GpFinalWar 0.7 apha – WarGame Maker beta 1

Wargame Maker manual

0 - Table of contents:

1 – Programs overview:

Wargame Maker is composed of 4 programs:

- TileSEditor
- UnitEd
- MapEdit
- CampEdit

2 of them are new, TileSEditor and UnitEd, You will use them to make the actual mod for the game (composed of a .fwd file, a .tsd file, and a bunch of resources files).

TileSEditor is, logically, the one you will start with if you want to make a full mod for the game. It lets you load a .bmp which contains your tiles (the background gfx, or terrain), and then define, for each tile, if you can build on it, if it's destructible, …

You can also add a foreground for each tile, and define tile groups, which will be used in the missions editor to make maps more quickly.

TileSEditor will save a .tsd tileset file, part of the mod.

UnitEd is the other program needed to make a mod. It will allow you to make your own units and buildings.

Making a set of units and buildings takes 4 stages. First, you'll load up all your units gfx (bases and turrets), and define their "hot spot" (center of the picture).

Then you will need to define attack scripts that you will affect to units and buildings.

Then, you can set the units definitions (their armor, attack, …), and do the same for the buildings.

And finally, give it a name, put in your name, and find some names for each team that will fight for control of whatever you'll imagine !

Mapedit is almost the same as before, a map editor with basic functions, nothing more. But it will need a tileset and a definition file, so you will be able to edit a map more quickly with the tile groups you made before, and also you'll be able to edit missions, with more options than before.

Note: Mappy can be used to edit maps. You'll need my map converter to convert them to a .map file. Then you would open it with MapEdit to add the mission data to the file.

CampEdit is the last step. You have to make all your mission files for all the teams before you can make a campaign. In this program, you'll simply link all your mission files to the correct team in the correct order, and define the campaign title, author, and other team select and ending texts.

3 other programs are also in the archive:

First, MapConv, is a Mappy to .map converter. This means you can create your maps using Mappy, and only use the MapEdit to add mission data to the files. You'll need to cut the 1st block of your tileset to use it in Mappy. Then New map -> click Yes to set the size of the blocks. Set it to **24**x**20**, and accept. Then import your big tileset bmp (the ?x**20** one). Edit your map, save it, and Export it. Select Data as text, click Ok. Then select Map Array, leave it on 2D format, and click Ok. Now you can open the .TXT file exported with MapConv.exe and convert it to a valid .map file.

PalInject, will be used for palette stuff. It will inject the GpFinalWar's user interface palette into your custom palette (which should have only **232** colors, the other ones will be trashed). See the chapter on palettes to see how the program will help you.

And the last one, Bmp2paK, will let you build a pack file from almost all the bmps of your project (at least, the ones contained in the .fwd file, and the *tileset.tsd* file, which includes everything in the sub-directories, and the photos. See the last part, it explains how it works.

2 – Before you start

First, take a look at the chapter about the palettes. This will be useful, as you'll have to convert your objects to 8 bit color mode anyway.

As the game will look for the right files at the right places and not anywhere else, the TileSEditor and UnitEd programs will look for the files using the same rules. So you'll need to organise folders and put the gfx in them before you start working.

The game will look for both a *gamedir*.fwd file and a *gamedir* folder. In this folder, you'll need 4 sub-folders:

- foreg
- base
- turret
- build

Your tileset file MUST be named tileset.tsd and be placed in the *gamedir* folder, along with the tiles bmp.

The names of the folders are explicit, foreg is where you will put the foreground gfx, base the bases gfx, turret the turrets gfx, and build is where you will put the buildings' background gfx.

The other files just go into the *gamedir* folder.

Your files **MUST** be named with **8.3 dos restrictions**, because they will be loaded from the smc. Note, that if you plan to use the Gamepak creator (*Bmp2paK.exe*), the restriction is then **11.3** (**15** as filename maximum length). This applies to all the files for the game.

IMPORTANT: There should not be 2 pictures with the same name in different drawers – or the game simply won't load the second and assume it is the same picture than the 1st !!!

The bmp files must be 8 bits bmps with palette. Their palette won't be used, but it does matter, as they all will be displayed with the main palette of the game. So before you start adding the files to your mod (the editors won't tell you that the bmps are not in the good color mode or palette), convert them all to the same palette, using for example Photoshop. Please see the chapter on the palettes, as there are some limitations and a lot of stuff will not look good if you don't respect these rules.

After you have done that, you can start working ! You can either begin with the tileset editor, or the units editor, it doesn’t matter.

3 – The tileset editor

This is TileSEditor.exe. It can save a .tsd file needed for a complete mod. Note that this step is not necessary if you just plan to replace / complete a set of units. This program is used for background gfx (terrain).

a) Loading a tiles bmp

The tiles bmp is the file that will contain all your tiles. The tiles in GpFinalWar are **24** pixels wide and **20** pixels tall. So the tiles bmp has to be a .bmp file, with a height of **20**, and a width dividable by **24**. All your tiles will be side by side in a big bmp file (the max width is 48000, which makes 2000 tiles).

Example: A 240x20 bmp file, so it contains 10 tiles.

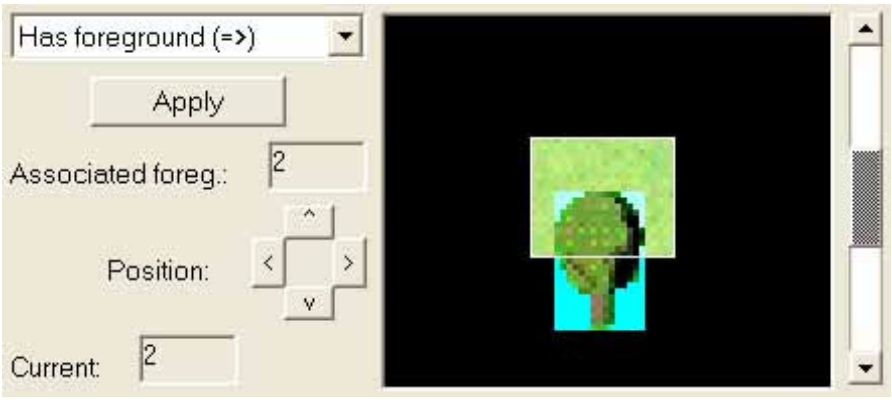
b) Setting tiles' attributes

For each tile, you have to define wether:

- The tile is buildable destructible, which means you can build on this tile, and also, if an explosion occurs onto the tile in-game, it will change to the next tile.
- The tile is buildable, you can build on it.
- The tile is destructible, if an explosion occurs onto the tile in-game, it will change to the next tile.
- Ok, you can go onto it, but not build on it.
- Can't go – you can't even go on it, it is an obstacle.
- The tile has a foreground – it means that if someone is one the tile in-game, the foreground picture will be displayed onto the unit / building.

c) Setting a foreground

To set a foreground, Click on the Load foreground gfx button. Once loaded, move it with the arrow buttons so that both the tile and the foreground match and the good colors show up.

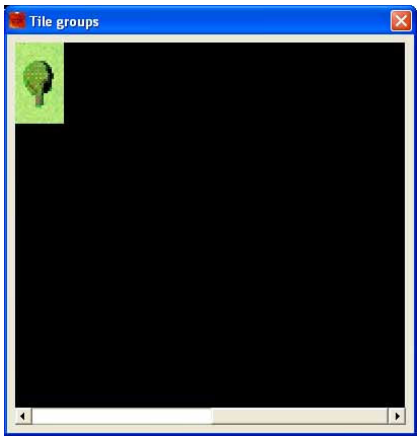


Once both images match, click **APPLY**, and then the foreground becomes associated. You can set the same foreground to more than one tile (but it must have the same position), or load the same foreground picture several times and give it different positions, the foreground will be loaded only once in memory.

d) Tile groups

Tile groups are only used by the map editor. In allows you to define groups of tiles that you will be able to put onto a map in one click later. Hit create group to add a new group. Click on the black box to put the current tile onto it at the right position.

Example:



4 – The units / buildings / scripts editor

This is UnitEd.exe. It can save a .fwd file needed for a complete mod. This is where you get to add new units and set their attributes.

a) Overview

First, you have to know a little bit what how the game works. Units are made of a base, and a turret. They also need more stuff like a in-game menu picture, and attributes.

A turret is composed of **16** pictures, which have a hot spot for a better animation when rotating.

A base is composed of **24** pictures, with hot spot.

Buildings are composed of a background (which will replace the tiles under it), a turret, photos, and attributes. Note that for buildings, it's you who define the number of pictures used as turret (but you must use **16** if this is a turret – we'll see that later).

In case the unit / building can attack, you have to create an attack script and affect it to that unit / building.

In the program, we have 4 windows. The main window is the big – titled one. Here you will edit stuff that will be needed in the unit and building windows. The last one is for the mod's attributes.

b) Add turrets and bases

This can be found on the main window. First click "New turret", to create a new turret object. Then you can load the turret pictures (one by one, or the **16** at once). When you have **16** pictures loaded for current turret the preview gets enabled, and helps to see how the turret rotates. Then, for each picture, select the hot spot by directly clicking on the picture.

If the turret is to be used with a building, you won't always need **16** of them. If it's not a turret you'll need the turret's idle pictures, then the turret's action pictures, then same thing for the broken building pictures. We'll see this later.

The same applies to the bases, except that you will always need **24** of them.

c) Add attack scripts

Same logic here, hit "New script" to create room for your script. A good idea would be to set a default script (the first preferably) and just put a wait into it. This would help avoid confusion when editing units' attributes. Then, you can add your custom scripts. To do that just drag and drop the actions from the right list into the script's action list. Don't forget to put wait actions between other actions. Also for each action, you can set it's value (only for the canon and missile actions – subject to change - the interface will change to show you what's possible anyway) or it's length. You can also define what sound will be used (see the sound chapter), and the turret decay (not always applied – depends on the action).

Last word, but maybe most important: **APPLY** !!! You have to apply when you changed something to an action **BEFORE** you click on another action, or change the script !! Just an habit to take, make sure you take it since the beginning !

d) Buildings

This is maybe the most tricky part. Here you will load the building's background frames, associate a turret, and set the attributes for each building. It is preferable to start with the buildings, because some units will need to be linked to buildings. Note that this is not an obligation, you can always go back and change stuff to any of the edited objects.

New building to create a new object. Then, don't care about the shape combobox, it will be automatically set.

Instead, load your background gfx. Here the rule is simple: they must have one of this sizes:

- 24x20, or 1x1 tile (mostly for turrets)
- 48x40, 2x2 tiles
- 72x40, 3x2 tiles
- 72x60, 3x3 tiles
- 48x20, 2x1 tiles

Contact me if you need me to add more sizes.
This background will have to mix with the "buildable" tiles.

On the building preview window, you have to set the hot spot of the background. As all the backgrounds must have the same size, you only have to set it once for each building. Note that the turret will be displayed exactly on that hot spot.

The background frames **MUST** be put in that order:

- idle frame
- action frames (must make a loop)
- blinking frame (when the building is blinking in game, this frame is displayed)
- idle frame (broken)
- action frames (broken)
- blinking frame (broken)

There must be as many pictures for normal mode as for broken mode.

Then we go for the turret. Just hit "Associate turret" to set the current turret as this building's turret (you can refer several times to the same turret). If the building doesn't need one, just don't associate one and leave 0.

There are 2 sorts of buildings. Attacking buildings (turrets), and the others (those which have a function). An attacking building must have a compliant turret, which means a turret with **16** frames.

The thing is, buildings can be damaged. So you need more frames for the damaged building (as for the background frames of the building, as many pictures for damaged state as for normal state).

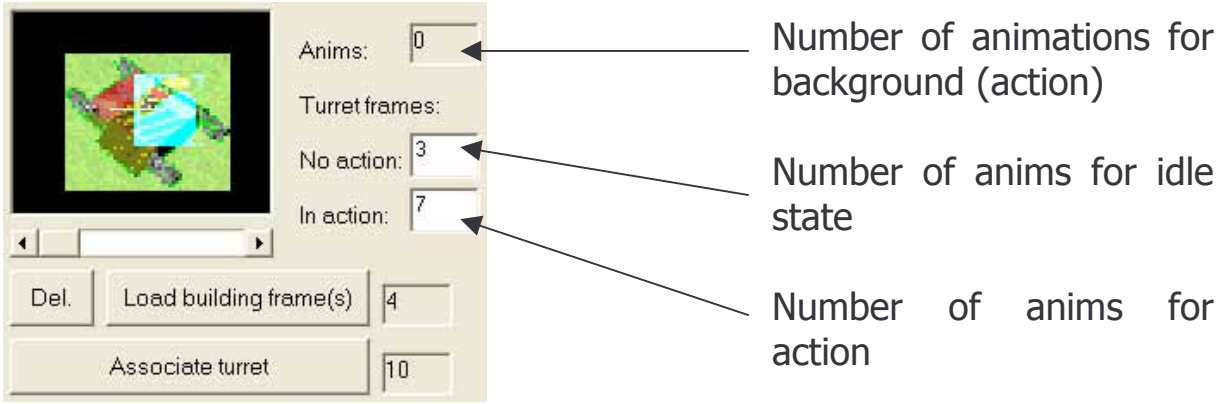
<u>Attacking buildings:</u>

You may find this strange, but you **MUST** enter **14** as "In action" number of turrets if you set a building as attacking (and **0** for idle state). This will enable the game to jump to the next turret bank to get the damaged one (don't ask me why it isn't 16, weird animation stuff which works very well in-game). Note that in this case, you have to put the damaged turret in the next bank of turrets (the one after the normal turret of the same building).

There is a possibility to make a turret which changes when it fires. Contact me, this may force me to add attack scripts.

Other buildings:

These buildings can be animated. That's why it gets a bit more complicated… The editor will ask you 2 things about turret animations:



The number of animations for background only applies to action mode. No animation on background at idle mode can be set. As it only shows the number of additional frames for animation, it is 0 if the building has no background anim in action. 7 means that there are 7 more pictures for animation, so 8 states of animation in total.

The number of animations for idle state corresponds to the number of additional frames of turret for animation when the building is in idle state. 0 means no animation. 7 means 7 pictures **PLUS** the default picture, so the animation will have 8 steps.

For opening buildings (War factories, Nuclear silos…), you can animate both the background and the turret at the same time. To have matching animations, make sure to count the default turret picture and the default anim picture. This means to have the same amount of pictures for both animations, if *Anims* = 3, *In action* = 2.

Same goes for the number of animations for action. 0 means no animation, so the turret will stay in idle state (you'll need one picture only). But be careful, 7 means that the animation will have 7 steps **PLUS** the default building's action state, so 8 steps also here.

When the building is in action, there are 2 cases:

- The building opens, so the anim goes one way (image index increased), then when the building closes the anim goes the other way (decreases). Applies for factories only for now.
- The building cycles. The anim goes all the way increasing, then loops at 0.

After that, you need to set the attributes of the building. Here's the list:

- Armor: the higher, the better. Resistance of the unit.
- Attack: the higher, the better. 0 means no damage dealt.
- Range: in blocks, range limit for attack and view
- Reload time: the lower, the better. If it is too short, the unit may not have enough time to finish her script, which may cause weird stuff, and also freeze the game.
- Select size: size of the selection mask around the unit.
- Power need: power needed by the building. Set to 0 for construction yards (at least the deployable ones). We will see the amount produced by the power plants in the **value** option.
- Function: the role of the building. Air base is for buildings where flying units should land on. You'll define which lands where on each unit directly.
- Value: amount of service provided by the building **(never 0 !!!)**. This is an important value, used for several things. 1st, it is mainly used for the technology limitation, it must be **strictly less** than the limit for the mission or you won't be able to build it. Also, it is used to define a correct order for in-game building. For example, the light construction yard will have a value of 1, the medium one will have 10, the big one 40. Remember that **tech limitation** which is affected by the values, can be set by steps of **5** and up to **45** or **unlimited** in the mission editor. It is important regarding some functions also, as it defines how the building will act:

- For **power plants**, the value is multiplied by the building's energy (which is between 0 and 15000 for the large buildings). So a power plant with a value of 2 (in the first step of tech limitation, so always buildable) and an average width (the width of the objects affects their energy) will produce 9000 x 2 = 18000 electricity points.
- For radars, once their value passes 10, they will display flying units.
- The silo's storage amount equals it's value x 1000. Money limit will increase of this amount.
- For the super weapons, the value lets you choose which weapon will be activated after its **reload time** has passed. At the end of this document there is a list of available values, which should guide you when you'll choose the values and power need for the other

buildings. A wrong value can do weird stuff, but shouldn't freeze the game.
- The value will also affect the efficiency of the resources-bringing building called here derricks (but it can be what you like – as long as you make your own tileset).
- For defense buildings, the value will affect the AI, and tell it the defensive value of the turret.

- Prev. fct. (back to the rest of the building editor): this is the previous function that must be enabled before you can build this building.
- Value (2nd): the value that you must have reached for the *prev. fct.* before you can build this building.
- Deploys / receives at: this is the decay in blocks from the unit's position when it deploys to this building. The building's position is it's upper-left block. Only useful for deployable building, or buildings that will receive units (or create them). These values will depend on the unit's values (must be the same), or will depend on the unit deploy move value or the shape of the building. We'll see that later, moves section.
- Unit out move decay: in pixels this time, this is the number of pixels a unit which would get out of this building would move in x and y. They depend on the *Unit move index* setting.
- Unit move index: the index of the pre-defined move units that would go out of this building would take. See the end of the document to see a list of the pre-defined moves (and their x and y decays). A wrong value can do weird stuff and has a good chance to freeze the game.
- Default team: team that will be able to build this building.
- Price: Price for the building.
- Attack script: index of the attack script associated to this building. Only for attacking buildings.
- Load menu gfx: allows you to choose the building's in-game menu picture, which must be a **42**x**26** .bmp file.
- Targets aerial and both checkboxes: if you check only *Targets aerial*, then this turret won't attack ground units.
- Name: name of the building (displayed in-game when you leave the cursor on it, or when you choose it on the constructions menu.

IMPORTANT: APPLY !!!!!!!!!!!!!!!! Or the changes won't be saved automatically if you change the current building.

e) Units

Units are simpler to set up. The gfx part is very easy. Assign to the unit to right base and turret. If the unit doesn't need a turret, just don't associate one.
Note: flying units will take only 16 pictures for rotation, so in the base editor you'll need to fill the other frames with precedent frames (or you won't be able to associate the base to a unit).

After that, some stuff differ from the building attributes. Here are the differences:

- Speed: speed of the unit. Only between 1 and 3 for now.
- Some functions disappeared. Air attack appeared – set this **only for flying units** ! Flare launcher also, and some other may appear later on.
- Here there is no value for the unit's function. It is the *Prev. fct.*, *Value* setting that is used for tech limit instead.
- Propulsion: type of propulsion of the unit. Don't use infantry yet, as it's not implented. No difference between wheels and chains yet.
- Runs over infantry: not used yet.
- Max fuel: amount of fuel the flying unit will carry (0 – 1024). Corresponds to the number of block position change of the units while moving. Crossing a 200x200 map from left to right would take 200 fuel points.
- Docks to: id of the building which can accept this kind of flying unit; leave it if it doesn't fly.
- Deploys to: id of the building this unit will deploy into. Leave it if you don't want the unit to deploy.
- Deploys at: here we are, the deploy decay. If it is –1;0, then the deployed building's upper left corner would be at one block on the left of the old unit's position. It must be the same for the corresponding building, essentially for the resources-bringing ones. If not, the building will check for resources at the wrong place.
- Deploy move: index of the move the unit will take when you deploy it. See the embedded moves list at then end of this document.

That's it !!! Now you know how to edit buildings and units. But I guess it will take a few tries before they do what you want them to. Go check out the support in case you have questions.

f) Informations

Here’s where you can leave your signature. Fill the name of the mod, of yourself, and of your teams. Don’t forget to give names to the teams, and also set the number of teams you have created.

5 – Other stuff needed

Some additional files will be needed for your mod to be complete. Here's the list:

- a menu.bmp file, 224x160, which will be displayed when the in-game options menu is activated.
- btn0x0y.bmp files. These bmps will contain the additional buttons for the special weapons of each team. x is the number of the team (1 – 4), while y is the index of the picture (1 – normal weapon 1, 2 – pressed weapon 1, 3 – normal weapon 2, 4 – pressed weapon 2). They must be 16x16.
- a menu.mod file, background music for the menus.

6 – Missions editor

You will get to use the mission editor only when your mod is finished – or to create missions for some existing mod.

There are 2 ways to edit the map. Either you use the tile groups you created with the tileset editor (or that were created before), or you can use the box on the upper-right of the map. On this box you can make rectangles to select more than one tile at once.

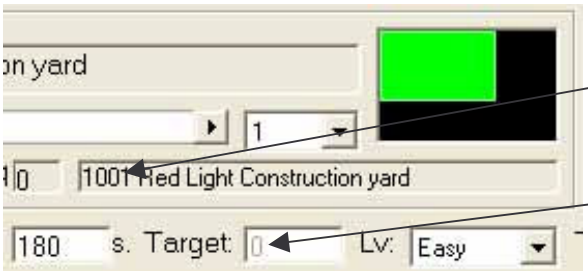
You can resize the map, don't forget to **APPLY**.

A left-click on the map puts objects on it. A right-click on the map deletes them, and if you were putting tiles, the it will reset to a default tile (0).

After that you can save the map as a simple .map file, or add mission informations on it. It consists of 2 things:

- Buildings and units already in place at the beginning of the mission. You select the building / unit you want to put in the map, its team, unit's orientation, and you click on the map to add it. The number of buildings / units should increase.
- The attributes of the mission. Here's the list:

- Type: type of mission. Survival makes the cpu attack at the end of the time specified just on the textbox on the right. Destroy and Timed are classic modes, timed will end by a "mission lost" when time is over. Protect and Target will make the player protect a building / unit or destroy it, in a given time. Here the target textbox lets you enter the id of the building / unit to protect / destroy. Money, Nb units and Nb buildings modes end when the player's dead, or when he reached the amount of money, number of units or buildings defined in the target textbox.



Id of the unit / building under the mouse pointer

Put it there if you want it to be the target of the mission

- Time: time of day the game will start in.
- Weather: choose here the weather condition for the mission. The weather may vary up to this in-game.
- Lv: Cpu difficulty.
- Resources: choose between low, mid and high. This will set the amount of resources that can be found on the map, and also the default money each team has.
- Team: choose here what team this mission has been made for.
- Nb: here you set the number of teams that will fight for the mission. Wrong values here or for the team may make the mission end as soon as it starts.
- Tech. limit: it is where you will set the technological limit for the mission. Only building that have a **VALUE** less than tech. limit, or units that have a *Prev. fct.* **VALUE** less than tech. Limit, can be built.
- Screen start: here you can see where the player's screen will be at the beginning of the mission. Right-click on the overview of the map to set it.
- Briefphrase: this is where you'll enter the briefing of the mission. It is displayed right before the beginning of the mission.

Some important things:

- Before you put a building or unit on the map, check if you gave it the right team number. Note that you can affect a building or unit to any team. It's color will just not be right.
- When you're going to save a mission, be careful: select .MIS as extension !!!!!!! There should be a warning message otherwise.

7 – Campaign editor

This is the last step of the process… You must have all your mission files ready, for all the teams, before you'll use this program. Well maybe you'll use it before to test your missions, but here you will only be able to link missions together, for each team, in the good order.

The campaign name is the title displayed on top of the screen when the briefing and other mission menus appear.

Team choose is displayed, well, when player will choose his team at the beginning of a new campaign.

Then you can add your missions for each team. If you made only 2 teams, no need to put missions in the other teams.

The campaign text is displayed when the player chose his team, just before the 1st mission.

The ending text is displayed when the player finished the last mission for the team.

8 – Sounds and palettes

The sound loading system is pretty basic right now… The game waits for the good sound with the right name, and no other. If a sound is missing, the game stops loading the other sounds.

This means the order of the sounds should be kept, from the **1st** sound to the **26th**. You can replace them, but keep in mind that sounds are **16bit 22kHz mono**, .wav format. Their name starts with *sound01.wav*, up to *sound99.wav*.

After that, you can add up to **73** sounds (**99** in total). Then you can use them in your attack scripts. You can also use some of the default sounds in your attack scripts. Here is the list:

01: explosion 1
02: explosion 2
03: explosion 3
04: menu 1
05: menu 2
06: alert
07: breaking
08: build & sell
09: rain 1
10: rain 2
11: bullet
12: thunder
13: glass
14: zoom
15: building voice
16: on hold voice
17: cancelled voice
18: repairing voice
19: construction complete voice
20: selection voice 1
21: selection voice 2
22: selection voice 3
23: confirm voice 1
24: confirm voice 2
25: confirm voice 3
26: cannon 1

The palette stuff is a bit more complicated. As the game still embeds a lot of gfx objects, there are some limitations for the palette settings.

For now all the explosions and missile objects are still embedded in the engine (and also the user interface, cursors and selection masks). Using another palette than mine will then create a lot of weird coloured objects (each explosion, the cursors…). But there is a way to limit this.

In the archive containing WarGame Maker, there is a file called *default.act*. You can open it with Photoshop. It contains the part of my palette where the colors don't change whether it's day or night. Almost all the embedded objects only use these colors.

a) Setting up your default palette

This is what you should do first. Either you choose to create your own palette and don't care about the embedded objects (and wait till I put them out of the next release and load them from smc). Or you choose that you don't want to replace all the explosions and stuff even if the next version can load them, and then you'll have to mix my default palette with yours.

Here's how you'll do that:

Open Photoshop, and load up some of your unit gfx, terrain and buildings from each team (if your gfx are still in 16bit or more color mode, otherwise your already have a palette – but you can change it anyway, this case just load one of your gfx with a palette). Copy and paste them all to the same picture, and then Picture -> Mode -> Indexed colors

Then a window appears. Select Local (perceptive) Palette. Enter **232** colors, no forced colors, no transparency. Apply, and save the palette you just created (Picture -> Mode -> Color table) as a .act file. Now you have a palette with some room for the default colors. So before you can use your palette, you have to inject the default colors into your palette using PalInject.exe. From the **73nd** to the **96th**, my colors will be injected, and your custom colors will be decayed by 24.

From now on, you can use the new palette created by PalInject.exe. This is this palette you will use to convert your pictures.

b) Day / night cycle – screen flashing

These effects are generated using several palettes in-game. So if you change the palette, you'll have to create as many palettes as the game needs.

The game needs **35** palettes. The in-game's default palette is the number **15**. Their name start with *pal00.act* to *pal34.act*, so *pal15.act* is the default palette (day, no flash). You have to put it in the directory of the mod (where the tileset is).

From **0** to **14**, these palettes are displayed for day / night cycle. So 0 is for the full night palette and should be darker.

From **16** to **34**, these are the palettes for flashes. The **34**th is usually entirely white.

c) Tips

If you want to make a day / night cycle, you should not change the default colors' brightness (at least for the **14** 1st palettes – used for day – night cycle). To have no weird effect using this, you should not use the default colors on your gfx. To do that, open the palette you injected the default colors into, and before you use it to change colors of your gfx, copy it and on the copy set all the default colors to **FF00FF** (flashy pink). Use this palette to convert your gfx.

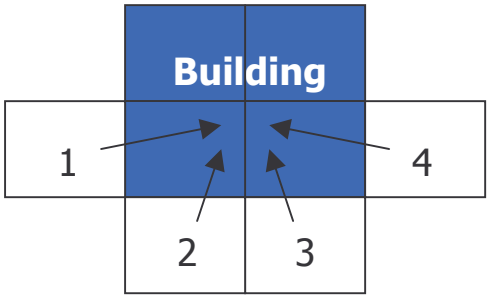
To convert gfx using a pre-defined palette on Photoshop, here's how I do it:

- Open your 16bit or more picture
- Picture -> Mode -> Indexed colors
- For Palette, select Other
- Load your palette with the default colors injected (and reset to FF00FF if you use the day / night cycle)
- Check for the best settings (don't check Keep exact colors)
- Apply

Unit moves, as referred above, are important things that you have to set correctly, or strange things may happen (and possibly reboot of the Gp32 in-game).
Moves are used for 3 things for now:

- Unit deployment to buildings (defined in the unit specs)
- Unit getting out of a war factory (defined in the building specs)
- Unit entering and getting out of a service depot (building)

The service depot will use more than one move, as the unit will enter it, and get out of it. And also from more than one place. This kind of building will need exactly **8** moves. **4** for entering, **4** for exiting, from the left to the ring (through down). So you'll need one move when the unit is on the left of the building. Then one move when the unit is on the left and down…



Same order for the exiting moves, they have to directly follow the entering moves.

This is not really useful right now, as you won't be able to edit scripts. This will be one of the next features. But still it's good to know how the moves are used, as you will have to affect the already existing ones to your mod, to the right units and buildings.

Here's the list:

- 0: unit deploy on the right (-1;0)
- 1: unit deploy on the left (+1;0)
- 2: move out of war factory – right and down (+3;+2) (-48;-40)
- 3: move out of war factory – down (+1;+2) (0;-40)
- 4: move out of war factory – left and down (0,+2) (24;-40)
- 5: service depot (move #1)
- 12: service depot (move #8)

Now things are simple: if this is a unit, it's a deploy move. You want it to deploy on the right, put 0 in the *Deploy move index* textbox. Then put the numbers following in the *Deploy at X* and *Y* textboxes. These numbers are for buildings, to tell them on which block they will work exactly. A derrick, for example, will look for gas on this block. You have to set these coordinates from the upper-left corner of the building (or the building that the unit will deploy into).

If it's a building, 3 cases:

- a unit will deploy into it. So you have to set the same values into the *Deploys / receives at X* and *Y* textboxes than the unit.
- the building is a war factory (will create units). The *Deploys / receives at X* and *Y* will be used to tell the game where the unit will be created (from the upper-left block of the building). Just put the first values of the move you'll use (which you'll put in the *Unit move index* textbox) in the *Deploys / receives at X* and *Y* textboxes. Then put the other values in the *Unit out move decay X* and *Y* textboxes.
- it is a service depot. Set the first index of the service depot moves into the *Unit move index* textbox.

For all the other buildings, the *Deploys / receives at X* and *Y* settings tell the game where the activity of the building will be. So it's useful for derricks (money earning) because this is where the building will look for the resources.

Air bases (airports, helipads or others, and service depots also) will put the unit docked on them on this position on the map. A 1x1 building should not receive units. Also try to avoid setting (0;0) in this case.

Superweapons:

There is not a lot of superweapons available in the game for now. This may change later, as there is room for them in my system.

To define which building will cast which weapon, you will use it's value. This value will refer to a superweapon id, but be careful.

Some superweapon ids are used in-game for big explosions. So only some ids should be used. Here's the list:

- 5: A bomb
- 6: Heavy mortar
- 10: H bomb

Quite short for now, but can easily be increased. And probably there will be another script format for superweapons. We'll see that if there is interest enough, and when explosions will be loaded from disk also.

Bmp2paK.exe:

This program will let you create a Gamepak, file that will contain almost all the small bmps that your project use. This will increase loading speed and upload speed of the mod on the smc.

The program is quite simple. Click Convert, and select your .fwd file. The other files should be placed in the good directories (*tileset.tsd* in yourmod\, your turret pictures in yourmod\turret\ …).

The Gamepak file should be named as yourmod.pak, and placed in the yourmod directory. For example, my mod named *gpfw.fwd* on the smc, is in gp:\gpmm, *tileset.tsd* and *gpfw.pak* in gp:\gpmm\gpfw …

The bonus is, both loading methods can be mixed. So if there is a missing picture in the Gamepak, the game will look for it on the smc.

9 – Future - credits

For now future will depend on how the users will welcome this. So feedback is welcome, see my support forum.

The things I'd like to add, if the program is used, are:

- loading explosions. So you'll be able to add your own effects.
- a move editor.
- maybe a color conversion, if palettes cause too much trouble. This would mean the game would be able to load gfx from 16bit bmps, and convert them directly to a palette correctly. At cost of a very long loading time probably.

Credits:

Well, pretty much me for everything ☺. Also CLONE, for the API calls in VB which made my map editor usable. Also thanks to all the Gp32 community, the most impressive scene I've seen since a long time.