This will take some tooling with to get the exact results you desire but this should provide a good overview of what to do.

(Mageking17) 7. How can I change the player into a ship (like the minigame with the escape pod)?

This is actually more simple than making a mountable vehicle believe it or not. All you need to do is have the player change into the player race that represents the ship with Effect 34. You'll also want to make sure it overrides the controls with Effect 56 so that you can make the controls vehicle controls. In the actual game there other exclusions as well like locking the inventory with Effect 58.

(Doogsy) 8. How to make a sophisticated ally?

This question has so many layers and venues it's hard to cover them all. I'm going to give you a general idea and basically let you run with it. The most base way to give anything more sophisticated AI is simply to change its AI through a creature block. By using multiple AI's you can have a creature aggressive while it is healthy yet scared when it is close to death. Even something as simple as what I just described creates a more intelligent AI and gives the illusion of self preservation.

(ZeXLR8er!!) 9. (How can you make a creature that shoots more than one weapon like the Dopplegangers?

The secondary or tertiary(third) weapon, or even beyond that can all be done by simply firing a weapon with Effect 24 using a creature's timed effect block. There are ways to complicate it, like using a bar that acts as a clip, but it is not required. All you really need is a weapon and a block to run it with its own unique conditions.

(ZeXLR8er!!) 10. How do you make a weapon that shoots creature-seeking missiles?

This question has been around for some time actually so I thought I'd make a note to answer this. First thing you'll obviously need is a creature, you're going to adjust its minimum and maximum speed so that it is always in motion, generally meaning you set them both to the same value making it unable to stop. How tight it can turn is up to you, but generally you don't want it being to turn too tight or wont propel forward as much. The creature itself will need a very close range weapon that is very precise. The weapon should kill the creature that fires the weapon. So now you've got a destructible creature when it hits something, all you need to do is use another weapon that drops the creature. Lastly incase I forget you'll want the missle type creature to have a very aggressive AI where it will go straight for an enemy. There you go, you should have a creature seeking missle following these guidelines. Being that these are just guidelines you can adjust them to fit your need and possibly even change them entirely.

(Zombie) 11. How do you make and use multiple inventories?

To make an inventory you merely switch to that inventory to active one with Effect 52, there are automatically 11 inventories. You do not have to use the 11 inventories but you cannot create more. You're generally going to use Effect 16 to place things in a specific inventory. There is no condition to check which inventory the player is currently using however so you will have to get creative and make a bar specifically for that and make sure to set the bar to a specific value each time you set the inventory to a specific value. If you've set up the bar that stores the supposed inventory number you can than check the bar like

any other and force things to happen or not happen depending on which inventory is in use.

(Zombie) 12. How can you make buildings like houses, power plants, base structures, etc. for various mods and what are uses for them?

A building is literally whatever you want it to be. To make an outside structure with no roof like ruins perhaps you'll probably want to create several walls. The walls should be props unless you're going to interact with them than they need to be a plot object. A opening and closing door for example would need to be a plot object. To make facade for a building you'll simply create what the outside will look like with either a single object or several props and plot objects and when the player interacts with a part of it or the whole thing even than the player is sent to an area that represents the interior of the building. The most obvious purpose of any base or building is for protection. Secondly building can be used to stockpile items or even for a variety of story based events.

(Zombie) 13. How can you make objects like electric plants, health plants, and fungi respawn?

This is something I consider to be more or less an iffy thing to do. In other words I'm not guaranteeing exact results using this method but it will respawn plot objects. For this to work you must already have a plot object setup for whatever you want to be respawned. The new plot object is going to be a spawner and should not be visible itself. You're going to set its use to timed so that the effect happens at set intervals. The first effect you're going to use will surprise you, but its going to be Effect 40 to get rid of any previously existing plot object of the same type, thus making sure there is only ever one no matter how many times the effect is run. The second effect you'll use is one to drop the plot object you want spawned at the exact position with Effect 19. That is it, if done properly you should have a new mushroom, plant or otherwise spawn after a given amount of time when there isn't one there. If there is one there than all that will do is replace it, making sure there is only ever one at that spot. I consider this a little messy, but it should get the job done.

Credits 1.3Q

Modding FAQ written by Michael Quigley, AKA Quanrian.
Additional editing and proof reading is done by the infamous Ville Monkkonen.
Casanova for conversion from document format to PDF and for all the great modding feedback he's provided on the forum.

I'd personally like to thank these people from the forum for the questions they provided...

Casanova
Doogsy
MageKing17
ZeXLR8er!!
Zombie

I’d like to thank everyone in the Notrium forum for his or her continued support and questions!

I know this FAQ isn't very clearly written but there is just so much information to cover and I did my very best to cover as much as I could remember myself. If you have further questions you are always welcome to use the Notrium Modding Forum for that purpose.

I can be reached at quanrian@yahoo.com for any questions you might have.