MAPMAKING AND SCENARIO DESIGN FOR CIVILIZATION IV

v1.0 (based on Civilization IV v100, BMPtoWBS Converter v186)
By Gabriele "Rhye" Trovato

Index

1 Introduction

1.1 Scope of this document

This tutorial will explain how to design Civilization IV maps and scenarios using just external applications.

Through an example, it will be shown how to produce the source pictures, how to convert the bitmaps into a Civilization IV map, and how to edit a simple python file to store all the scenario settings. This tutorial isn't meant to explain programming in python, as its scope ends within the limits of the WBS (short form of Civ4WorldBuilderSave, the basic Civ4 scenario extension) and of the basic structure of a mod.

To understands the topics that will be mentioned, there is no particular knowledge needed. Just the patience that photo-editing and programming usually require.

1.2 Why external applications only?

Civilization IV offers a nice in-built utility: the World Builder, which loads and saves files in .Civ4WorldBuilderSave format.

However, there is one main reason that makes external programs more efficient: the fact that when you draw a map and start placing resources and cities on it, you can't turn back.
Let's suppose that while placing the cities, you realize that one continent should be shifted one tile left (with all the cities, units and resources in it)…..good luck.

Using the BMPtoWBSConverter tool instead, you can turn back anytime and solve the problems with quick photo-editing or line-editing.

2 BMP to WBS Converter

2.1 How to execute the application

In order to run the application, you must have both Python 2.4 (http://www.python.org/2.4.1) and PyPIL (http://www.python.org/pypi/PIL/1.1.5) installed.

Then, you can unzip the files of the converter into the folder where you have Python installed (it could be C:\Program Files\Python24\ ).
The zip contains the converter itself, a python file that acts as a template (EmptyScenario.py), another Python file, used for the example that's being shown (DefaultScenario.py), and some default bitmaps.
To run the application, you just have to double click on the .py file corresponding to your scenario, and it will automatically launch the actual converter. For example, if you execute EmptyScenario, it will create an empty map. So, when editing your own scenario, just make a copy of the template file and rename it.

So, is some knowledge of Python needed?
The answer is no, at least for changing the parameters of the applications.

The parameters can be changed editing the scenario file, right clicking on it and selecting "**Edit with IDLE**"

2.2 class ScenarioClass

Now, let's see what the file contains.
Ignore the first lines, and the very last line.

The class called **`ScenarioClass`** contains all the parameters you need to change that will be written in the WBS.
Here follows the description of all the variables you need to know.

2.2.1 Member objects

- **`iDefaultWidth`** is, as its name says, the width of the map we are going to create.
- **`iDefaultHeight`** is the height of the map.
- **`iDefaultRiverSensitivity`** is the parameter that regulates the way rivers are placed by the converter during the resizing process. Its value goes from 1 to 7: higher means fewer river plots are put on the map based upon the river map - 5 is recommended. We'll speak again about this later.
- **`iDefaultCoastsAssignment`** is the flag that decides how ocean plots may be turned to coast:
  0 means that no ocean plot will be turned to coast
  1 means that the pixels manually coloured of blue will correspond to coast (in fact, you might deliberately leave some plots next to the land as ocean, to obstruct the passage of galleys)
  2 means that every plot next to some land will be automatically turned to coast
  3 contains 1 and 2: every tile next to the land or corresponding to a blue pixel will be turned to coast
- **`iDefaultFloodPlainsAssignment`** works in the same way as the coasts assignment flag:
  0 for no flood plains
  1 for manual placement (placing light green pixels). You may not want a river through a rocky mountain to be a fertile land.
  2 for automatic placement: flood plain will be added in every desert plot next to a river
  3 for manual + automatic
  Please note that manual green areas of the bitmap are filtered by the converter, which deletes any flood plain assigned to a plot which either isn't desert or isn't close to a river.
- **`iDefaultXWrapping:`** set it to 1 for a round world, 0 for a flat world.
- **`iDefaultYWrapping:`** same as above: if set to 1, you can cross the top or the bottom of the map.
- **`iDefaultNumberOfCivs`** obviously, contains the number of civs set by default.
- **`HeightMapPath`** is a string that contains the path of the height map, the map where sea and relieves are shown.
- **`TerrainMapPath`** contains the path of the terrain map, that shows where desert, grassland, plains, tundra and ice are.
- **`FeatureMapPath`** contains the path of the feature map, which indicates jungle, forest, sea-ice, oasis, flood plains and fallout.
- **`RiverMapPath`** contains the path of the river map
- **`BonusMapPath`** contains the path of the map that contains the placement of the resources
- **`dltRevealPlots[#] = []`** is used to set the fog of war. It contains the list of the plots revealed to a certain player. The first couple of square brackets contains the index

of the civ (from 0 to `iDefaultNumberOfCivs` – 1). The second brackets contain the list of the coordinates, in the format `(X1, Y1), (X2, Y2), (X3, Y3),...` Remember that the last set of coordinates shouldn't be followed by a comma.

- Terrain improvements work in the same way. You have to fill the space between the brackets corresponding to the following variables:

```
dltFBoats = []
dltWBoats = []
dltMine = []
dltQuarry = []
dltCamp = []
dltFarm = []
dltPlantation = []
dltWinery = []
dltPasture = []
dltWindmill = []
dltLumbermill = []
dltWatermill = []
dltFort = []
dltOPlatform = []
dltWell = []
dltWorkshop = []
dltCottage = []
dltHamlet = []
dltVillage = []
dltTown = []
dltRuins = []
dltGoodyHut = []
dltRoad = []
dltRailroad = []
```

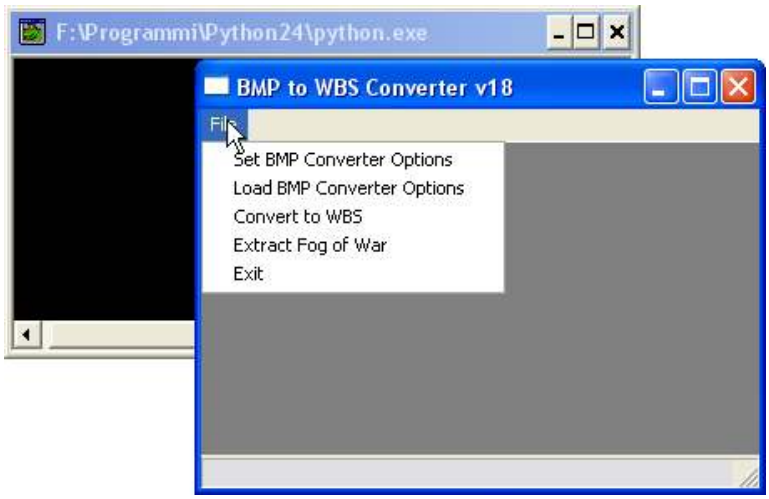
2.2.2 Member functions

- **`TerrainChanges`** contains last minute changes to the map. If your scenario needs a particular change but you don't want to edit the bitmaps (because you need them for other uses and you want to keep just one copy), you can edit the map here.
- **`WriteStartingPlots`** writes `StartingPlot` in the plot info. A civilization will start in that location.
- **`WriteCitiesUnits`** stores the list of the plots that contain cities or units.
- **`getExtraWBSText_GameSettings`** is the list of the general game settings, such as victory types, initial game turn, type of calendar, start year, or the description of the scenario.

- **`getExtraWBSText_Diplomacy`** contains team info. That is, the list of technologies discovered by each civ, and their war/peace relationships.
- **`getExtraWBSText_Civs`** stores the basic settings for each civilization. That includes civilization and leader name, flag decal, starting gold, starting civics, a city list, handicap settings and much more.
- **`getExtraWBSText_WorldSettings:`** contains the basic world settings such as size or sealevel.

2.3 Launching the Converter

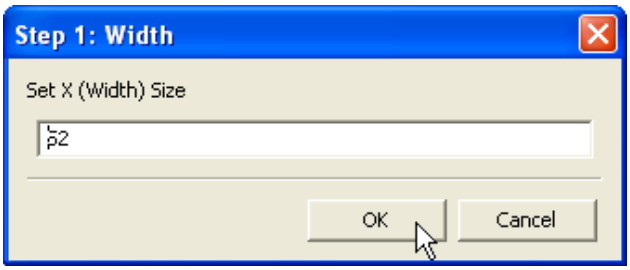
Now let's see how the converter works.
When you launch EmptyScenario.py, two windows open:



The background window contains the standard output. It will show some log messages.
The foreground window is the main application and offers some options we'll examine now.

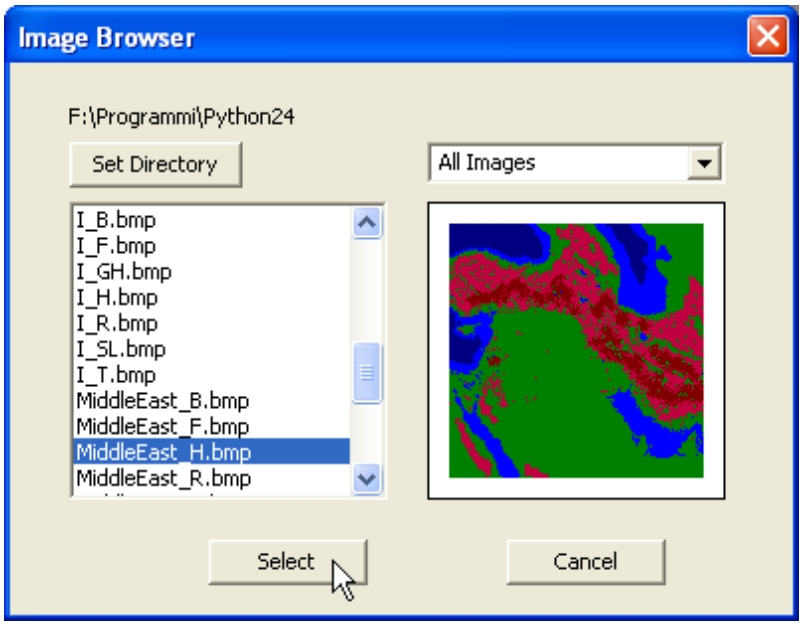
2.3.1 Map setup

If you click "Set BMP Converter Options" some dialogues will open.



You can enter the values or leave the default ones (which are taken from the scenario file, in this case EmptyScenario.py), and click OK.

The Converter will ask the map size, river sensitivity, coast and flood plains assignment, wrapping settings, number of players and the path of the bitmaps.
The image browser will let you choose the five bitmaps (height, terrain, feature, river, bonus).



Actually, only the first is required to generate a map. You can just select the heights map, and press Cancel for the others.
In this example, we'll choose Empty.bmp for all the maps. It is a plain green bitmap, which therefore contains no data.

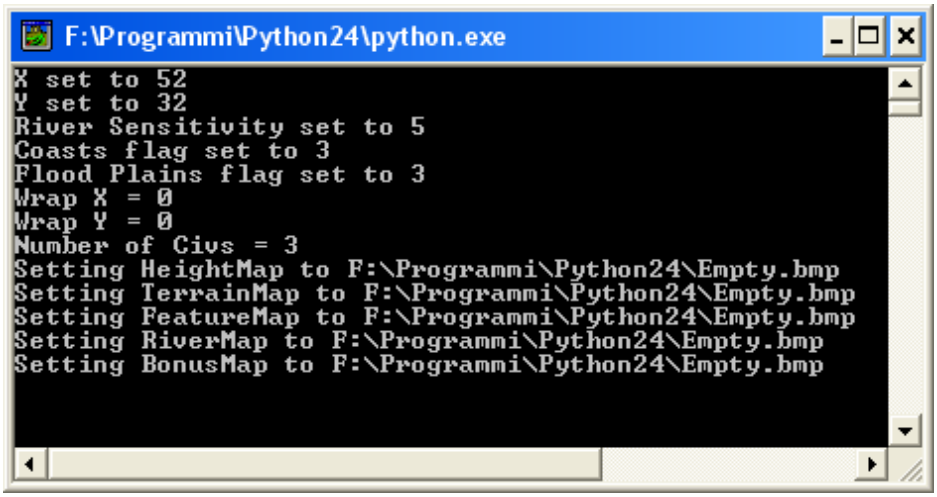
2.3.2 Loading map settings

When you've stored your data in the scenario file, you don't actually need to go through the map setup.

```
iDefaultWidth = 52
iDefaultHeight = 32
iDefaultRiverSensitivity = 5
iDefaultCoastsAssignment = 3
iDefaultFloodPlainsAssignment = 3
iDefaultXWrapping = 0
iDefaultYWrapping = 0
iDefaultNumberOfCivs = 3
HeightMapPath = "F:\Programmy\Python24\Empty.bmp"
TerrainMapPath = "F:\Programmy\Python24\Empty.bmp"
```

```
FeatureMapPath = "F:\Programmy\Python24\Empty.bmp"
RiverMapPath = "F:\Programmy\Python24\Empty.bmp"
BonusMapPath = "F:\Programmy\Python24\Empty.bmp"
```

Having stored these settings, you can save time if you just load them through the second option of the Converter menu.
The output shown will be the same as if you'd have chosen the settings manually.



2.3.3 Converting the map

After the settings setup (in either the way you've chosen), the last step is to select the destination file and click on "Convert to WBS".
If everything worked, you should see "Done" written in the standard output window. An external debug log is also produced: it is DebugLog.txt.

2.3.4 Other options

"Extract Fog of War" is a sort of stand-alone option. It converts the coordinates of a coloured zone from a bitmap to the proper format, so that can be used with **`dltRevealPlots[#].`**
This aspect will be examined later.

Click "Exit" to quit the application.

3 The map

Civilization IV maps are very different from their two latest predecessors.
The biggest difference is that the plots of Civ4 will be plain squares, rather than the diamond view we had with Civ3 and Civ2.
A consequence of this is that the area is calculated simply **width*height**, rather than **(width*height) / 2**.
This means that Civ4 maps may look smaller, while they actually aren't.
A Civ3 160x160 map for example had 12500 plots. A Civ4 160x160 map will have 25600 plot. So, the Civ4 equivalent of the Civ3 map would be 112x112 (12544 plots).

Please note that width and height in most of the cases won't be equal now. Due to the isometric view, Civ3 tiles were composed by 128x64 pixels, so a squared map ended up looking rectangular. Without this oddity, you're free to choose any width/height ratio in Civ4.
A cylindrical map such as the Earth will be best represented by a rectangle, rather than a square.

3.1 Calculating the size

Before creating a map, it's important to calculate the final size of the map first.
Standard Civ4 settings might help:

Number of players	Width		Height
3	52	x	32
5	64	x	40
7	84	x	52
9	104	x	64
11	128	x	80

Keep in mind that the proportion should depend on number of settleable plots rather than the pure area. Water percentage is important here to make the map appropriate to the number of players.

3.2 Photo-editing

Now, let's begin creating a sample map.
It's possible to use MS Paint, but a better application is highly recommended. Even a freeware or shareware product may make this work much faster and easier.

When you'll save your images, remember that:

- File extension must be .BMP (Bitmap for Windows)
- colour depth must be 24 bit (16 million colours)

3.2.1 Source bitmap

We'll use this fictional map.

Our goal is a 35x26 map, with 3 players.

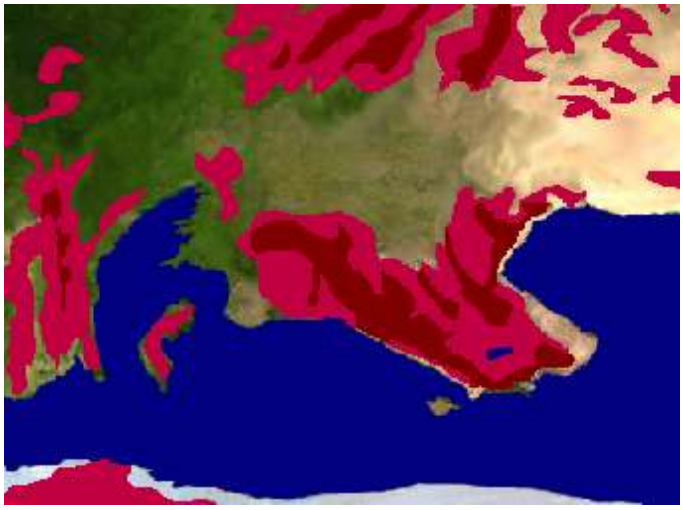
3.2.2 Height Map

To create the height map, you have to divide the source bitmap in five zones, corresponding to the following colours:

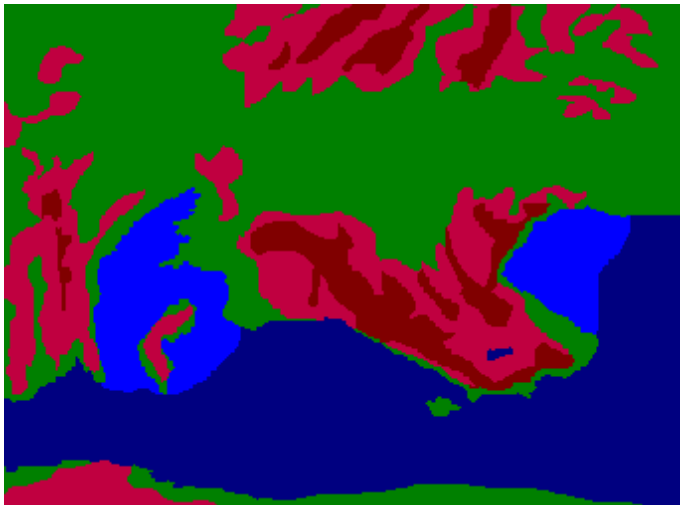
The final map heights will vary depending on the colour they match. It's important that the RGB (red/green/blue) combination is the same as the one that you get from the above key or from this list:

```
Peak        = (128, 0, 0)
Hill        = (192, 0, 64)
Valley      = (0, 128, 0)
Coast       = (0,  0, 255)
Ocean       = (0,  0, 128)
```

First, let's colour the sea dark blue, then the mountains rust (keeping in mind that they will be impassable in the game), then the other minor relieves.



The rest of the land can be replaced with green.
We are planning to set `iDefaultCoastsAssignment` to 3: that is, we are replacing with lighter blue the areas where we want coasts (the gulfs), but we're leaving the land borders to the automatic algorithm.



Now, we can save the map as Default_H.BMP.

3.2.3 Terrain Map

This time, this is the key:

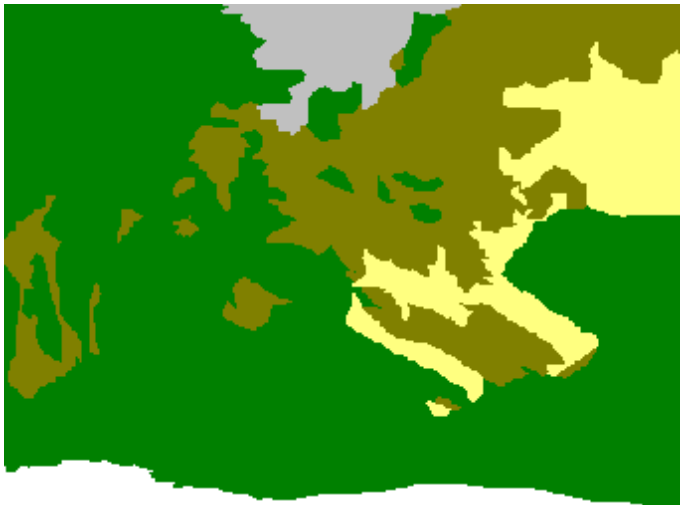

```
Desert     = (255, 255, 128)
Plains     = (128, 128, 0)
Grassland  = (0, 128, 0)
Tundra     = (192, 192, 192)
Ice        = (255, 255, 255)
```

Start painting desert (you can recognize it from the satellite photos by its lighter yellow) and plains (darker yellow), then tundra and ice.

Try to form jagged zones.

When you've done, replace everything else with green. Including the sea. In fact, there is no need for a specific colour for it. Sea zones could be coloured with desert or plains: the result won't change, as they're covered with water. In fact, notice the fact that the lake has been replaced with plains instead (so that, if it is deleted from the heights map, it will match the surrounding terrain).

Then save the image as Default_T.BMP.



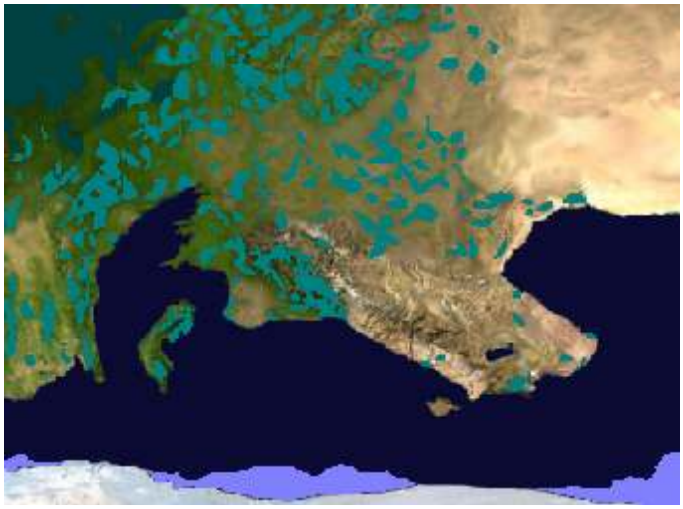
3.2.4 Feature Map

This is the feature key:

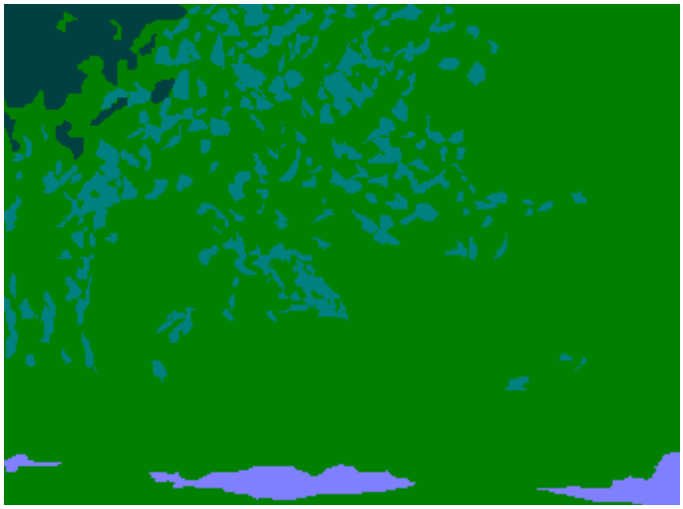

```
Sea Ice          = (128, 128, 255)
Forest           = (0, 128, 128)
Jungle           = (0, 64, 64)
Oasis            = (192, 255, 255)
Flood Plains     = (192, 255, 0)
Fallout          = (32, 64, 0)
```

Place forests and jungle. Usually you can recognize them by a darker green. Try to place them (especially forests) the most scattered as possible.

When you've finished this, select a part of sea and replace it with the sea-ice colour (light blue). Keep in mind that those areas represent frozen waters, and will be impassable to all the ships.



Then, replace everything else with green, and save the image as Default_F.BMP.:



You don't need to place oasis, flood plains or fallout for now. You can do it later.

3.3 Intermediate result

Now here comes a turning point.
These 3 pictures can be used now to create a first draft of the map.

The Converter is able to resize the bitmaps from 340x252 to 35x26 and uses an algorithm to decide what the final terrains would be.

This is the intermediate result if we try to convert now:

Please notice that coasts have been added in the gulf, and all around the land borders.
But apart from that, as you can see, much data has been lost, in particular ice, forests and the lake.

Now you have two choices:

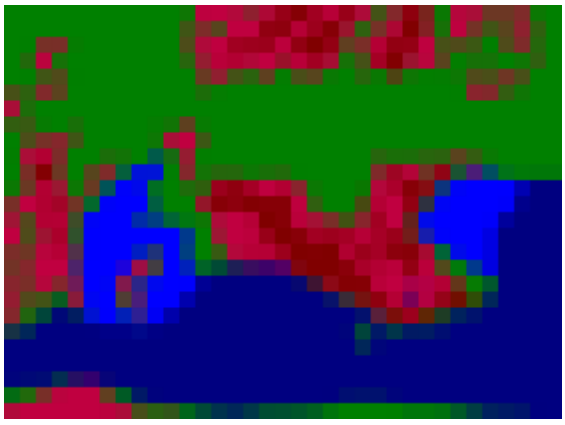
- Stop reading now (and skip to the next chapter), and place rivers, oasis, resources and the latest corrections from the World Builder
- Keep reading. We'll resize manually the bitmaps we've made so far and complete the set. In this way you'll preserve the flexibility mentioned in paragraph 1.2 and you'll have a better control on how resizing is done.

3.4 Resizing

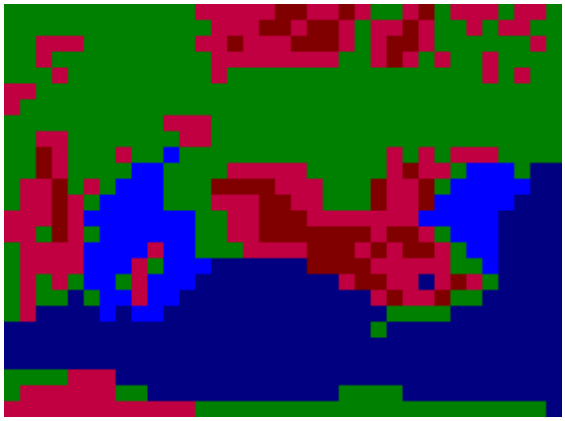
Resizing algorithms remove or add pixels as necessary to achieve the new height and width specified for the graphic.
A good image editor allows the user to choose the algorithm. Full products such as Corel Photo-Paint, or shareware applications such as Paint Shop Pro, or freeware such as VCW VicMan's Photo Editor allow this.

Two main algorithms use anti-aliasing (**Bicubic** and **Bilinear** resampling), while **Pixel Size** (also sometimes called Nearest Neighbour) doesn't.
With the former, edges will be slightly blurry due to the antialiasing that went on during the resize; with the latter, edges will be clear and sharp.

You must use Pixel Size, because a blurry image would alter the palettes. As you can see from this 35x26 image below (zoomed in 800%), some new colours have been added to attenuate the edges. Colours that aren't in the Converter pallete.



So, remember to turn any anti-aliasing off, and after having made a backup of the three images used so far, resize them to 35x26 (this size is maintaining the aspect ratio of the originals).



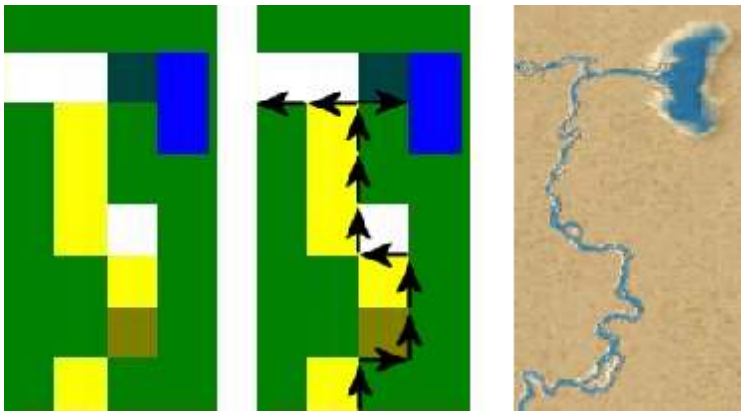
3.4.1 River Map

Make a backup copy of the heights bitmap and rename it DEFAULT_R.BMP. Now, you can place rivers on it.
Like in Civ3, rivers stand in the middle of two plots. Civilization IV format stores river info in the plots that have a river flowing at their right-hand side or at their bottom. River info includes the flowing direction.
There's no info for rivers flowing at their top or left-hand side. So, if a river should flow at the top of a plot, the info will be contained in the adjacent (above) plot, which has the river at its bottom.
The Converter follows this key:

```
From South-East to North-East                    = (255, 255, 0)
From North-East to South-East                    = (255, 0, 0)
From South-West to South-East                    = (0, 64, 64)
From South-East to South-West                    = (255, 255, 255)
From South-West to South-East and North-East     = (128, 128, 0)
From North-East to South-East and South-West     = (255, 128, 128)
From South-East to South-West and North-East     = (255, 255, 128)
From North-East and South-West to South-East     = (128, 64, 64)
```

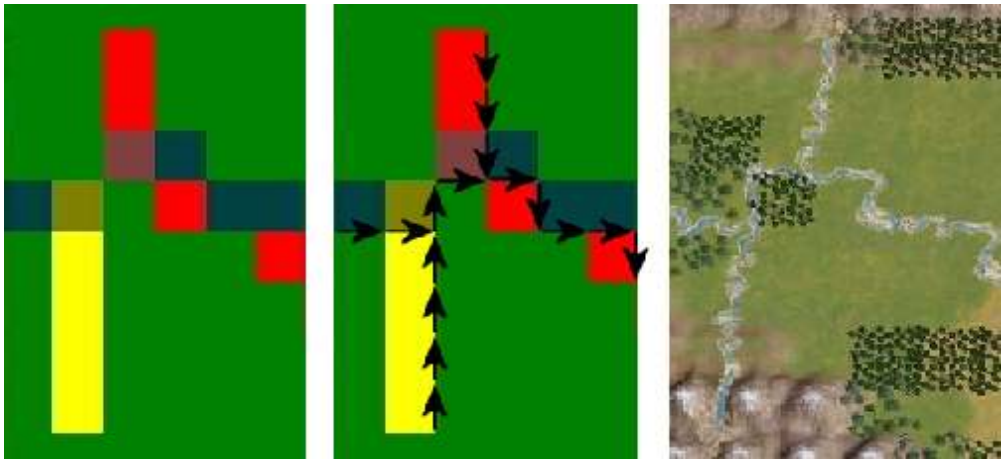
Now let's see some examples:

- Example 1 – A mouth



As you can see, you just need to trace the flow, basing on the key. In this example the river splits, with the right path leading to a lake: a further greyish blue pixel upon the lake isn't necessary, as it would make the river touch the opposite shore of the lake.

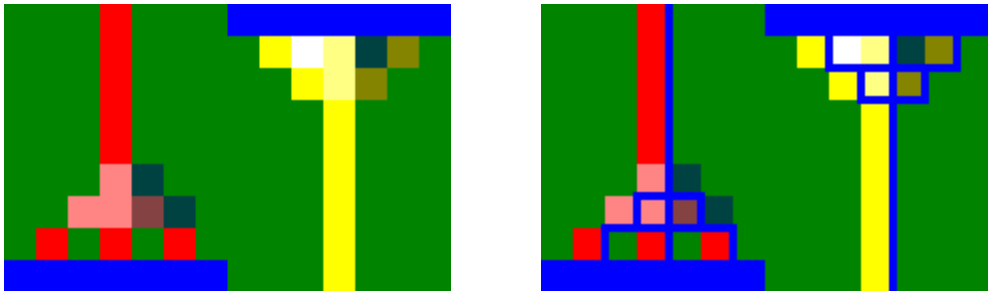
- Example 2 – A junction



This example shows how to use the dark pink pixel: “from North-East and South-West to South-East” dark pink pixel, and “from South-East to South-West and North-East” light yellow pixel aren’t very common. This junction is a case where it’s needed.

- Example 3 - How to make a delta

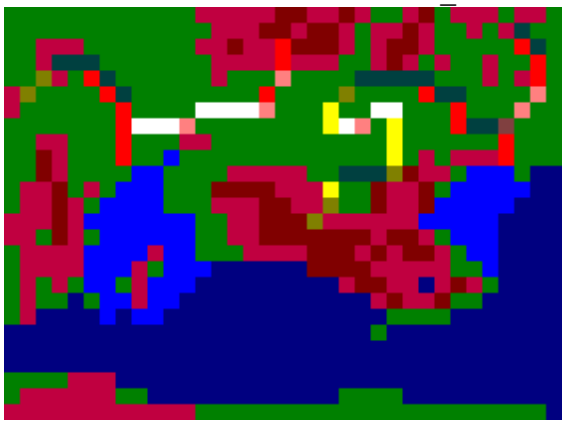
The left picture shows how pixels should be placed; the picture at the right-hand side shows what the result will be, accordingly to the key. Water fill flow towards the sea.



Always remember to pay attention on the flowing direction. You’ll obtain weird effects if a river doesn’t flow from a mountain or hill to a sea or lake, or if it meets another river in the opposite sense.

Now then, as you can see we’re working with pixels, which correspond to Civ plots. What happens if we want to place rivers without resizing first?
It can be done, but probably many angles won’t work and you’ll have to retouch them with the World Builder. Varying `iDefaultRiverSensitivity` value might improve the result.
For all the other cases, leave it as default (5).

Now instead we'll place the rivers in the resized DEFAULT_R.BMP:



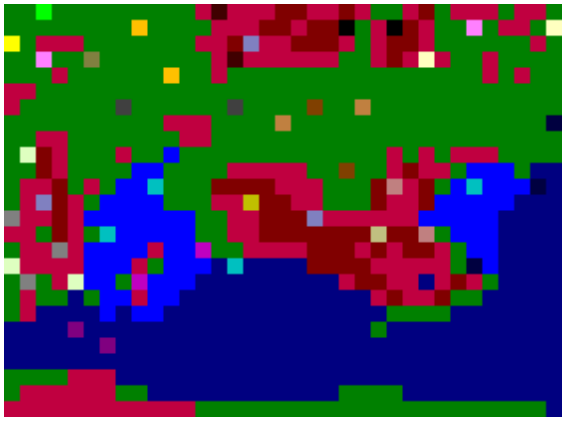
You should be able to trace these four rivers courses now.

3.4.2 Bonus Map

The last step is placing the resources.
Make a backup copy of the heights bitmap again, and rename it DEFAULT_B.BMP.

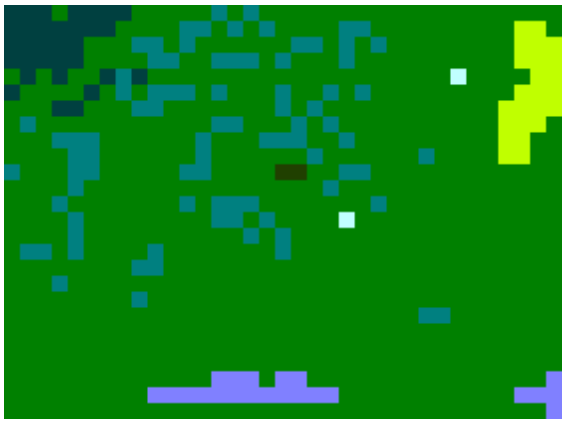
```
Aluminium  = (128, 128, 128)
Coal       = (0, 0, 0)
Copper     = (255, 128, 255)
Horse      = (192, 128, 64)
Iron       = (128, 128, 192)
Marble     = (0, 255, 255)
Oil        = (0, 0, 64)
Stone      = (192, 192, 128)
Uranium    = (0, 255, 0)
Dye        = (255, 64, 255)
Fur        = (64, 0, 0)
Gems       = (64, 255, 192)
Gold       = (192, 192, 0)
Incense    = (192, 128, 128)
Ivory      = (128, 64, 64)
Silk       = (128, 192, 255)
Silver     = (222, 222, 222)
Spices     = (255, 64, 0)
Sugar      = (0, 192, 0)
Wine       = (192, 0, 192)
Whales     = (128, 0, 128)
Banana     = (255, 255, 0)
Clam       = (0, 64, 64)
Corn       = (255, 128, 0)
Cow        = (64, 64, 64)
Crab       = (192, 0, 0)
Deer       = (128, 64, 0)
Fish       = (0, 192, 192)
Pig        = (255, 128, 128)
Rice       = (222, 255, 192)
Sheep      = (255, 255, 192)
Wheat      = (255, 192, 0)
```

Now let's place the dots, keeping an eye on the terrain map, so that resources match their appropriate terrain type:



3.4.3 Features map again

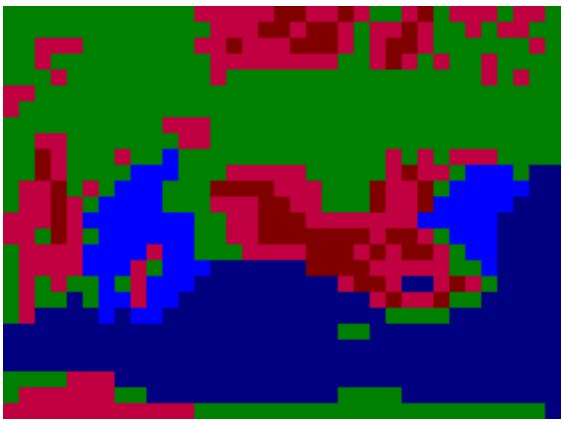
Now it's time to place the remaining features: fallout, oasis (remember to avoid any plot already occupied by a resource) and flood plains.
If `iDefaultFloodPlainsAssignment` is set to 2 or 3, you don't need to bother placing flood plains: they will be automatically added in every desert plot adjacent to a river.
You'll need to bother in case you want some rivers in desert NOT to form flood plains, and we'll suppose that this is the case.

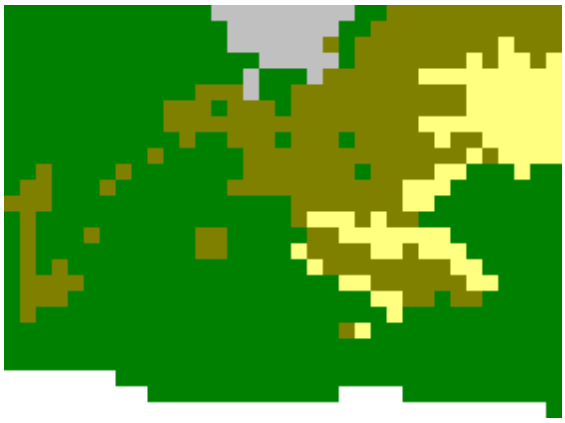


As you can see, flood plains (light green) have been placed around the right-hand side river only.

3.4.4 Last corrections

The resizing process sometimes deletes important information.
For example, in our case an island has become attached to the land. Then, we need to edit the heights map first, and then the other maps, if needed.





Now the island has been divided from the main land, the lake has been broadened, and a couple of coastlines have been refined.

The terrain map had to be edited too, because otherwise the moved island would have been just grassland (remember, sea is replaced by green in the terrain map). The other bitmaps weren't affected by the latest changes, therefore there's no need to edit them again.

3.4.5 Final result

We can now convert again, with the appropriate coasts and flood plains options.

The map is now complete.

4 Setting up the scenario

Despite being complete, the map is still empty.
Let's see how you can fill it.

4.1 Civ4WorldBuilderSave format

First of all, it's important to know how a WBS is made, because if there's anything wrong in its syntax, the scenario won't work.

A WBS is made of:

- The first line: **`Version=11`**
- Indented game info, delimited by the two lines **`BeginGame`** `and` **`EndGame`**
- If there is any team info, then for each player, it should be indented and delimited by the two lines **`BeginTeam`** and **`EndTeam.`**
- For each player, indented player info, surrounded by the two lines **`BeginPlayer`** and **`EndPlayer.`** It should include at least Leader type, Civ type and Team number.
- Indented map info, delimited by the two lines **`BeginMap`** `and` **`EndMap.`** It should include at least the world size.
- **`### Plot Info ###`** dividing line.
- Indented plot info, surrounded by the two lines **`BeginPlot`** `and` **`EndPlot.`** It should include for each plot, at least the coordinates, the terrain type and the plot type (that is, the height). If there's any feature, then the feature type in needed together with **`FeatureVariety`** flag.

If you want a wider list of all the possible entries of the WBS headers, try to save a map with the World Builder.
However, you'll see many possible settings with this example.

4.2 The scenario file

First, you need to know some basic notions about Python syntax.
These advices are taken for granted if you're a programmer, but take a look at them if you're not:

- When you open a .py file with IDLE, you'll see some red text. That is a comment. It is ignored, so you can ignore it as well.
- Text in green is delimited by quotes. That's what you have to edit most of the time.
- `\n` means new line; `\t` means a tabulation. They're required to set the indentation of the WBS.
- "\" (backslash) means that the green text continues in a new line. Don't leave empty lines below it.
- Indentation in Python is important. If you don't indentate your code, it won't work.
- Numbers are counted beginning from 0. So, if the map height is N, its plots are numbered from 0 to N-1.
- A good rule is don't touch anything you're not sure of what it is about. Just copy and paste text following the examples.
- Whenever you find words in capital letters, divided by "_", such as `GAMESPEED_NORMAL` or `LEADER_NAPOLEON` it means that it's an XML content. You'll find the other content you can change them to (such as `LEADER_ISABELLA`) in their corresponding XML file, between start and end tags.
- Check always that you don't misspell anything. Even a wrong comma can screw everything up.

4.2.1 General Settings

First of all, set the variables at the top of the scenario file this way:

```
iDefaultWidth = 35
iDefaultHeight = 26
iDefaultRiverSensitivity = 5
iDefaultCoastsAssignment = 3
iDefaultFloodPlainsAssignment = 1
iDefaultXWrapping = 0
iDefaultYWrapping = 0
iDefaultNumberOfCivs = 3
HeightMapPath = "Default_H.bmp"
TerrainMapPath = "Default_T.bmp"
FeatureMapPath = "Default_F.bmp"
RiverMapPath = "Default_R.bmp"
BonusMapPath = "Default_B.bmp"
```

EmptyScenario.py contains inside `getExtraWBSText_GameSettings()` the following general settings:

```
WBSText = "Version=11\
\nBeginGame\
\n\tSpeed=GAMESPEED_NORMAL\
\n\tCalendar=CALENDAR_DEFAULT\
\n\tGameTurn=0\
\n\tStartYear=-4000\
```

```
\nEndGame"
return WBSText
```

- Game speed can be forced to the setting specified with `Speed=` with this line:

```
\n\tForceControl=FORCECONTROL_SPEED\
```

  Any other setting contained in Sid Meiers Civilization 4\Assets\XML\GameInfo\CIV4ForceControlInfos.xml can be set this way.

- Now, if we want to change the starting year to 1500 with one month increments, with 100 turns as the length of the scenario, we should add the following line, replacing the existing ones if needed:

```
\n\tStartYear=1500\
\n\tCalendar=CALENDAR_MONTHS\
\n\tMaxTurns=100\
```

- Some game options contained in Sid Meiers Civilization 4\Assets\XML\GameInfo\CIV4GameOptionInfos.xml and CIV4MPOptionInfos.xml (see those files for the full list of the content you can use) can be set like this:

```
\n\tOption=GAMEOPTION_NO_CITY_RAZING\
\n\tMPOption=MPOPTION_SIMULTANEOUS_TURNS\
```

- To add victory types, add the lines you need from the following:

```
\n\tVictory=VICTORY_SCORE\
\n\tVictory=VICTORY_TIME\
\n\tVictory=VICTORY_CONQUEST\
\n\tVictory=VICTORY_DOMINATION\
\n\tVictory=VICTORY_CULTURAL\
\n\tVictory=VICTORY_SPACE_RACE\
\n\tVictory=VICTORY_DIPLOMATIC\
\n\tTargetScore=0\
\n\tMaxCityElimination=0\
```

- You may want to specify a mod path to load automatically, in case the scenario features a mod. In this case, add this line:

```
\n\tModPath=Mods\....your mod's name...\
```

- Furthermore, you can add a short description of the scenario that will be shown when loading the scenario, adding this line:

```
\n\tDescription=This is just an example\
```

4.2.2 World Info

This is the basic world info that should be included:

```
	WBSText = "\n\tworld size=WORLDSIZE_STANDARD \
	\n\tclimate=CLIMATE_TEMPERATE \
	\n\tsealevel=SEALEVEL_MEDIUM \
	\n\tnum plots written=%d\nEndMap\n" %(bmpConverter.iSizeX *
bmpConverter.iSizeY)
	return WBSText
```

- World size will affect some parameters such as maintenance.
  So, let's change it to tiny:

  ```
  \n\tworld size=WORLDSIZE_TINY \
  ```

- You can change wrapping settings from here. The converter automatically puts here the default values you set at the top of the file:

  ```
  \n\twrap X=1 \
  \n\twrap Y=0 \
  ```

- You decide if polar ice caps will be shown or not, adding these two lines.

  ```
  \n\ttop latitude=90 \
  \n\tbottom latitude=-90 \
  ```

  90 and -90 will make caps shown. Lower values (closer to the equator) won't.
  These values also influence how the world looks when zooming out: if you want a flat view, set them to lower values. Otherwise, when you discover Astronomy, the world will look spherical.
  Even forest varieties will be affected by latitude. Closer to the poles (90 and -90) snowy trees will be more frequent; pine trees will be more common in temperate latitudes, and there will be only leafy trees near the equator and in warm regions.

4.2.3 Civilization Info

For each player, the WBS should contain at least:

```
BeginPlayer
	LeaderType=NONE
	CivType=NONE
	Team=#
EndPlayer
```

With # as the player number, from 0 to 17 (18 is the barbarian civ).

So, EmptyScenario.py contains inside `getExtraWBSText_Civs`:

```
"\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=0\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=1\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=2\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=3\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=4\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=5\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=6\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=7\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=8\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=9\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=10\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=11\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=12\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=13\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=14\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=15\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=16\nEndPlayer \
\nBeginPlayer\n\tLeaderType=NONE\n\tCivType=NONE\n\tTeam=17\nEndPlayer"
```

That is, no player info. In this case, the number of players will determines automatically from CIV4WorldInfo.xml, accordingly to your world size setting.
We'll add 3 civs: 1 playable and 2 not playable.

```
\nBeginPlayer \
\n\tTeam=0 \
\n\tLeaderType=LEADER_MAO_ZEDONG \
\n\tCivType=CIVILIZATION_CHINA \
\n\tPlayableCiv=1 \
\n\tHandicap=HANDICAP_NOBLE \
\nEndPlayer \
\nBeginPlayer \
\n\tTeam=1 \
\n\tLeaderType=LEADER_NAPOLEON \
\n\tCivType=CIVILIZATION_FRANCE \
\n\tPlayableCiv=0 \
\n\tHandicap=HANDICAP_NOBLE \
\nEndPlayer \
\nBeginPlayer \
\n\tTeam=2 \
\n\tLeaderType=LEADER_BISMARCK \
\n\tCivType=CIVILIZATION_GERMANY \
\n\tPlayableCiv=0 \
\n\tHandicap=HANDICAP_NOBLE \
\nEndPlayer \
```

Leader type and Civ type are simply the links to leader personality stored in CIV4LeaderHeadInfos.xml and to civilizations settings stored in CIV4CivilizationInfos.xml.

- We can change labels and aspect of the first civ, adding:

```
\n\tCivDesc=Italian Empire \
\n\tCivShortDesc=Italy \
\n\tCivAdjective=Italian \
\n\tLeaderName=Victor Emmanuel II \
\n\tFlagDecal=Art/Interface/TeamColor/FlagDECAL_Laurels.dds \
\n\tColor=PLAYERCOLOR_BLUE \
\n\tArtStyle=ARTSTYLE_EUROPEAN \
```

  These settings will override the ones contained in CIV4CivilizationInfos.xml.
  Unfortunately, civ name, adjectives and leader name aren't localized. To avoid this problem, you'll have to edit CIV4GameTextInfos_Objects.xml, the file that contains these names.

  You should then add

```
\n\tCivDesc=TXT_KEY_CIV_ITALY_DESC\
\n\tCivShortDesc=TXT_KEY_CIV_ITALY_SHORT_DESC\
\n\tCivAdjective=TXT_KEY_CIV_ITALY_ADJECTIVE\
\n\tLeaderName=TXT_KEY_LEADER_VICTOR\
```

  instead of the lines written previously, and add the corresponding entries in the XML file, located in Sid Meiers Civilization 4\Assets\XML\Text.
  Creating a subfolders structure inside Sid Meiers Civilization 4\Mods and editing only mod-related files is highly recommended (see next Chapter, "Expading the Scenario").

- You can specify that the flag should ignore the player colour, adding

```
\n\tWhiteFlag=1 \
```

  That's useful when you have flags with more than two colours, like this one:

- Starting era can be changed as well. This setting will affect era-related music and artstyle.

```
\n\tStartingEra=ERA_RENAISSANCE \
```

- The new civ requires its own city list:

```
\n\tCityList=Rome \
\n\tCityList=Milan \
\n\tCityList=Turin \
\n\tCityList=Florence \
\n\tCityList=Venice \
\n\tCityList=Pisa \
\n\tCityList=Livorno \
\n\tCityList=Naples \
\n\tCityList=Bologna \
\n\tCityList=Siena \
\n\tCityList=Palermo \
```

  If no city list is specified, then the default one in the .xml file will be used.

- Some civs can be turned to Minor civs with this line:

```
\n\tMinorNationStatus=1 \
```

  Minor civs are always at war with all the others. Their leader cannot be contacted.

- You can set starting gold, state religion and starting civics for each player this way:

```
\n\tStartingGold=30 \
\n\tStateReligion=RELIGION_CHRISTIANITY \
\n\tCivicOption=CIVICOPTION_GOVERNMENT, Civic=CIVIC_DESPOTISM \
\n\tCivicOption=CIVICOPTION_LEGAL, Civic=CIVIC_BARBARISM \
\n\tCivicOption=CIVICOPTION_LABOR, Civic=CIVIC_SLAVERY \
\n\tCivicOption=CIVICOPTION_ECONOMY, Civic=CIVIC_DECENTRALIZATION \
\n\tCivicOption=CIVICOPTION_RELIGION, Civic=CIVIC_PAGANISM \
```

- As leaders can be renamed, it's possible to change their mindset in an elegant way. For example, you can replace

```
\nBeginPlayer \
\n\tTeam=2 \
\n\tLeaderType=LEADER_BISMARCK \
\n\tCivType=CIVILIZATION_GERMANY \
\n\tPlayableCiv=0 \
\n\tHandicap=HANDICAP_NOBLE \
\nEndPlayer \
```

  with

```
\nBeginPlayer \
\n\tTeam=2 \
\n\tLeaderType=LEADER_GENGHIS_KHAN \
\n\tCivType=CIVILIZATION_GERMANY \
\n\tLeaderName=Bismarck \
\n\tPlayableCiv=0 \
\n\tHandicap=HANDICAP_NOBLE \
\nEndPlayer \
```

Now Otto von Bismack will get Genghis Khan's traits and attitude.
Being Germany a minor civ, Genghis Khan's face won't be shown.

- There are civ-specific handicap levels.
  Player difficulty setting will determine AI bonuses/maluses, while civ-specific levels will affect game modifiers such as maintenance.
  So, if you lower handicap level to an AI-controlled civ, it will perform better. If you raise it, the opposite will happen.

  We are setting now an Emperor handicap level to France:

  ```
  \n\tHandicap=HANDICAP_EMPEROR \
  ```

- Attitude between leaders can be changed this way:

  ```
  \n\tAttitudePlayer=#, AttitudeExtra=## \
  ```

  Where # is the number of the player, and ## is the modifier.

  In our scenario, we can set

  ```
  \n\tAttitudePlayer=1, AttitudeExtra=5 \
  ```

  in team 0, and

  ```
  \n\tAttitudePlayer=0, AttitudeExtra=-10 \
  ```

  in team 1.
  Now Italian leader is pleased with French leader, while French dislikes Italian.

Another part of data is stored in the team info, inside `getExtraWBSText_Diplomacy()`

In EmptyScenario.py, team info for each civ is:

```
\nBeginTeam \
\nEndTeam \
```

- Team data includes technologies:

```
\n\tTech=TECH_MINING \
\n\tTech=TECH_BRONZE_WORKING \
```

Be sure to check that the previously selected civics match the technologies that enable them.

- Team data includes diplomacy too:

  To set a Italy-France war, we'll set in player 0

  ```
  \n\tAtWar=1 \
  ```

  and in player 1

  ```
  \n\tAtWar=0 \
  ```

  To enable contacts between two civs that can't see each other yet, such as Italy and Germany, you should set in player 0

  ```
  \n\tContactWithTeam=2 \
  ```

  and in player 1

  ```
  \n\tContactWithTeam=0 \
  ```

  But we won't add this last thing in the scenario, because being Germany a minor civ, it can't have any contact.

  Furthermore, you can set open borders with:

  ```
  \n\tOpenBordersWithTeam=# \
  ```

  As usual, adding this line for both the civilizations.

4.2.4 Cities

Now let's place some cities.

Delete this line inside `WriteStartingPlots()`

```
bmpc.DebugLogging("No starting plot entries\n")
```

and replace it with an `if` statement:

```
if iPlotX == 14 and iPlotY == 10:
```

```
bmpc.OutputFile.write("\tRouteType=ROUTE_ROAD\n\tBeginCity \
\n\t\tCityOwner=0 \
\n\t\tCityName=Rome \
\n\t\tCityPopulation=6 \
\n\t\tPlayer0Culture=100 \
\n\t\tPlayer2Culture=10 \
\n\t\tBuildingType=BUILDING_PALACE \
\n\t\tBuildingType=BUILDING_COLOSSEUM \
\n\t\tBuildingType=BUILDING_WALLS \
\n\t\tBuildingType=BUILDING_CASTLE \
\n\t\tBuildingType=BUILDING_OBELISK \
\n\t\tBuildingType=BUILDING_MARKET \
\n\t\tBuildingType=BUILDING_COURTHOUSE \
\n\t\tBuildingType=BUILDING_CHRISTIAN_CATHEDRAL \
\n\t\tBuildingType=BUILDING_CHRISTIAN_MONASTERY \
\n\t\tReligionType=RELIGION_CHRISTIANITY \
\n\tEndCity\n")
```

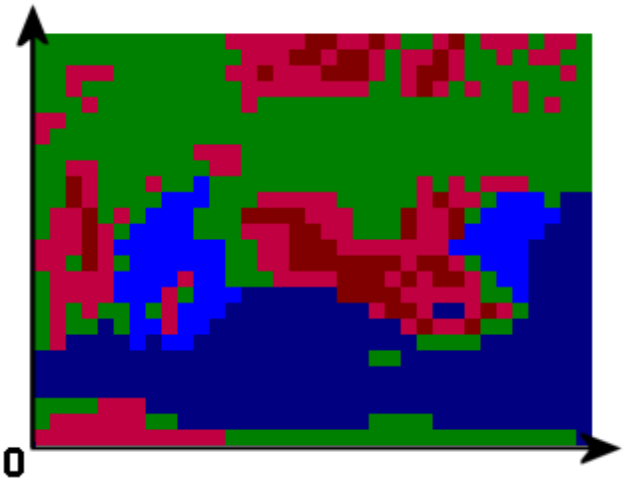
That is, in plot (14, 10) will be placed Rome, belonging to Italy, with some buildings and a small percentage of German culture. If the city had only the owner civ's culture, instead of writing

```
\n\t\tPlayer0Culture=100 \
```

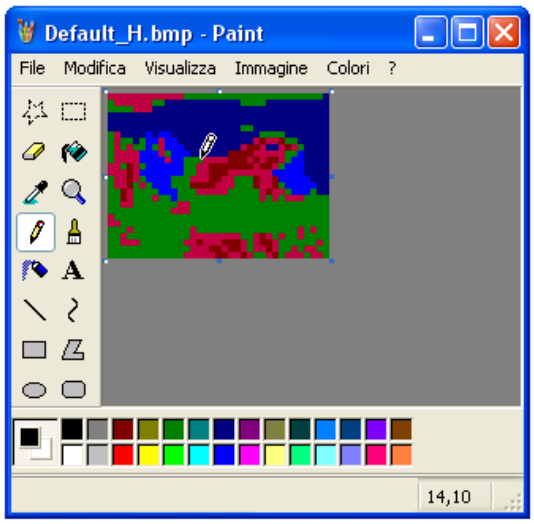
you could specify it with:

```
\n\t\tCityCulture=100 \
```

- Now, how do you know the coordinates?
  You should count pixels from the bottom left.



But then it's easier if you flip vertically the map, and look the cursor position. Even MS Paint shows it.



Now let's place a second city, this time a holy city:

```
if iPlotX == 19 and iPlotY == 10:
      bmpc.OutputFile.write("\tRouteType=ROUTE_ROAD\n\tBeginCity \
      \n\t\tCityOwner=1 \
      \n\t\tCityName=Paris \
      \n\t\tCityPopulation=5 \
      \n\t\tPlayer1Culture=100 \
      \n\t\tBuildingType=BUILDING_PALACE \
      \n\t\tBuildingType=BUILDING_LIBRARY \
      \n\t\tBuildingType=BUILDING_MARKET \
      \n\t\tBuildingType=BUILDING_ISLAMIC_TEMPLE \
      \n\t\tBuildingType=BUILDING_ISLAMIC_CATHEDRAL \
      \n\t\tBuildingType=BUILDING_ISLAMIC_SHRINE \
      \n\t\tReligionType=RELIGION_ISLAM \
      \n\t\tReligionType=RELIGION_JUDAISM \
      \n\t\tHolyCityReligionType=RELIGION_ISLAM \
      \n\tEndCity\n")
```

By the way, remember to remove Rome and Paris from Italian and French city lists (otherwise a settler may found a city of the same name). It will be easy for Rome; to remove Paris instead you can either add the whole French list to the WBS and then remove a line, or edit CIV4CivilizationInfos.xml.

4.2.5 Terrain changes

You may have noticed that (19, 10) corresponds to a peak.
If you don't want to move the city elsewhere (let's suppose that's its historical location), there's the opportunity to turn it a hill from the scenario python file.

You could simply edit the bitmaps, but in case you want to keep them unchanged for other uses (and you want to keep just one copy of it, so that you don't have to update twice), it's better this way.

```
if (iPlotX == 19 and iPlotY == 9) or \
      (iPlotX == 19 and iPlotY == 10) or \
      (iPlotX == 20 and iPlotY == 9):
            bmpc.aaiPlotTypes[iPlotX][iPlotY] = 1
```

Adding these lines into `TerrainChanges()` will make the plot occupied by the city, and the surrounding coastal zones passable (1 means 'Hill')

```
if (iPlotX == 18 and iPlotY == 10):
      bmpc.aasTerrainTypes[iPlotX][iPlotY] = 'TERRAIN_PLAINS'
```

Instead, these lines will make a plot adjacent to Paris a plain instead of a desert.

4.2.6 Units

You can add a unit in Rome adding the following lines inside the `if` statement, indented the same as the other `bmpc.OutputFile.write`

```
if iPlotX == 14 and iPlotY == 10:
      bmpc.OutputFile.write("\tBeginUnit \
      \n\t\tUnitType=UNIT_ARCHER, UnitOwner=0 \
      \n\t\tLevel=1, Experience=0 \
      \n\tEndUnit\n")
      bmpc.OutputFile.write(". . . ")
```

We are adding now a German settler and worker:

```
if iPlotX == 9 and iPlotY == 18:
      bmpc.OutputFile.write("\tBeginUnit \
      \n\t\tUnitType=UNIT_SETTLER, UnitOwner=2 \
      \n\t\tLevel=1, Experience=0 \
      \n\tEndUnit\n")
      bmpc.OutputFile.write("\tBeginUnit \
      \n\t\tUnitType=UNIT_WORKER, UnitOwner=2 \
      \n\t\tLevel=1, Experience=0 \
      \n\tEndUnit\n")
```

Barbarian index is 18:

```
if iPlotX == 20 and iPlotY == 22:
      bmpc.OutputFile.write("\tBeginUnit \
```

```
\n\t\tUnitType=UNIT_SPEARMAN, UnitOwner=18 \
\n\t\tLevel=1, Experience=0 \
\n\t\tUnitAIType=UNITAI_ATTACK \
\n\tEndUnit\n")
```

We've added here a further line (UNITAI) to change the role a spearman has for the AI to an attacker.

4.2.7 Starting locations

If you just want an empty map with starting locations, and you don't want to put starting units manually, then you have to edit `WriteStartingPlots()`.

```
if iPlotX == # and iPlotY == #:
            bmpc.OutputFile.write("\tStartingPlot\n")
```

These two lines will add a starting plot (replace # with actual coordinates).
If there are no players set, then the random civs in the game will start from those plots (remember, with no player data, the number of players will determined automatically from CIV4WorldInfo.xml, accordingly to the world size).

Instead, if you want to assign a starting plot to a certain civ, you'll have to edit player info again, adding:

```
\n\tStartingX=#, StartingY=# \
```

And of course, the coordinates must match the ones of one of the starting plots.

4.2.8 Improvements and Fog of War

Terrain improvements, routes and fog of war are placed in the same way. Just put the list of the coordinates inside the corresponding area, with the following format:

```
dltGoodyHut =     [   (1, 7),
                      (3, 20),
                      (9, 9),
                      (13, 20),
                      (23, 11),
                      (30, 8)
                  ]
```

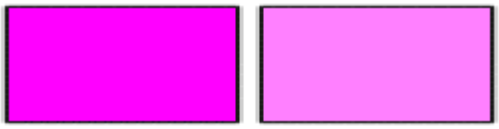
Be sure to assign fog of war lists to the correct index:

`dltRevealPlots[0]` will contain Italian revealed area; `dltRevealPlots[1]` the French one; `dltRevealPlots[2]` the German one. The rest of `dltRevealPlots[]` can be ignored.

- Instead, if you want to reveal the whole map for certain civs, add in the team info (together with technologies and diplomacy) this line:

  ```
  \n\tRevealMap=1 \
  ```

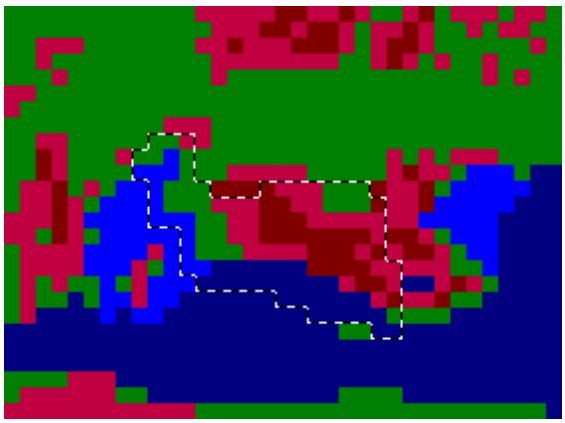
Of course, the list of revealed pixels can be huge and can't be written manually.
But you can extract the list from a bitmap, using the Converter.
Selecting "Extract Fog of War" will let you choose a bitmap that should contain at least the following colours:



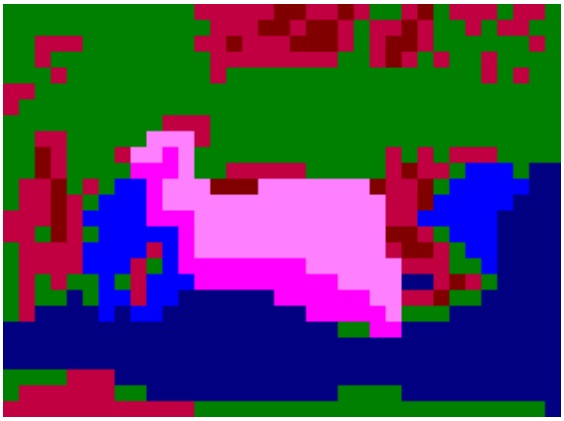
```
Fog of war (darker)   = (128, 128, 128)
Fog of war (lighter)  = (0, 0, 0)
```

They are equivalent: both represent a plot that is revealed to a certain civilization.
The best way to use them is following these steps:

First, make a backup copy of Default_H.BMP. Let's call it Default_FOW.BMP.
You can select a region with the lazus (an instrument that any photoediting program has)



Then, replace the whole region, which represents the explored area, with either one of the two shades of magenta, or with both (here you can see an effective use: darker magenta is explored see; lighter is explored land).



When you select the option in the converter main menu, you just have to choose the bitmap, in this case Default_FOW.BMP.
The converter will create a file named FOW_Output.txt, which will contain the list of the coordinates, in the proper format:

```
(8, 15), (8, 16),
(9, 12), (9, 13), (9, 14), (9, 15), (9, 16), (9, 17),
(10, 12), (10, 13), (10, 14), (10, 15), (10, 16), (10, 17),
(11, 9), (11, 10), (11, 11), (11, 12), (11, 13), (11, 14), (11, 15), (11, 16), (11, 17),
(12, 8), (12, 9), (12, 10), (12, 11), (12, 12), (12, 13), (12, 14),
(13, 8), (13, 9), (13, 10), (13, 11), (13, 12), (13, 13),
(14, 8), (14, 9), (14, 10), (14, 11), (14, 12), (14, 13),
(15, 8), (15, 9), (15, 10), (15, 11), (15, 12), (15, 13),
(16, 8), (16, 9), (16, 10), (16, 11), (16, 12), (16, 13), (16, 14),
(17, 7), (17, 8), (17, 9), (17, 10), (17, 11), (17, 12), (17, 13), (17, 14),
(18, 7), (18, 8), (18, 9), (18, 10), (18, 11), (18, 12), (18, 13), (18, 14),
(19, 6), (19, 7), (19, 8), (19, 9), (19, 10), (19, 11), (19, 12), (19, 13), (19, 14),
(20, 6), (20, 7), (20, 8), (20, 9), (20, 10), (20, 11), (20, 12), (20, 13), (20, 14),
(21, 6), (21, 7), (21, 8), (21, 9), (21, 10), (21, 11), (21, 12), (21, 13), (21, 14),
(22, 6), (22, 7), (22, 8), (22, 9), (22, 10), (22, 11), (22, 12), (22, 13), (22, 14),
(23, 5), (23, 6), (23, 7), (23, 8), (23, 9), (23, 10), (23, 11), (23, 12), (23, 13),
(24, 5), (24, 6), (24, 7), (24, 8), (24, 9)
```

All you have to do now is copy this list, and paste it into the scenario python file, in the element of `dltRevealPlots[]` corresponding to the civilization.

Repeat this process for each civilization you need to set up fog of war.
As you may have thought, you can re-use this option for placing roads or any other terrain improvement.

4.3 Reference

The way a WBS works is stored inside Sid Meiers Civilization 4\Assets\Python\pyWB\CvWBDesc.py
If you have any doubt, that's the real source to consult.

4.4 Final result

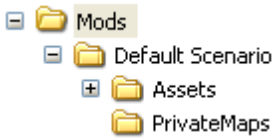
DefaultScenario.py now contains all the mentioned additions. Following the instructions yourself, you should obtain the same final, playable scenario.

5 Expanding the scenario

We have created just a WBS so far.
Now, this appendix describes how to expand it.

5.1 Directory structure

Create a Mods subfolder and give it a name. Then create two own subfolders: Assets and PrivateMaps



Sid Meiers Civilization 4\Mods\Default Scenario\Assets folder can contain other subdirectories, such as Art, Python or XML. Just open the main Civilization IV folders, and re-create a part of the structure. Each file placed in these subfolder will superimpose the original one.

Here's a list of where you can find the most common info:

- **Buildings**: Assets\XML\Buildings*.*
- **Civilizations**: Assets\XML\Civilizations\CIV4CivilizationInfos.xml
- **Leaders**: Assets\XML\Civilizations\CIV4LeaderHeadInfos.xml and Assets\XML\Civilizations\ArtDefines\ CIV4ArtDefines_Leaderhead.xml
- **Traits**: Assets\XML\Civilizations\CIV4TraitInfos.xml
- **Civics**: Assets\XML\GameInfo\CIV4CivicInfos.xml
- **Timeline**: Assets\XML\GameInfo\CIV4GameSpeedInfo.xml
- **Difficulty levels**: Assets\XML\GameInfo\CIV4HandicapInfo.xml
- **World size**: Assets\XML\GameInfo\ CIV4WorldInfo.xml
- **Colours**: Assets\XML\Interface\CIV4PlayerColorInfos.xml and CIV4ColorVals.xml
- **Technologies**: Assets\XML\Technologies\CIV4TechInfos.xml
- **Terrains**: Assets\XML\Terrain*.*
- **Civilopedia**: Assets\XML\Text\CIV4GameText_Civilopedia_*.xml
- **Diplomacy text**: Assets\XML\Text\CIV4DiplomacyText.xml

- **Localized civ and leader names**: Assets\XML\Text\CIV4GameTextInfos_Objects.xml
- **Rest of the text (including localization)**: Assets\XML\Text\
- **Units**: Assets\XML\Units\CIV4UnitInfos.xml
- **Unit scale**: Assets\XML\Art\CIV4ArtDefines_Unit.xml
- **Miscellaneous**: Assets\XML\GlobalDefines.xml
- **Event manager**: Assets\Python\CvEventManager.py
- **Ingame interface**: Assets\Python\Screens
- **Map scripts**: Assets\PublicMaps*.py
- **Icons**: Assets\Art\Interface\Buttons*.*
- **Small icons**: Assets\res\Fonts\GameFont.tga and GameFont75.tga
- **Flag decals**: Assets\Art\Interface\TeamColor*.* and Assets\Art\Interface\Buttons\Civilizations*.*
- **Terrain graphics**: Assets\Art\Terrain*.*
- **Leaderheads**: Assets\Art\LeaderHeads*.* and Assets\XML\Art\CIV4ArtDefines_Leaderhead.xml
- **Unit graphics**: Assets\Art\Units*.*

5.2 Ini file

Now, create the file Default Scenario.ini and place it in \Mods\Default Scenario. It should contain this info, which needs no explainations:

```
[CONFIG]

; This mod is only for single player games
SinglePlayerOnly = 1

; Set to 1 to disallow use of WorldBuilder scenarios
IgnoreWorldBuilderScenarios = 0

; Allow public maps to be used with this mod
AllowPublicMaps = 0

; Mod Image file
ImageFile = 0

; Name of Mod
Name = Default Scenario

; Description of Mod
Description = Generic Mod
```

5.3 Loading a mod

There are 3 ways to load a mod:

- Put the WBS inside Sid Meiers Civilization 4\PublicMaps and make sure that it contains the line `ModPath=Mods\Default Scenario` inside. Now, you can load a mod selecting "Load a Scenario"
- Put the WBS inside Sid Meiers Civilization 4\Mods\Default Scenario\Private Maps. That folder contains WBS maps that can be loaded only if running this mod. Now you can load the mod from inside the game (with the 'LOAD A MOD' option).
- If you edit CivilizationIV.ini (usually in My Documents\My Games\ Sid Meiers Civilization 4), specifying a mod folder (such as Mods\Default Scenario, '0' for none) where you find '`Mod =`', the game will automatically load the mod at startup.