This documentation holds the information about the process, how the European Forest Scenery scenery was created.

Creating the European Forest Scenery

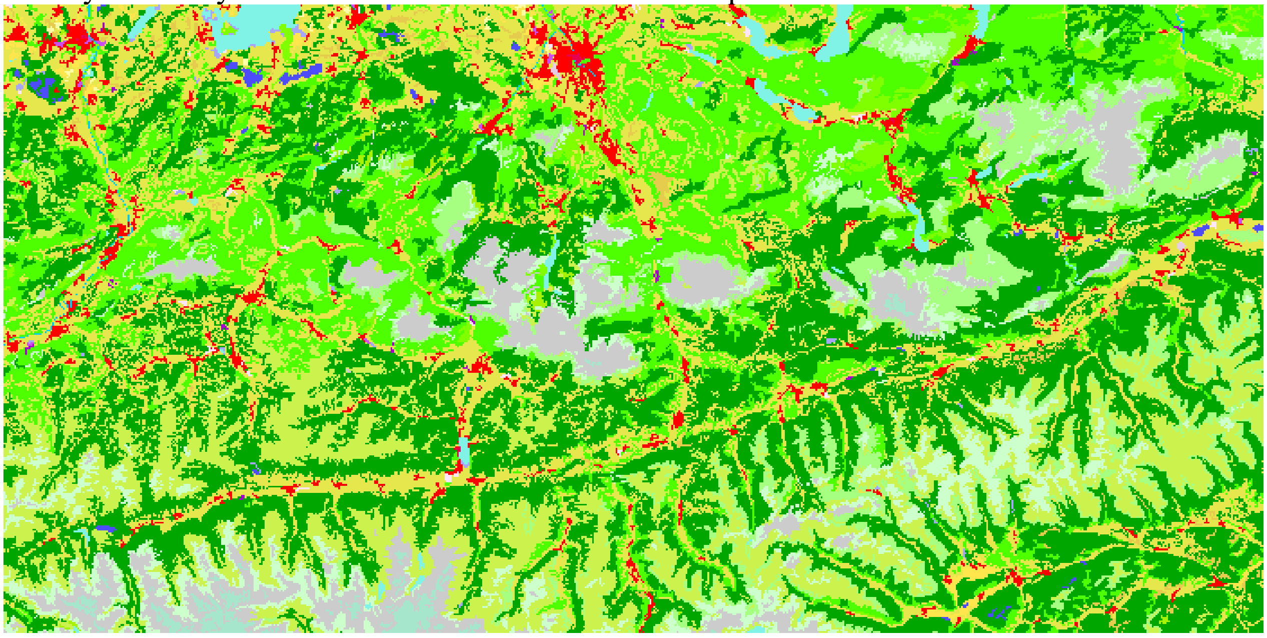
http://www.alpilotx.de

by Andras Fabian, 03.05.2007

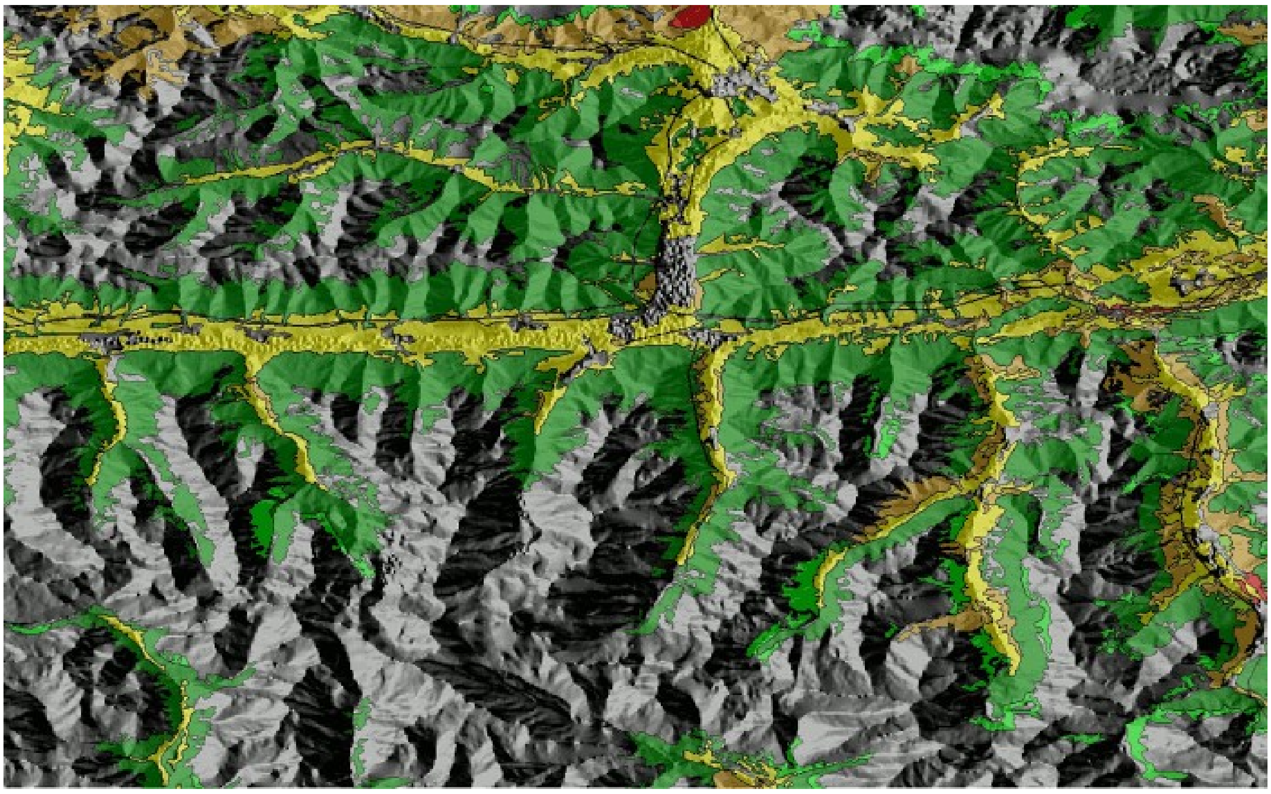
Content

Data source

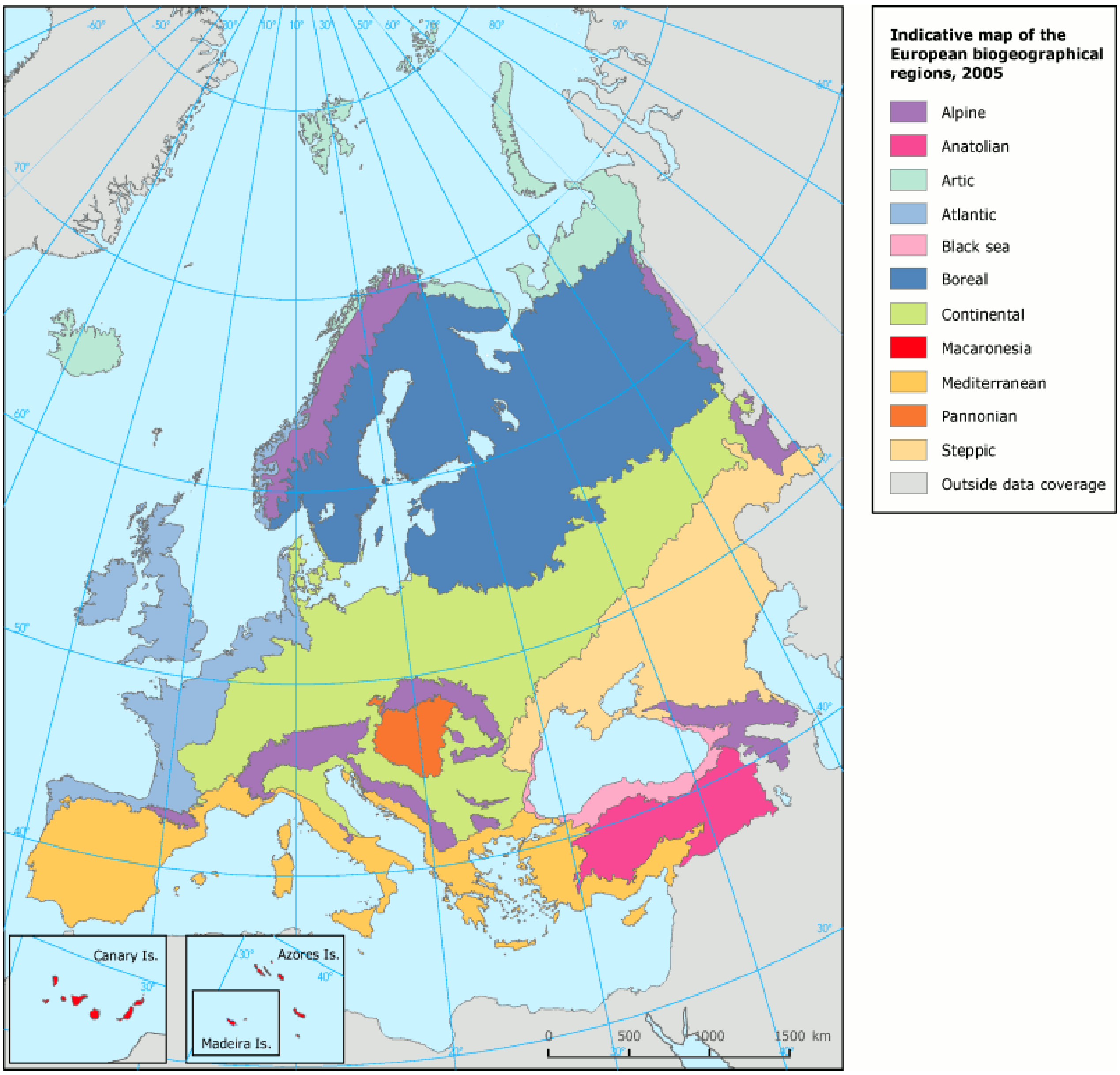
The scenery data is built from mainly 3 data sources, which all take their part in some way. This data are:

- Corine land cover 2000 vector by country (CLC2000) from the European Environment Agency (EEA). This is the most important part because it holds all the land cover information with a fairly accurate resolution of around 100m. It knows about 44 types of land, and 8 of them are used in this overlay scenery. **CORINE** looks - for example - like this in a GIS tool:

Those are the interesting layers:
  - Fruit trees and berry plantations (CLC_ID: 222)
  - Olive groves (CLC_ID: 223) - in Mediterranean area
  - Broad-leaved forest (CLC_ID: 311)
  - Coniferous forest (CLC_ID: 312)
  - Mixed forest (CLC_ID: 313)
  - Moors and heathland (CLC_ID: 322) - it tuned out, that this do represent most of the high altitude low coniferous vegetation.
  - Sclerophyllous vegetation (CLC_ID: 323) - in Mediterranean area
  - Transitional woodland-shrub (CLC_ID: 324)
  - **NEW:** Agricultural areas - Pastures (CLC_ID: 231) - for sparsely distributed trees (not "real" forests)
  - **NEW:** Agricultural areas - Land principally occupied by agriculture, with significant areas of natural vegetation (CLC_ID: 243) - for sparsely distributed trees (not "real" forests)

  So at the end, what I used as data would look like this (as seen in GRASS GIS with some SRTM shaded relief as "background" - around "Zell am See" in Austria):



- Biogeographical regions, Europe 2005 is a dataset, which consists of polygons, describing the different biogeographic regions of Europe. They are very useful, in wastly improving this scenery, by helping to assign the "right type" of forest/trees in each region. This means, for example, a conifers in Sweden don't look like conifers in Greece. To get an idea, here is the image from the EEA, which should give you an idea, what I'm talking about:



Because - at least at the coastlines - this data not allways covered exactly thcoastline of the **CORINE** data, I had to extend this areas a bit by hand. For this task I used the vector editing features of Quantum GIS 0.8.

- VMAP0 Vector Data is also known as the Digital Chart of the World (DCW), which X-Plane uses in the g for all the vector features like roads, railways. And because **CORINE** does not know about all those roads and railroads, I had to cut them out "by hand" from it. Also the "built up" (city) areas were used for minor trimming of the forests. You can find a very good introduction to **VMAP0** at mapAbility.com.
- DAFIF (since October 2006 not freely available to the public) is the United States Flight Information File which holds airport polygons, which are also the basis for most X-Plane airports. I had to cut out them - with some fair amount of buffering - so that there are no trees at airports.

Software used

Throughout the transformation process, I only used open source software (especially as I did it under Linux). The main players were:

- GRASS GIS 6.2 for all the tasks with the geodata.
- Quantum GIS 0.8 for some editing of the biogeographic regions data (and for some visualisation tasks).
- BASH scripts were used for all the repetitive tasks. It is a big pluss that, you can script every **GRASS** process from **BASH**.
- DSFTool.exe from the X-Plane Scenery SDK for the text to DSF transformation. For this I also had to use WINE, which lets you run many Windows programs under Linux.
- And basic Linux tools like **grep** and **perl**.

Preparing the data

The first - and maybe easiest - step was, to import all the polygons from the **CORINE county datasets** into so that I can work with them in **GRASS**. **CORINE county datasets** are coming - well as the name already says - in one ESRI Shape file (.SHP) per country. And **GRASS** supports loading it through the OGR Simple Feature Library. I also used a filter here, so I could separate the different forest layers for every country. I only had to use some consistent naming schema, so I did not mix up things later.

As I already indicated above, the **CORINE** data wasn't usable "out of the box", because although it is very detailed, it doesn't "know" the roads, railroads and airports as they are represented in X-Plane.

Interestingly it conforms very well with the coastlines and water bodies, because X-Plane used the Shuttle Radar Topography Mission Water Body Dataset (SWBD) for this, and after some checking it turned out, that the EEA seems to have used it too. So there was no need for cutting out **SWBD** polygons. Only in very few areas you will see some trees in the water.

Rivers are more problematic. As it turned the X-Plane developers derived from different datasets, and corrected their locations, so that they are running at the deepest point of valleys. They had to do this, because normally when you combine "plain" Shuttle Radar Topography Mission (SRTM) - on which the X-Plane mesh is based - with the **VMAP0** rivers, they don't agree very well. So all in all, they "invented" their own rivers, for which I plainly had no chance to find "raw data" to work with. I talked about this issue with Ben Supnik and he plainly told, that not even he has the data in an easily usable form. His idea was, to extract such information directly from the original **Global Scenery DSF** files. And this is one thing, which I did not do, because that way this project would have get out of hand.

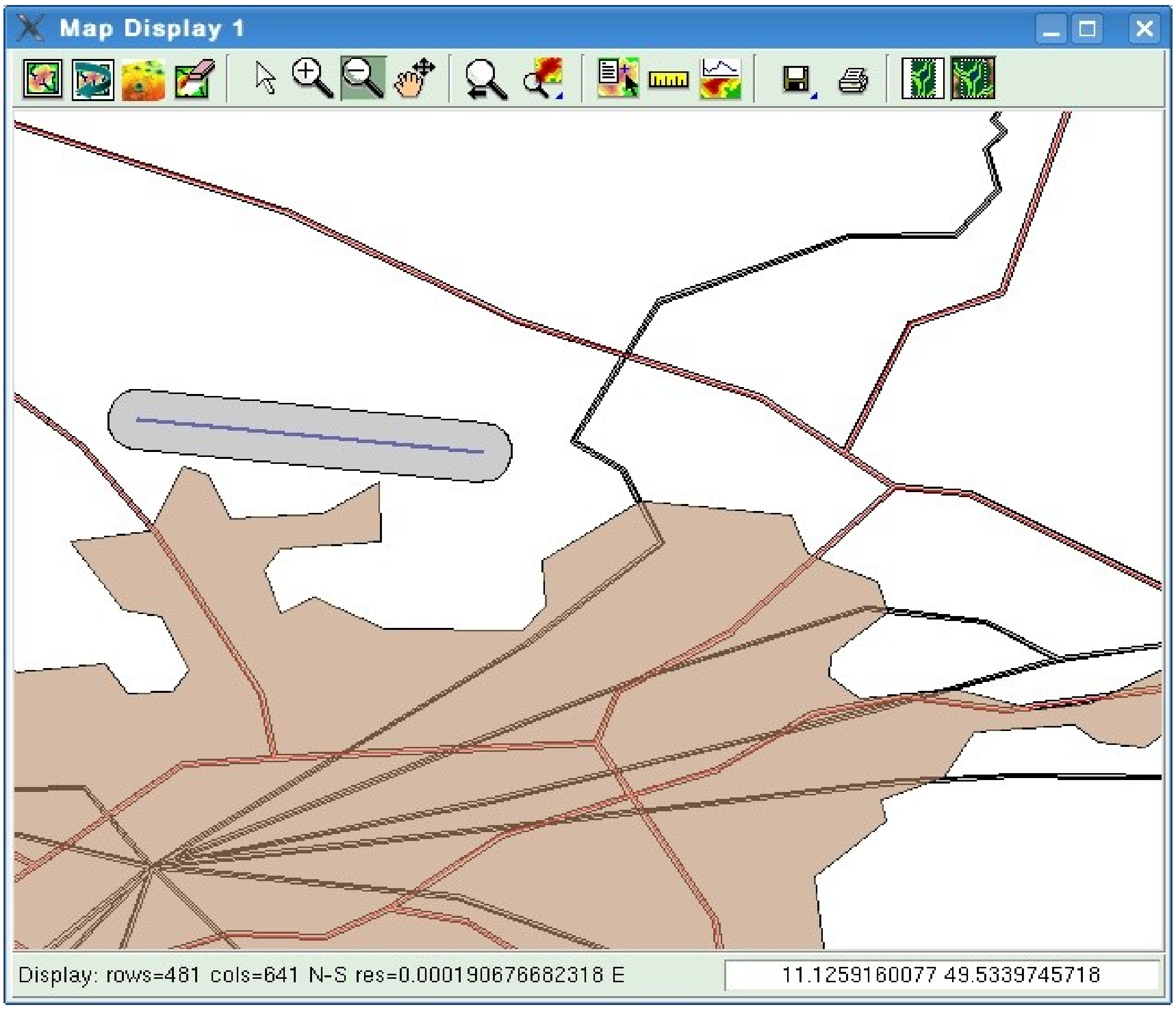
So after clearing all this informations, I decided to "only" go with:

- VMAP0 roads
- VMAP0 railroads
- VMAP0 cities (built up area)
- DAFIF runway polygons

So, now I had to find out, how I can "cut out" all this features from the original **CORINE** polygons. The cities were the easiest, because they are already polygons with the right extent.

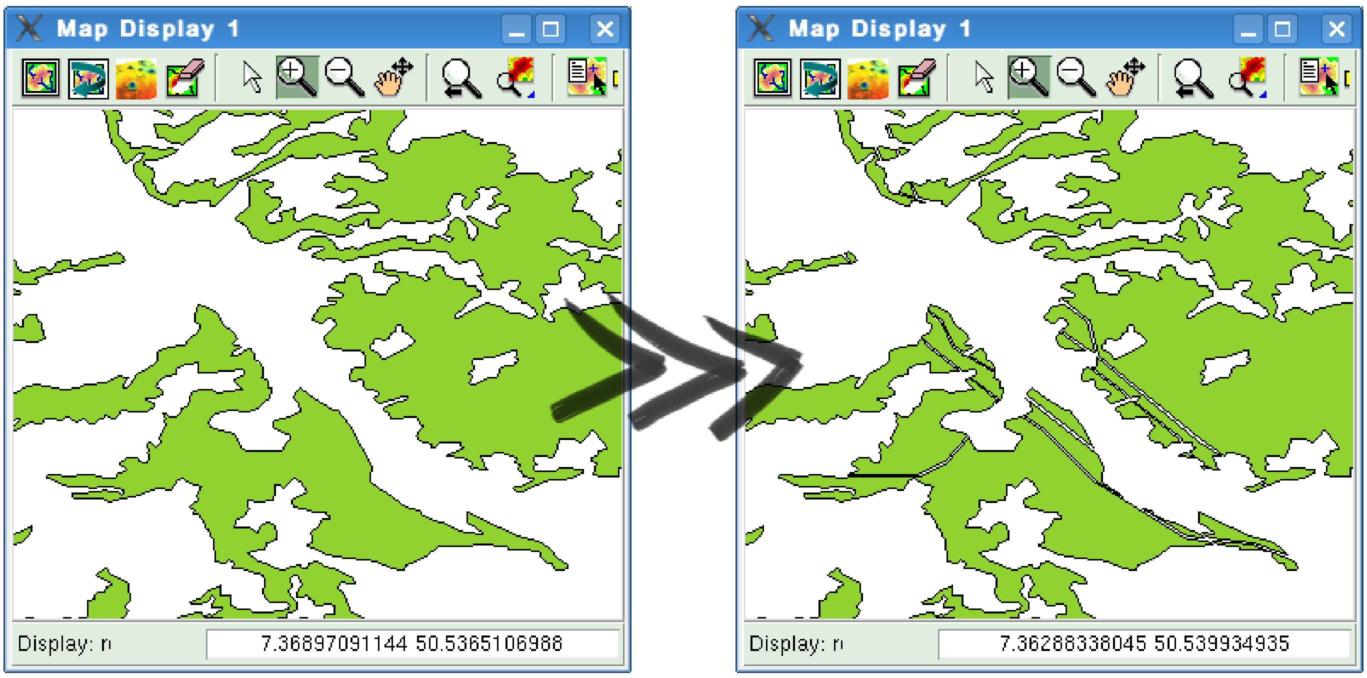
Runways were more complicated, because they only represent - again, as the name says - the runways, but not the surrounding area. So I experimented with **buffering**, which means that you expand the polygons by "inflating" them. I found a usable radius to be about *0.003 degrees*.

Roads and **Railroads** are a bit more complicated. In the raw data they are only lines, which is not very good, when you try to cut out areas in polygons. So I had to "convert" them to polygons. It turns out, that **buffering** is again the way to go. This way, **GRASS** can turn lines into polygons. The usable buffering radius was *0.0003 degrees* for roads, and *0.00025 degrees* for Railroads. In the following screenshot, you can see, how it looked in **GRASS** when I created the buffers.



After all this buffering I could combine all the polygons to one big layer, which turned into my "stencil template" for the late work (this process took a fair amount of computing power with **GRASS**)

Finally I only had to got through all the countries, all of their forest layers, and cut out the "unneeded" areas with the new combined road-railroad-city-airport "stencil template". **BASH** scripts - of course - were a big help in this process. And again, this took some days to complete with **GRASS**. Cutting out roads / railroads in the forest look like this:



Display: r 7.36897091144 50.5365106988

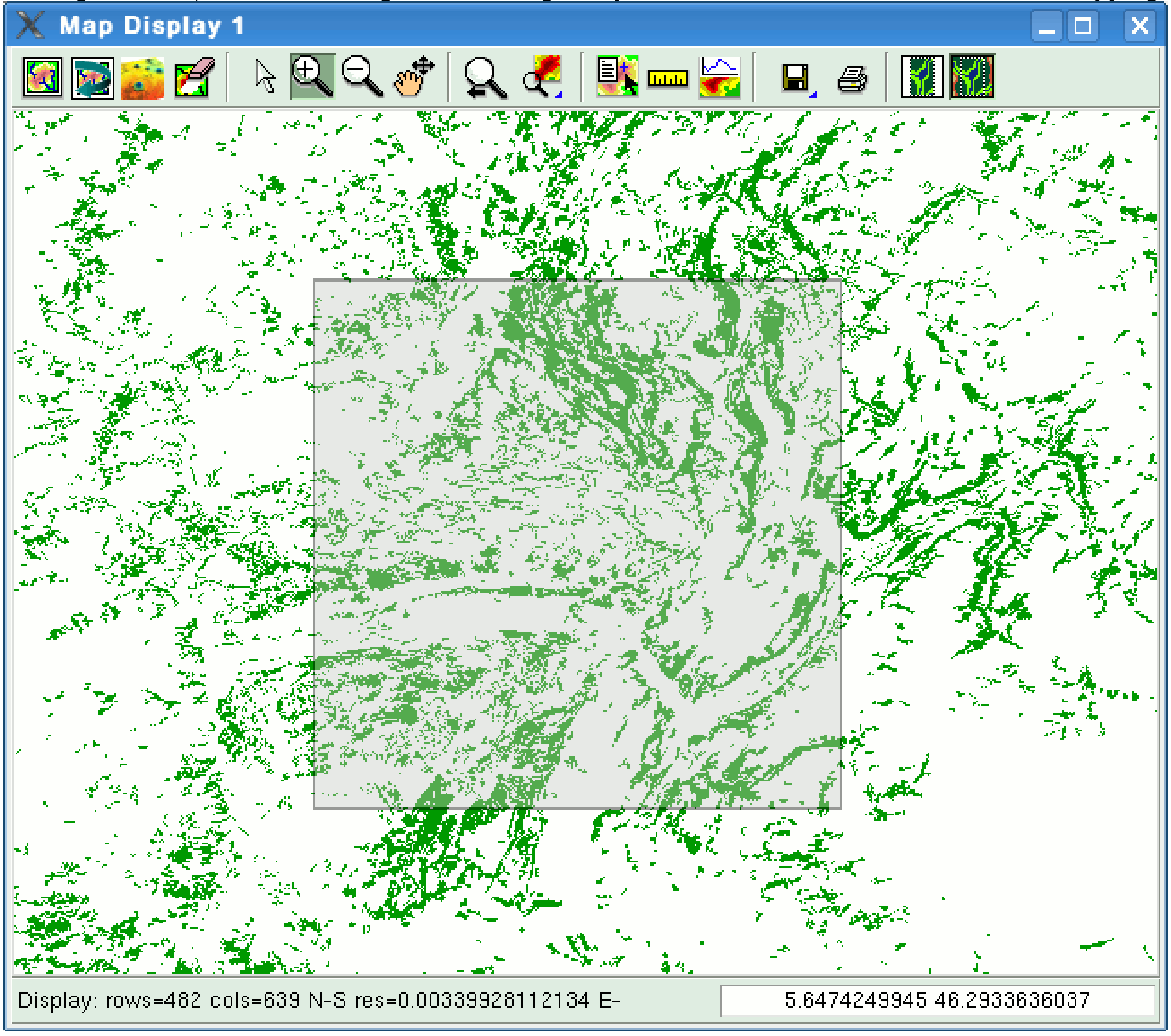
Display: r 7.36288338045 50.539934935

Now the layers were ready for further processing. Though still in split up layers for all countries and all forest types.

Making scenery out of the data

The X-Plane scenery system is working with 1x1 degree tiles, and so are forest overlays. So I had to find a way, to cut out such tiles from the GIS Layers, export them to some text format, transform it to a format which **DSFTool.exe** can convert to DSF files. **GRASS** was again a big help (and some basic Unix command line tools). But how exactly went this process?

1. **GRASS** by default has no clipping command, which lets you extract rectangular areas. BUT it has the v.overlay process, which can combine two layers to a new one with the logical operators **and** / **or** / **not**. And you can create synthetically a rectangular area with the two commands g.region and v.in.region. So using **v.overlay** and combining one **CORINE** layer with this new rectangular area using the **and** operator, I got what I wanted. A clipped part from a forest layer. To speed things up, I also first used v.select on the forest layer, which is much much faster that **v.overlay**, to preselect which polygons will fall in my rectangle (and then only clip this small area to the rectangular area). The following screenshot gives you an idea, what I meant with the clipping:



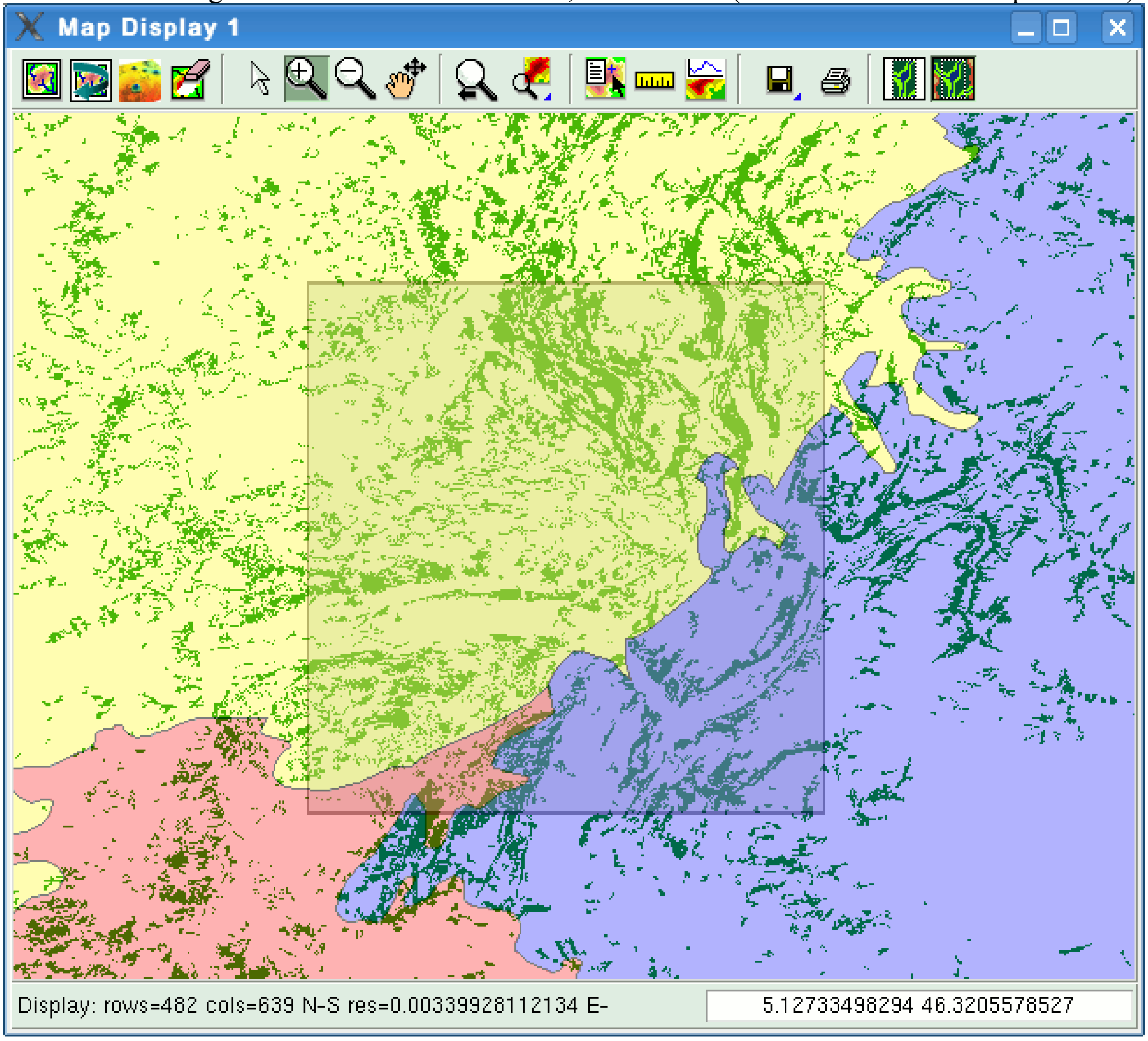
Display: rows=482 cols=639 N-S res=0.00339928112134 E-

5.6474249945 46.2933636037

2. Since **V2** of the forest scenery, there is an additional step involved, which looks in each tile, to which **biogeographic region** the forests in it belong. In some cases, there can be up to 3 (at least this is the highest number I have observed) of them meeting in one scenery tile. So this step further cuts the polygons of one forest type in the different biogeographic types. This way I can export them in the next step, one-by-one, and assign in the DSF scenery different forest definitions to them. This way, if we are in a broadleaf layer, we can get for example:
   - Broadleaf-Continental

- Broadleaf-Mediterranean
- Broadleaf-Alpine

And because images are worth thousand words, here is one (which fits to the example above):



Display: rows=482 cols=639 N-S res=0.00339928112134 E-

5.12733498294 46.3205578527

3. Then I export the newly clipped little forest area to a text format. I had to do some experiments with the available **GRASS** export tools, to find the best format. GML - the Geography Markup Language turned out to be a very good format, because it has an XML like structure, which you can very easily transform with some **grep** and **pearl** commands. And it had an almost identical structure to what X-Planes **DSFTool.exe** awaits as text file when you convert it to **DSF**. **GRASS** supports this through the command v.out.ogr -c input=export type=area dsn=$rawdir/export.gml .
4. Then came some "script magic". For each 1x1 degree tile, I had up to 8 layers - well not always, as it depends on the vegetation - to extract data, and then combine it up into one txt file (for **DSFTool.exe**). The transformation of the **GML** to the necessary **TXT** format went by
   - throwing away most of it with
   - then replacing **GML** keywords with **DSF** keywords
   - rearranging the coordinate values and correctly formatting them
5. Then you have to do this for all the tiles you want to extract.
6. Finally, because sometimes **GRASS** made some very tiny clipping errors (meaning, that some coordinates were out of the 1x1 degree box - which **DSFTool.exe** does not like!) I had to write a clean up script which corrects this problem in the files (again some **BASH** scripting magic).

After all, I had a folder full of **DSF** tiles in **TXT** format. And now I only needed - again a script - to go through them, throw them at **DSFTool.exe**, and sort the resulting **.DSF** files in the usual X-Plane **Earth nav**

data subfolder structure (you can read about it here)

OK, there was also another work. Because the country borders do not align with 1x1 degree tile boundaries, I always had to combine some countrys together, so that I have the data from every side in tiles that cover border areas. Because combining all the country layers from Europe to one big layer (well, still per forest type) would have killed my computer (and it almost did when I tried), I went with smaller work areas. So I always took 5-7 countries, and designed my extract areas the way, that they did not go over borders were I did not have the other side at the moment. And then do this for another country combination, and so on ... Seems a bit complicated, but it isn't (when **BASH** is your friend) and it is much faster in the end, because **GRASS** can work with smaller datasets.

For all the adventurous, here is a small package with the scripts I used in the process (of course you need **BASH** and **GRASS 6.2** for it to work). I recommend the text *how_to_work_with_the_scripts.txt* inside the package:

- http://www.alpilotx.de/Scripts.zip

Designing the forests

With the new **V2 forest scenery** we have introduced the distinction of forests of the same tpye, by their biogeographic location (as described further up on this page). For the definition of the forests, you use the Forest (.for) File Format Specification of X-Plane. This way you have always the ability, to change, how your forests look with only changing forest definitions - and don't have to bother with the **DSF** files (as long as you are happy with the polygons them self).

This time, the combination of all the forest tpyes (now we have 10) and all the biogeographical zones resulted in the wealth of **69 forest definition files** (of course, there are some, which got the same content - but not many). Now, how did we do this. Well, as this was the part - and the textures too - of **Albert Laubi** I let him to say some words on what he has done:

EuroTrees - The Visual Part

If you combine 10 tree layers with 8 climatic zones, you get 80 different forest types. Might look nice - ok, let's plant some trees.

But which ones? And how many? And in which way?
So I began this issue with some biogeographical research. What are the caracteristics of a region? Which are the dominating species there? Which trees will correspond to a given layer? Soon it began to get out of hands. There is such a variety in nature (and its descriptions too!), that one would need the tenfold of forest types at least.

But - are we forest hikers, botanists, tree spotters? (Some might be tree huggers, I admit). No, we are airborne and look to the world from a distance.
My distillation resulted in a residue of 66 plants (there's still some space left!), that would do for our purpouse. If trees form a forest, it's less important to have a bunch of variations than heaps of trees!
Now it was quite a stiff job to find suitable pictures of the essential trees and to work them up graphically for my Euro_trees.png. Thanks to Sergio Santagada whose work is still represented with 1/3 of all the trees.

The crucial point was then the composition of each forest definition, because in the end, this will determine the look of the scenery. But there were no spruce or pine layers to deal with, though these forests look pretty different, but "Conifer" only. And how does "Fruit" look like? Or what do scottish and asturian "Heathland" have in common, since they belong to the same definition? There is so much generalization therein, that you behold the visible forests best as a piece of art. But nevertheless I assure, that when you take off, you'll fly (low!) for hours in this eye-candy-scenery to every nook and cranny of old Europe (Switzerland excluded ;-) - enjoy!

Albert Laubi (xflyer)