X-Com Mapping Manual v. 0.2

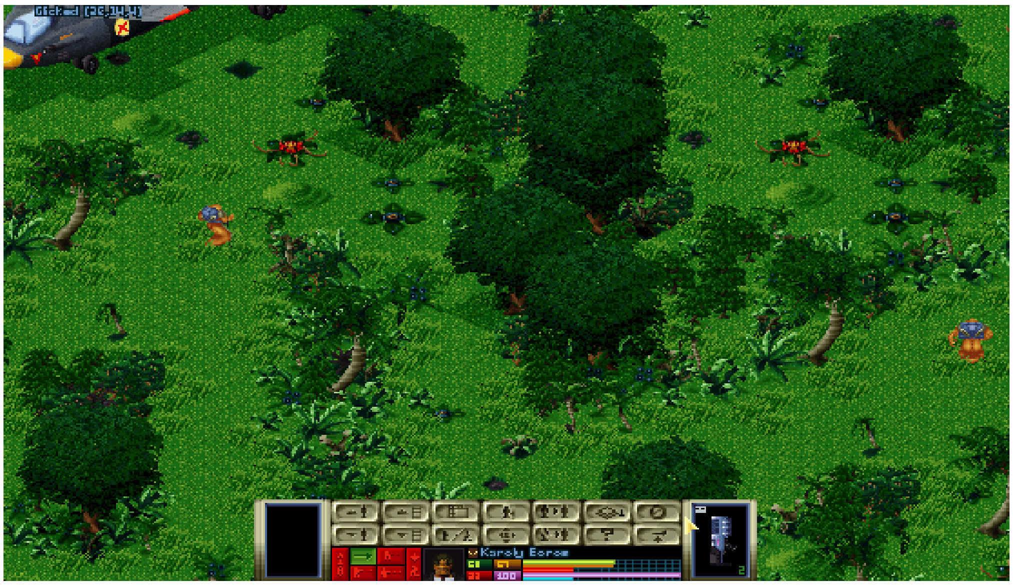
by Solarius Scorch

TERRAIN STRUCTURE OVERVIEW

The data to work with: terrain, maps and routes. These are directly used by the game.

Terrain

A terrain is the highest organizational level of the terrain files and is sometimes referred to as a "biome". Examples include FOREST, URBAN, DESERT or FARM. In other words, everything you see in battlescape belongs (usually) to just one specific terrain, except your craft and the UFO (if present).



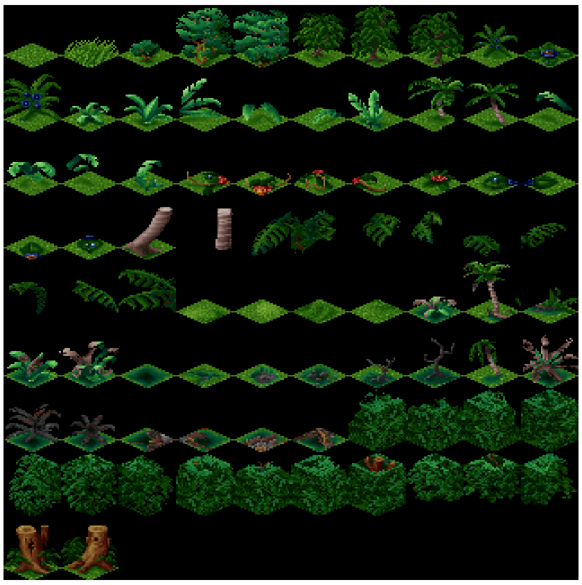
Good old standard JUNGLE terrain from vanilla.

Tilesets

Under the hood, a terrain is composed of so called tilesets. Tilesets are, well, sets of tiles.

Each tileset is composed of three files with the same name and extensions: .pck, .mcd and .tab. The .pck file contains all the graphics used for the tiles. The .mcd contains all tile properties (which graphics is used, can you move through it, does it shine, does it explode, etc.). The .tab is a metadata file, which you shouldn't worry about at all.

For example, the JUNGLE above is only composed of one TILESET. Here is how its .pck file looks like after unpacking:

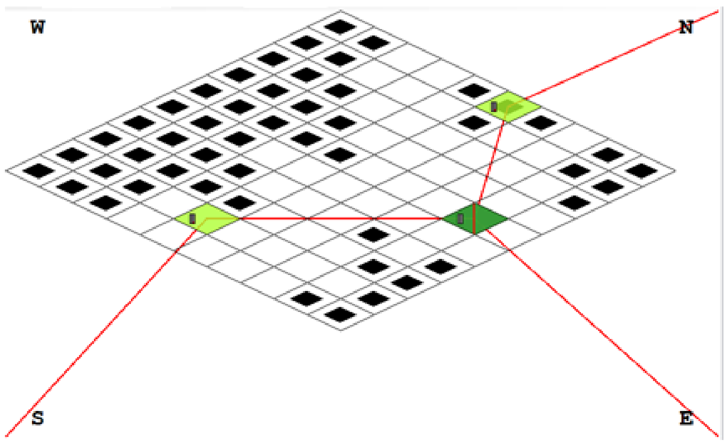

These cute little bitmaps have seen so much blood and killing since 1994!

Maps and Routes

If tilesets are sets of building parts, then maps (also called mapblocks) contain the finished product, assembled from those blocks; like a Lego set. They are stored in .map files, one file per mapblock.

One of many mapblocks for the jungles.

And ROUTES is a companion file to the mapblock which contains information for AI on how to move and where to spawn. It's part of the mapblock really, but for technical reasons it is stored in a separate file with extension .rmp.



AI units move along these lines. At battle start, they can also spawn on the light green node, but not the dark green one.

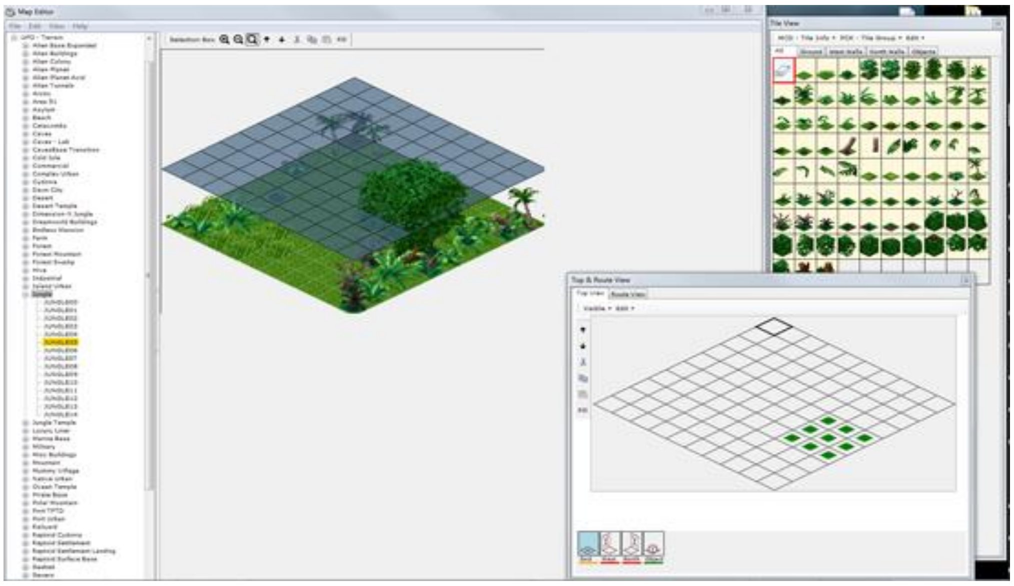
Each mapblock has assigned tilesets it uses. Good news: this means that you can just change what tileset is used for a particular mapblock and it will look completely different! Bad news: it will be a random scramble of game objects, therefore completely useless.

MAPPING SOFTWARE

There are several options to choose from if you want to make new maps for X-Com, but here we will focus on the easiest and most approachable solutions. After all if you had the knowledge necessary to go for something more challenging, you wouldn't be reading this manual, right?

MapView

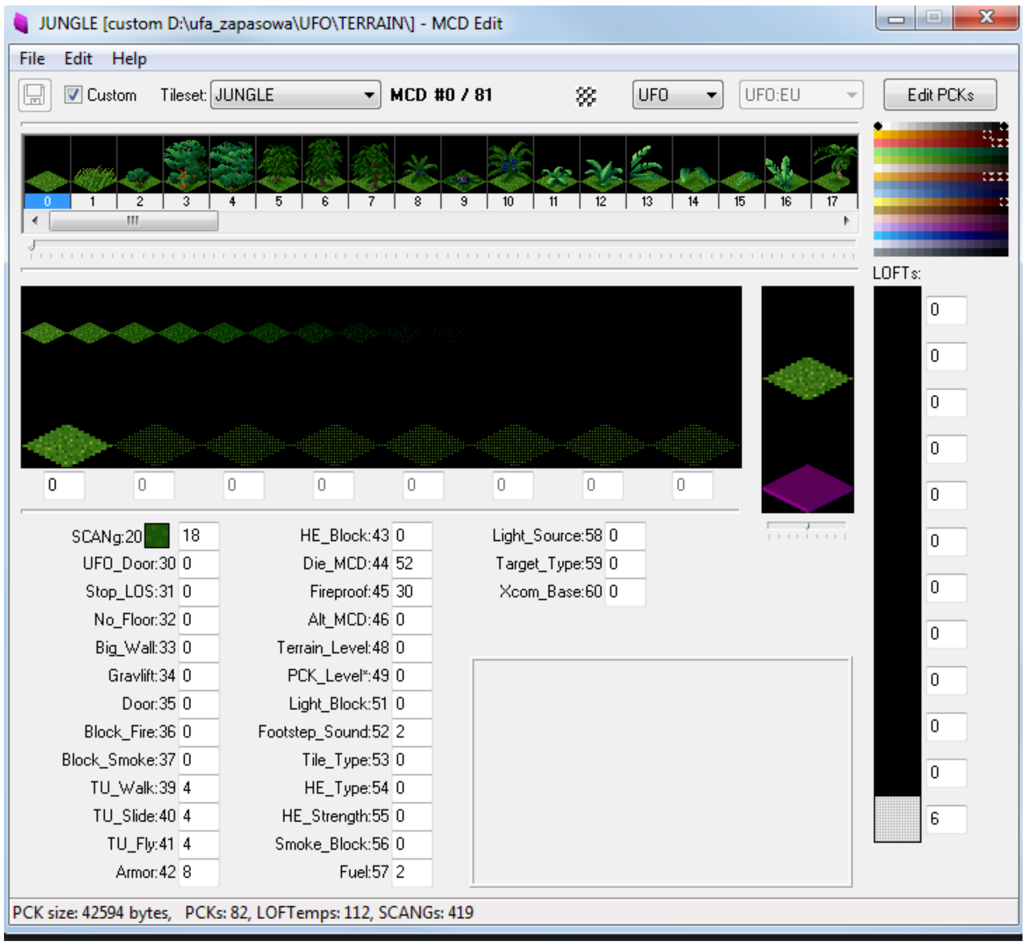
MapView is the program for making mapblocks (including routes). If you just want to make new mapblocks without making or changing any tilesets, this is the only thing you need.



Doesn't look too complicated, right? That's because it isn't!

MCDEdit

MCDEdit is a tileset editor. If you want to make new battlescape graphics, or combine two tilesets into one, or remove half of a tileset, then MCDEdit is your guy.



JUNGLE [custom D:\ufa_zapasowa\UFO\TERRAIN\] - MCD Edit

☑ Custom Tileset: JUNGLE MCD #0 / 81 UFO UFO:EU

0 0 0 0 0 0 0 0

SCANg:20	18	HE_Block:43	0	Light_Source:58	0
UFO_Door:30	0	Die_MCD:44	52	Target_Type:59	0
Stop_LOS:31	0	Fireproof:45	30	Xcom_Base:60	0
No_Floor:32	0	Alt_MCD:46	0		
Big_Wall:33	0	Terrain_Level:48	0		
Gravlift:34	0	PCK_Level*:49	0		
Door:35	0	Light_Block:51	0		
Block_Fire:36	0	Footstep_Sound:52	2		
Block_Smoke:37	0	Tile_Type:53	0		
TU_Walk:39	4	HE_Type:54	0		
TU_Slide:40	4	HE_Strength:55	0		
TU_Fly:41	4	Smoke_Block:56	0		
Armor:42	8	Fuel:57	2		

LOFTs: 0 0 0 0 0 0 0 0 0 0 0 6

PCK size: 42594 bytes, PCKs: 82, LOFTemps: 112, SCANGs: 419

It may not look like it, but it's even simpler than MapView.

SETTING UP THE WORKSHOP

Game Files Setup

First you should make a copy of your game files related to terrains. Sure, you could plug your MapView to work directly with your X-Com/OpenXCom installation, but only truly mad mapmakers would do something like this. (With emphasis on mad.)

First create a new folder, for example:

D:\UFO_terrains\

(You can choose any other path you want, but we'll stick to this one for the purposes of this presentation. If yours is different, always remember to adjust for the filepath.)

Inside this folder, create two folders X-CXom 1 & 2 (or just one of them, if you don't need the other). For the purpose of this manual, we'll assume folder names: UFO and XComTFTD.

Now go to UFO and copy there these folders from your X-Com game:

MAPS
ROUTES
TERRAIN

Then do the same for TFTD, with the same folder names and structure.

Done!

MapView Setup

MapView comes in two variants: 1.0 and 2.0. They both do pretty much the same things and it doesn't really matter which one you pick, just make sure you know which one you have, as they store their internal data in different ways.

Setting up MapView 1.0:

Unpack and copy the program to D:\UFO_terrains\ (same as your UFO and XComTFTD folders).

Then open \MapView\settings. There's a file called Paths.pth. Open it with some text editor (anything more advanced than Notepad will do[1]). Here you define the paths to your files. ${ufo}: should point to your path. Mapdata and images are in the same place as paths, but still need to be defined.

If you use the same paths as our examples, this file should look like this:

```
${tftd}:D:\UFO_terrains\XComTFTD
${ufo}:D:\UFO_terrains\UFO
mapdata:D:\UFO_terrains\MapView\settings\MapEdit.dat
images:D:\UFO_terrains\MapView\settings\Images.dat
useBlanks:false
cursor:${tftd}\UFOGRAPH
```

Now, open MapEdit.dat. At the top you have paths too. Make sure they also match your file location:

```
${varPath2}:D:\UFO_terrains\MapView\BLANKS\TFTD\
${varPath1}:D:\UFO_terrains\XComTFTD\ROUTES\
${varPath5}:D:\UFO_terrains\MapView\BLANKS\UFO\
${varPath3}:D:\UFO_terrains\UFO\MAPS\
${varPath4}:D:\UFO_terrains\UFO\ROUTES\
${varPath0}:D:\UFO_terrains\XComTFTD\MAPS\
```

So far, so good!

Now, time to get serious. Let's have a closer look at MapEdit.dat. It's basically the definition of all mapblocks seen by the program. The upper half is for TFTD, and the lower for X-Com 1 (UFO). (If you don't have TFTD or you are sure you'll never work on TFTD terrains, you can delete the whole section devoted to it as you please.)

Scroll down until Tileset:UFO - Terrain. You can see this part:

type:1

```
    rootpath:${varPath3}
    rmpPath:${varPath4}
```

[1] Notepad might, but I wouldn't trust it with a shopping list, and much less actual data definitions.

```
    blankPath:${varPath5}
    palette:ufo-battle
```

Just leave it alone!

Moving along, you can see the first terrain definition!

```
files:Jungle
        ${varDeps32}:JUNGLE
        JUNGLE00:${varDeps32}
        JUNGLE01:${varDeps32}
        JUNGLE02:${varDeps32}
        JUNGLE03:${varDeps32}
        JUNGLE04:${varDeps32}
        JUNGLE05:${varDeps32}
        JUNGLE06:${varDeps32}
        JUNGLE07:${varDeps32}
        JUNGLE08:${varDeps32}
        JUNGLE09:${varDeps32}
        JUNGLE10:${varDeps32}
        JUNGLE11:${varDeps32}
    end
```

Right, it's our JUNGLE again. But what does this code mean? Well, here's a quick analysis:

files:Jungle - Description of the terrain in MapView's GUI.

${varDeps32}:JUNGLE - This defines the tilesets used; in this case, only JUNGLE.

If you only had one mapblock for this terrain type, you could do something like this:

```
JUNGLE00:JUNGLE
```

Meaning that the mapblock JUNGLE00 uses tileset JUNGLE. But here we have 12 mapblocks (JUNGLE00 to JUNGLE11), so to avoid repetition, we only define the tilesets once as varDeps32, and then simply call for varDeps32 for every mapblock.

Note that each terrain ends with

```
    end
```

and then the next terrain starts.

If this is not clear to you yet, don't worry: you'll get it in practice. For now just make sure everything is as shown.

Apart from MapEdit.dat there is another important file in the same place: Images.dat. All tilesets used in MapEdit.dat must be also declared here, or the program won't see them. (You don't have to do anything with Images.dat, unless you add completely new tilesets.)

Setting up MapView 2.0:

Unpack and copy the program to D:\UFO_terrains\ (same as your UFO and XComTFTD folders).

Then open \MapView2\settings. There's a file called MapResources.yml. Open it with some text editor. Here you define the paths to your files, for UFO and TFTD respectively.

Now, open MapTilesets.yml. It's basically the definition of all mapblocks seen by the program. This file is a YAML file (same data format as used by OpenXCom). The upper half is for TFTD, and the lower for X-Com 1 (UFO). (If you don't have TFTD or you are sure you'll never work on TFTD terrains, you can delete the whole section devoted to it as you please.)

For the example, let's stick with the UFO. The first definition in this section is:

```
- type: JUNGLE00
  terrains:
    - JUNGLE
  category: Jungle
  group: UFO - Terrain
```

Right, it's our JUNGLE again. But what does this code mean? Well, here's a quick analysis:

type: JUNGLE00 - Defines the file name of this mapblock.

terrains: - Defines the tilesets used; in this case, only JUNGLE.

category: Jungle - Defines the section under which this map should appear. doesn't matter for the end result, it's just to make it appear in the right place in MapView.

group: - Similar to the above, but at a higher level; denotes whether it should show up in the UFO category, TFTD category, or some other (custom).

Bear in mind that every block must be declared separately, with all the information above. So the JUNGLE terrain looks like this:

```
- type: JUNGLE00
  terrains:
    - JUNGLE
  category: Jungle
  group: UFO - Terrain
- type: JUNGLE01
  terrains:
    - JUNGLE
  category: Jungle
  group: UFO - Terrain
- type: JUNGLE02
  terrains:
    - JUNGLE
  category: Jungle
  group: UFO - Terrain
- type: JUNGLE03
```

```
    terrains:
      - JUNGLE
    category: Jungle
    group: UFO - Terrain

  - type: JUNGLE04
    terrains:
      - JUNGLE
    category: Jungle
    group: UFO - Terrain
...
```

and so on.

But! Because it's YAML, you can use anchors to make it easier to read and taking up less space:

```
  - type: JUNGLE00
    terrains: &Jungle
      - JUNGLE
    category: Jungle
    group: UFO - Terrain
  - type: JUNGLE01
    terrains: *Jungle
    category: Jungle
    group: UFO - Terrain
  - type: JUNGLE02
    terrains: *Jungle
    category: Jungle
    group: UFO - Terrain
  - type: JUNGLE03
    terrains: *Jungle
    category: Jungle
    group: UFO - Terrain

  - type: JUNGLE04
    terrains: *Jungle
    category: Jungle
    group: UFO - Terrain
...
```

So with &andsometext you can create an anchor (a constant piece of data which keeps coming up), and with *andthesametext you can recall it. It's not specifically a MapView thing, but a YAML thing, so if you need more info, look for general YAML help.

MCDEdit Setup

[UNDER CONSTRUCTION]

MAPMAKING

Creating and Editing

So, let's go build some maps! The JUNGLE terrain is not a bad idea for a start. Let's make a new jungle mapblock.

For MapView 1.0:

First go back to MapEdit.dat and add one more mapblock to the list:

```
files:Jungle
        ${varDeps32}:JUNGLE
        JUNGLE00:${varDeps32}
        JUNGLE01:${varDeps32}
        JUNGLE02:${varDeps32}
        JUNGLE03:${varDeps32}
        JUNGLE04:${varDeps32}
        JUNGLE05:${varDeps32}
        JUNGLE06:${varDeps32}
        JUNGLE07:${varDeps32}
        JUNGLE08:${varDeps32}
        JUNGLE09:${varDeps32}
        JUNGLE10:${varDeps32}
        JUNGLE11:${varDeps32}
        JUNGLE12:${varDeps32}
    end
```

For MapView 2.0:

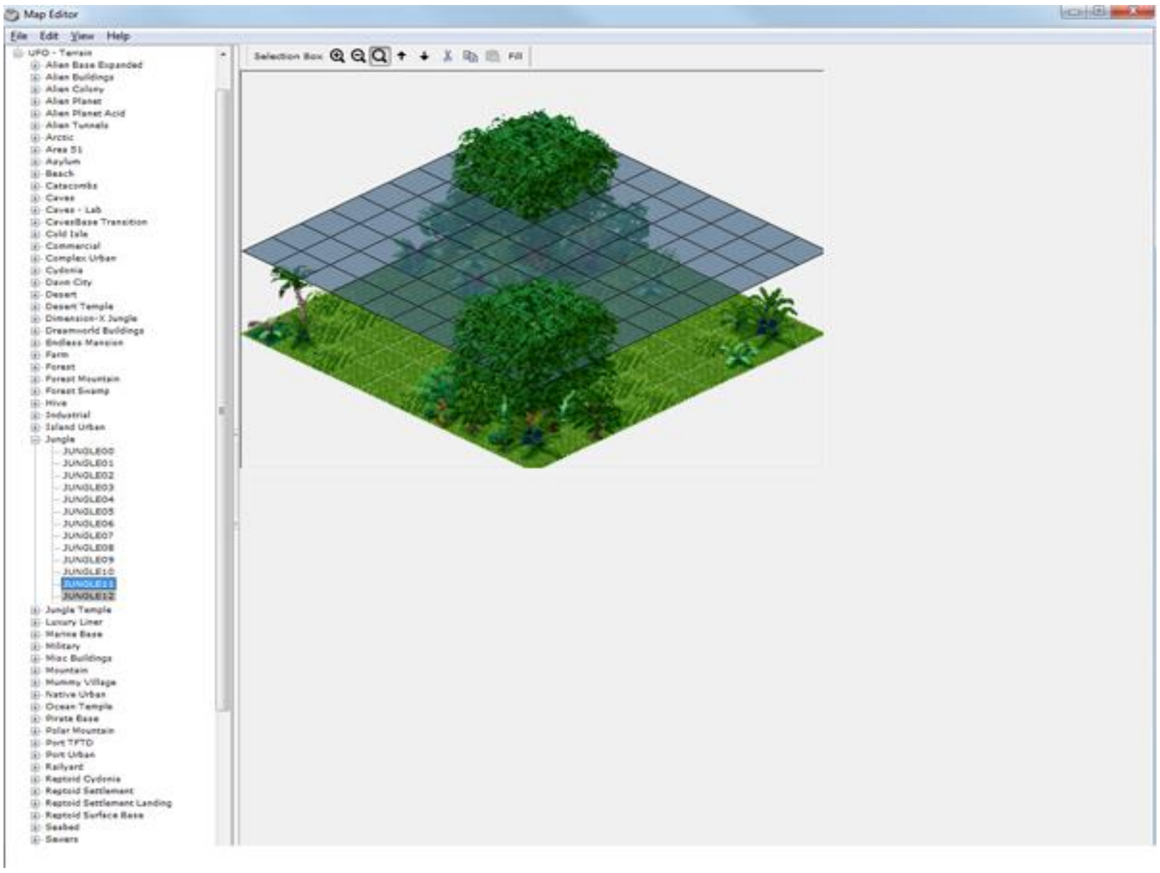
Go to MapTilesets.yml and add one more mapblock to the section:

```
  - type: JUNGLE12
    terrains:
      - JUNGLE
    category: Jungle
    group: UFO - Terrain
```

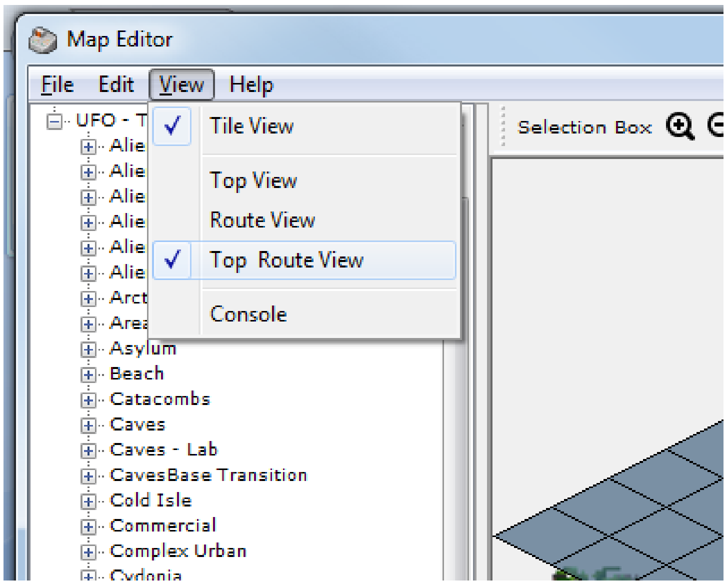
Now, we need the actual .map file. The quickest and dirtiest method (meaning how most modders do it) is taking an existing map and making a copy, then renaming it. (Some would advise to use the internally build engine, but screw that, it's insane. Trust me on that.)

Go to MAPS, take for example JUNGLE11.MAP and rename it to JUNGLE12.MAP. It'll be identical of course, but it is good - you don't have to build anything from scratch, you can just play around.

Now launch MapView and find your map.

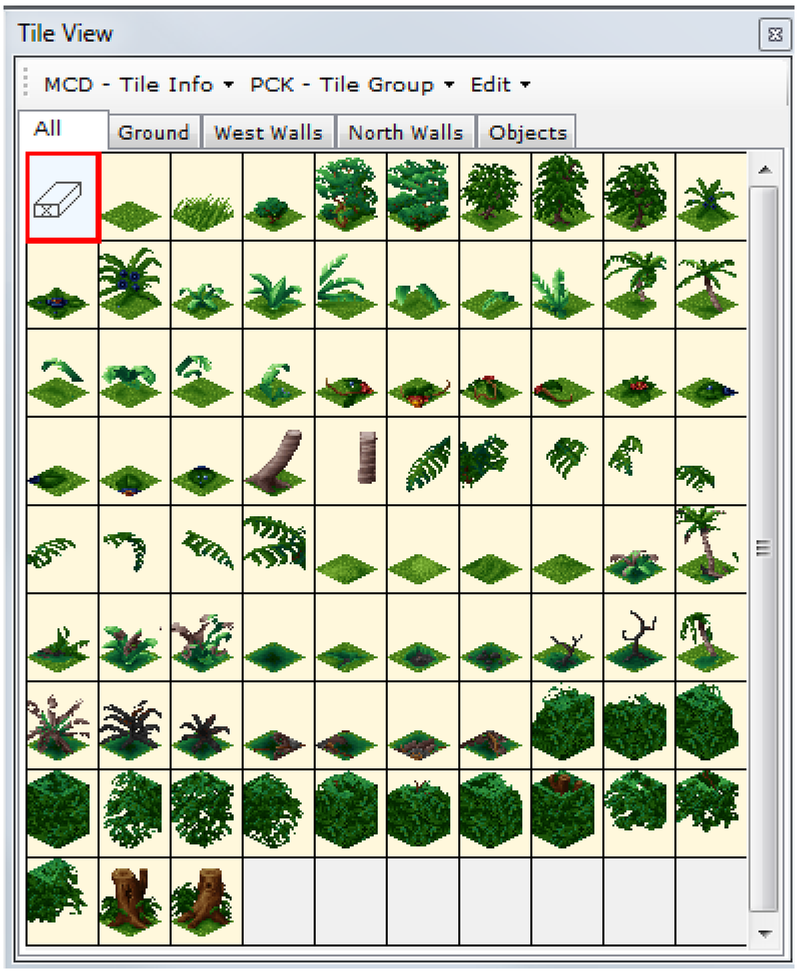


So, you see the map. It's cool for preview, but not very useful for editing.



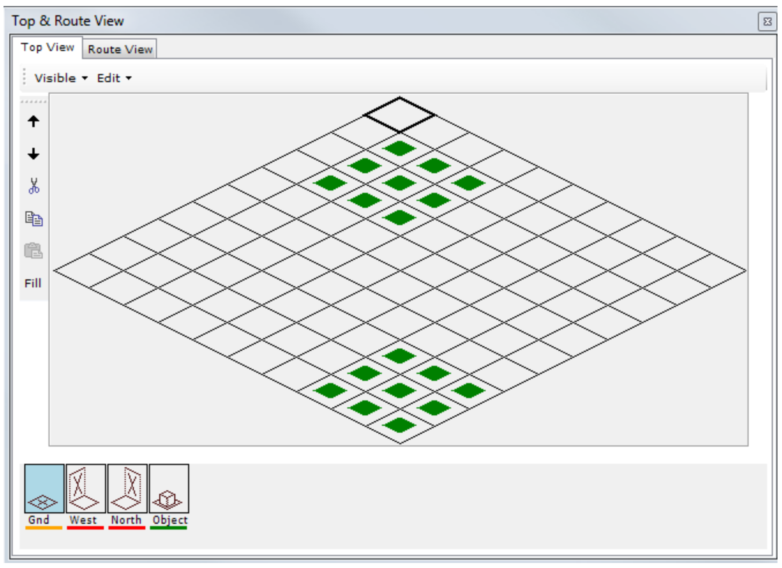
You need Top View and Route View. Or the third option, which is both in separate tabs. Oh, and the Tile View too.

Let's focus on the last one:



These are the building blocks. They are divided into 4 types: ground, west wall, north wall and object. The program conveniently separates them into tabs. (or you can see All, as shown above.) Note that JUNGLE has (unsurprisingly) no walls at all.

Now, if you check Top View, it shows two things:
1) A simplified map that is actually easy to use (if not to read),
2) 4 boxes.



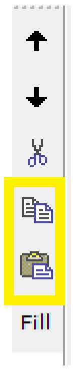
These four boxes correspond to the four tile types: floor goes to Gnd, west walls go to West, north walls go to North, and objects go to Object.

If you click on any tile, you can see what ground/walls/objects are placed there (if any). Note: the program normally starts displaying the very top level, but you can switch between levels using arrows on the left.

This is where actual mapmaking starts: you can add stuff by selecting it in the Tile View and then right-clicking in the correct box in Top View! The map element will be placed on the map. But remember the most important rule: even though you can place a tile in a different box type (like ground in the Object box), never do that, ~~unless you know well what you are doing~~.

If you want to remove something, right-double-click it.

You can also select a tile and copypaste it in another place, using these obviously looking buttons:



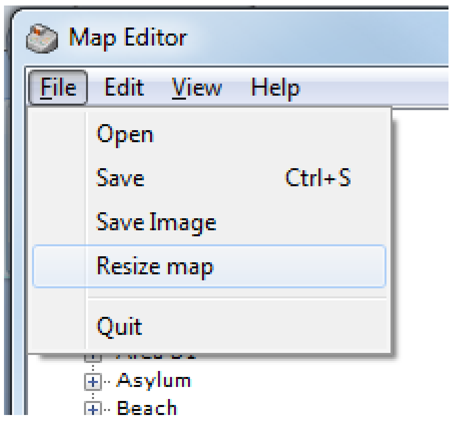
Also, Fill is very useful - it applies your last selection to a marked area.

You can also drag and select part of the map and then paste it somewhere.

Resizing Maps

A mapblock has three dimensions: width, depth, and height. Width and depth must be in full 10s: 10, 20, 30 etc. In practice, 30 and bigger are extremely rare. For terrain mapblocks (not UFOs), they should be squares (same x and y), unless you're willing to do some advanced mapScript magic.

If needed, you can change map dimensions here:



Laying Routes

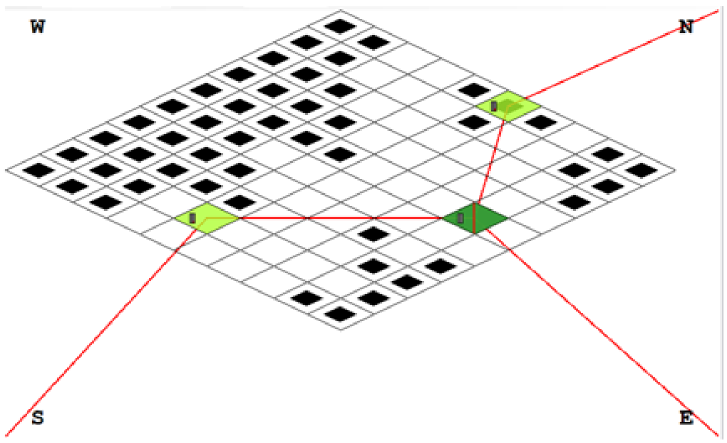
Nodes And Links

Nodes are specific points on the map used by the AI to navigate around the map. To put things simply, an AI unit each turn chooses a node to go to and tries to reach it by following a network of links.

Nodes must be connected with links to tell the AI how to move between them. They must be sent manually, as in “node X will connect to node Y. Also node Y will connect to node X.” Yes, this means that you can make one-way paths for the AI, if you so desire.

All these considerations only apply to “patrolling” units, roughly meaning those unaware of any enemies. Once they spot your guys, they stop caring about routes and employ tactical movement instead. Once they are no longer aware of your forces, they go back to patrolling routes.

To create a node, simply right-click on the nodes map. You can also use Copy, Paste and Delete to edit nodes quickly. (Copy only transfers patrol data and spawn data, not links.)



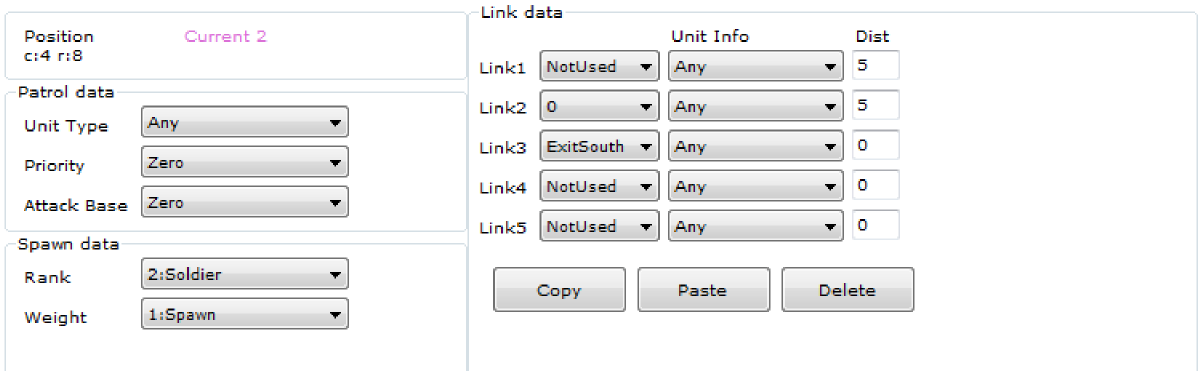
A simple routes map with 3 nodes. Links connecting them are clearly visible. Some reach lead beyond the map, to other mapblocks.

Spawns

Any node may also function as a spawning point. This only applies to battle start. Enemy units and civilians (and in some cases also player units) only spawn on appropriate nodes, unless there aren’t enough nodes on the whole battlescape.

Generic spawn points work for any non-player units, but they can be restricted by unit size or rank - see below.

Interface



Position c:4 r:8 — Current 2

Patrol data

Unit Type	Any
Priority	Zero
Attack Base	Zero

Spawn data

Rank	2:Soldier
Weight	1:Spawn

Link data

		Unit Info	Dist
Link1	NotUsed	Any	5
Link2	0	Any	5
Link3	ExitSouth	Any	0
Link4	NotUsed	Any	0
Link5	NotUsed	Any	0

Patrol Data:

Unit Type: here you can restrict the unit by choosing: Any, Small, Large, Flying, or Flying Large. The most common reason to use it is to prevent spawning large units in cramped rooms with no big exit, or non-flying units on rooftops with no access.

Priority: allows to determine how tactically important this node is. The higher the number, the more likely it will be for AI units to pay it a visit. You can increase this number for areas which you think should be closely guarded.

Attack Base: It's currently unclear if any modder has ever used this option, or even knows what it does. It is most likely applicable only to X-Com bases and helps the AI find targets to be destroyed.

Spawn Data:

Rank: you can choose to only spawn AI units of a certain rank (as determined by their position in the alienRaces section). This is usually used to ensure that engineers are spawned near engines, commanders on bridges, etc.

Weight: disregarding other considerations, the game first populates all spawns with weight 10, then all nodes with weight 9, etc. This is not exact science though — don't count on it too much!

Link Data:

In this section you create links between the nodes. Without links, an AI unit will just stand around doing nothing, as they have no path to patrol. (Remember that this goes out of the window after the unit spots a live target to engage.)

Each node can have up to 5 links to other nodes, or map edges. You form the connection by selecting the other node from a drop-down menu or right-clicking it. You can also restrict a link to a particular unit type (see: Unit Type above).