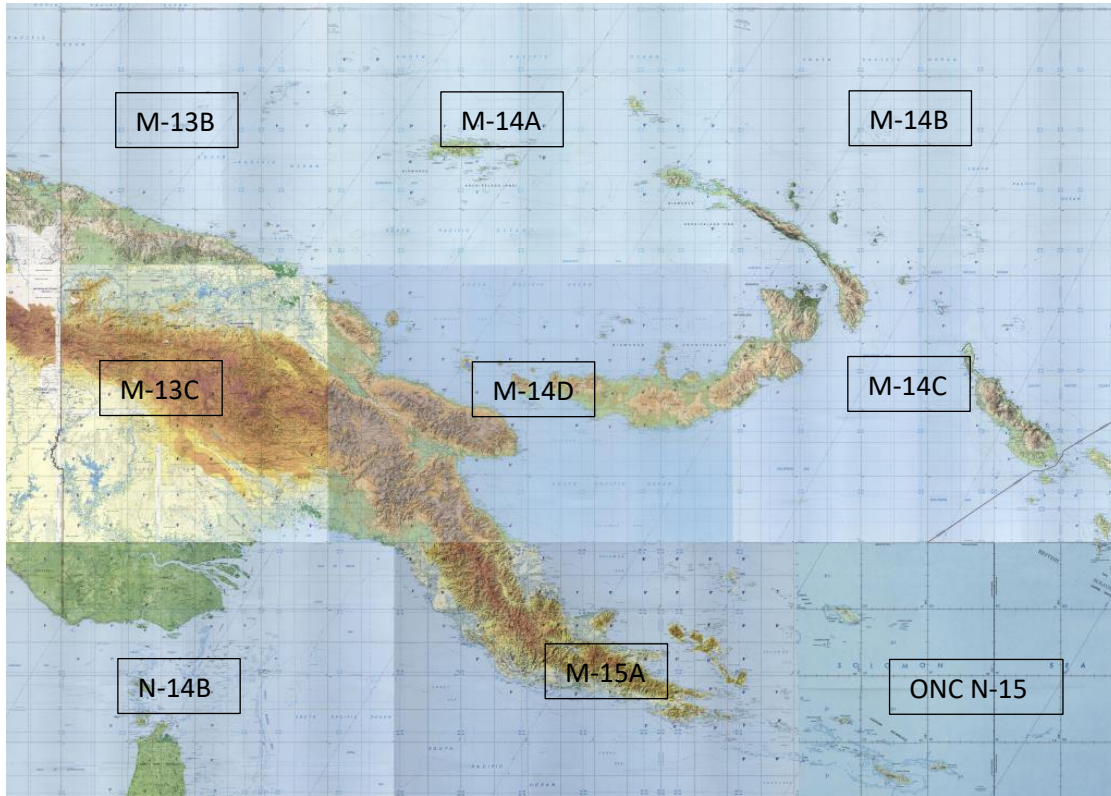# PNG Tiles Project V0.9

04-Apr-2021/bm98

Original PNG Maps are:

"Courtesy of the University of Texas Libraries, The University of Texas at Austin."
→ I assume they are public domain as this was stated in the Universities copyright section.

TO BE USED ONLY FOR PERSONAL ENJOYMENT (and Flight Simulator use may be…)

Derived from the Tactical Pilotage Maps downloaded from the public Texas Library.

- TPC M-13B (1978), M-13C (1979)
- TPC M-14A (1979), TPC M-14B (1976), TPC M-14C (1978), TPC M-14D (1980)
- TPC N-14B (1978)
- TPC N-15A (1998)
- ONC N-15 (1972) for the missing TPC N-15B chart of that area

For processing see below 'Map Processing'.

The tilelive/tessera Tile Server is used to provide tiles for open layers

The tileserver is exposed at port 8081

URL would be: http://host:8081/PNG/{z}/{x}/{y}.jpg

A mapping app is included – to be found at port 8080.
It accepts tracking GET requests at the route /api/track

A HTTP GET url is typically:
http://host:PORT/api/track/LAT,LON,ALT,HEADING,GSPEED,VRATE (Lat,Lon is required, Heading should be true heading)

e.g. http://loaclhost:8080/api/track/12.3,-12.5,2300,256,120,-300

See below when using a Docker Image how to map ports away from the default 8080/8081.
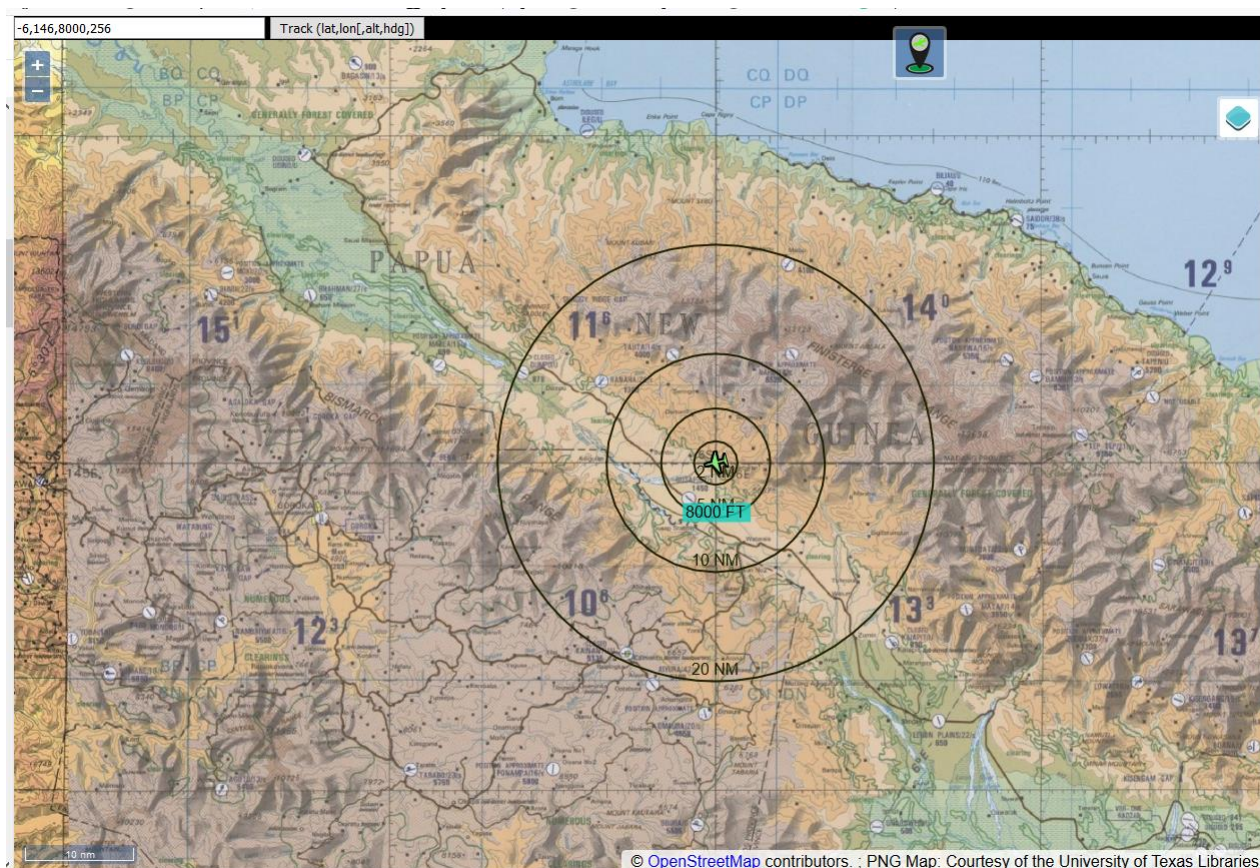
You may enter the tracking manually in the input box top left and hit the Track Button to locate.

The Tracking Icon toggles when clicking it

- Green moves the map to center the tracked location
- Red moves the tracked icon around (and outside..) the map

NOTE: you need to have tracking OFF (red) to move the map around, else it always snaps back and centers the last tracking location.

 – or having a program that collects such data from the flightsim and submits it as GET request to the server, all connected aps are retrieving the same information to track the same location.

The Layer Icon to the right provides means to en/disable and change the shown layers

Browsing should work with more recent Tablets as well.
More recent means the browser understands Javascript of recent spec.
(i.e. iOS 9.3 from an old iPad does NOT ..)

The web app includes layers for OSM Streetmap and the Samen Terrain Map.
If you own or want to use BING maps, get a key and add the key in the appconfig.js file in the src folder after installation of the software.
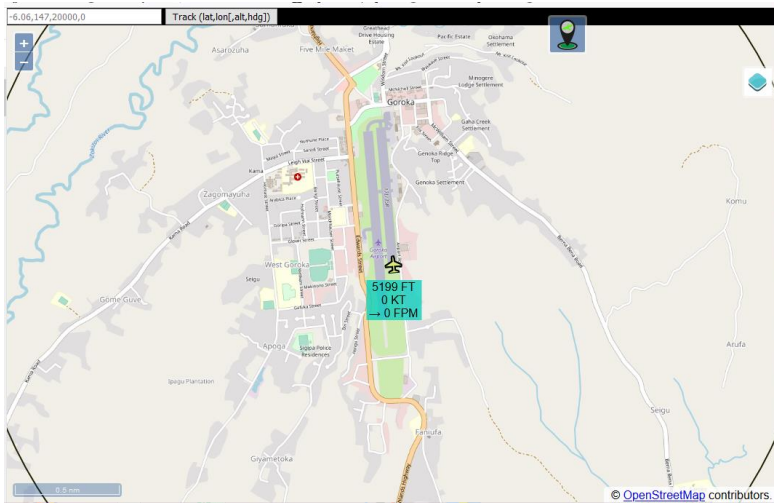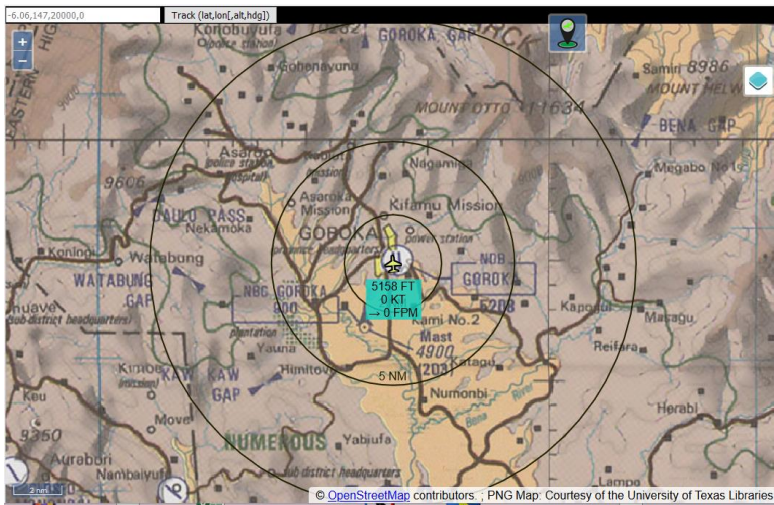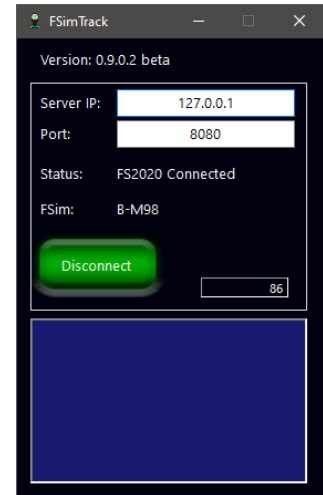No Google Map layer as I don't have a key for it..
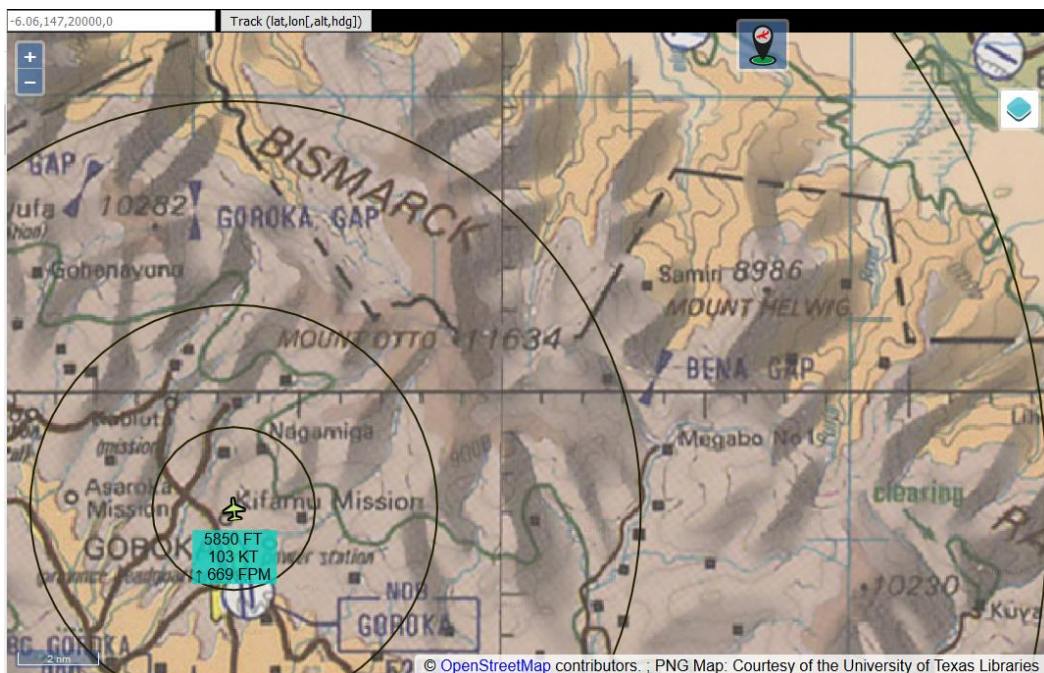
Tracker App

You may find a FS2020 (no XPlane yet) Tracking app here:

Link: https://github.com/bm98/FSimTrack

At RW35R – Zoom to OSM Detail where more details are provided

En route

Using the Docker Image

When using a NAS e.g. Synology – easiest way to get there

Install the Docker appliance and open its configuration GUI.

Get the Docker image **bm98ch/fsimpngtiles** from the Docker registry

Define the mounted Volume and map the Volume to a NAS share directory.

The container path is: `/usr/src/app/www/src`

In general on the NAS you may want to map ports away from 8080/8081 as they are likely already used by some other appliance. Ports between 49152 and 65535 are not reserved and usually safe to use.

Deploy **appconfig.js** from the GitHub src Folder into that share directory

Modify **appconfig.js** to have the

`MapTileService=` part the IP Address of the NAS

Browse to your NAS IP and the port mapped while configuring the Image

E.g. my NAS is at 192.168.1.100

`MapTileService=http://192.168.1.100:49081/`

Note the slash at the end – it's needed…

When using Docker Desktop

Get **bm98ch/fsimpngtiles** (.1.1 as of writing)

Run the image and open the Optional Setup part

Map Port 8080 and 8081 to your preference.

You need to click the plus sign right of the first port map to get to the second one.

Map the **src** folder to a folder on the host system, it needs to contain only the **appconfig.js** file.

The container path is:
`/usr/src/app/www/src`

Deploy the **appconfig.js** file from Github into the **src** folder above.
Adjust at least the entry for the TileService e.g.:
`MapTileService= http://`==**192.168.1.69**==`:`==8081==`/`

Point the IP address to the machine where the Docker image is running, the port is the host port mapped to Container Port 8081 above.
The best is to define the host IP address else a client such as a tablet would not get tiles served if the host is declared as localhost or a loopback address.

Run the Container …

Get your browser to `http://host:8080/` where 8080 would be the host port mapped to the Container Port 8080 above

Running without Docker

Harder but doable.

Install NodeJS (https://nodejs.org/)

Load the Github content to a folder

In this folder find and right click and "Run with Powershell" `server-install.ps1` this should download the need files from the Node registry and you have now a folder `node_modules` containing a lot of files to not touch.

Sometimes the downloader complains about outdated things but leave it as is, if it finishes. It creates the file `package-lock.json` which is not in the initial fileset.

Find and extract PNGmap.zip in **tiles** creating the folder **PNGmap** containing folders 6/..12/ and some files
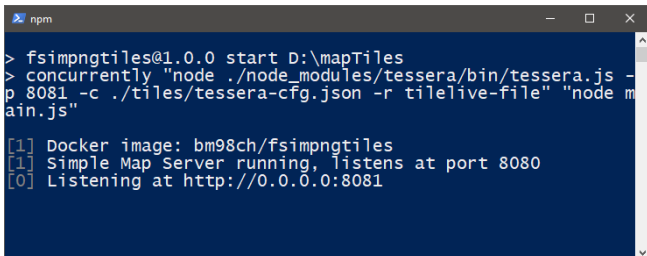Note: there are >35'000 tiles contained in that Zip so it may take a while to extract.

Finally it should have a layout as shown to the right.

Head to **www\src** and modify **appconfig.js** to have the line:

`MapTileService=http://`==192.168.1.100==`:8081/`

Edit the IP address to match the machine IP if you want to access from other devices else localhost is good enough.

Right click and "Run with Poweshell" `run-server.ps1` a shell window should pop up and now show something like:

```
> fsimpngtiles@1.0.0 start D:\mapTiles
> concurrently "node ./node_modules/tessera/bin/tessera.js -
p 8081 -c ./tiles/tessera-cfg.json -r tilelive-file" "node m
ain.js"

[1] Docker image: bm98ch/fsimpngtiles
[1] Simple Map Server running, listens at port 8080
[0] Listening at http://0.0.0.0:8081
```

Now head your browser to http://localhost:8080/

Note: if "Run with Powershell" is not available go to Win Search and type **PowerShell** and hit Enter, it should open a terminal like window.
CD to your folder and type `serv` and then the Tab key – should get you
`.\server-install.ps1` hit enter to run it, same with run-server.

To stop it - Click the terminal and hit Ctrl-C and the confirm with Yes (in your Windows language it might be Ja or Oui or...) the Window should close and the server is no longer running.

Also be aware that the tile server shows a log of all served tiles in that window so it will be pretty busy, just minimize it if it gets too annoying.
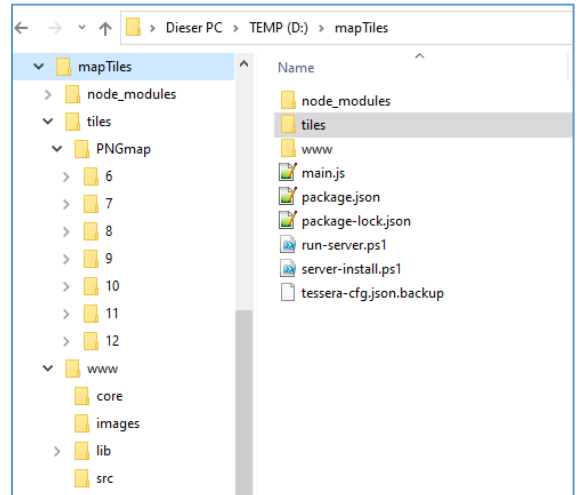
Map Processing

Processed using **QGIS Desktop** contained in OSGeoLive (https://live.osgeo.org) virtual machine the osgeolive-13.0-amd64 on Windows10 Oracle VM VirtualBox

Main process is:

- Cutting the frames from the original map images
- Apply image processing to the original map images before starting the next step
- Georeference all to WGS84 – select and use the Georeferencer Plugin
  Best done by using the coordinate grid on the map and sample about 20 or so points covering the complete map
- Combine in QGIS Desktop and save as GeoTiff file with WGS84 (EPSG:4326) (huge.. 1.6 GB)
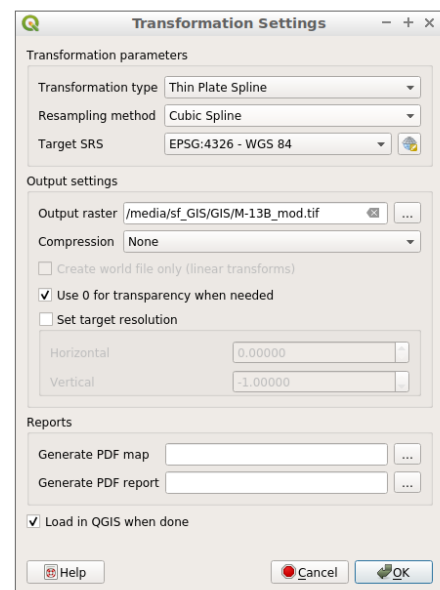- Tile converting using MapSlicer (also contained in the VM)

QGIS Desktop

Contained QGIS V 3.4.10

Note: All attempts to update failed – one would need to start from scratch or wait for a new OSGeoLive Distribution.
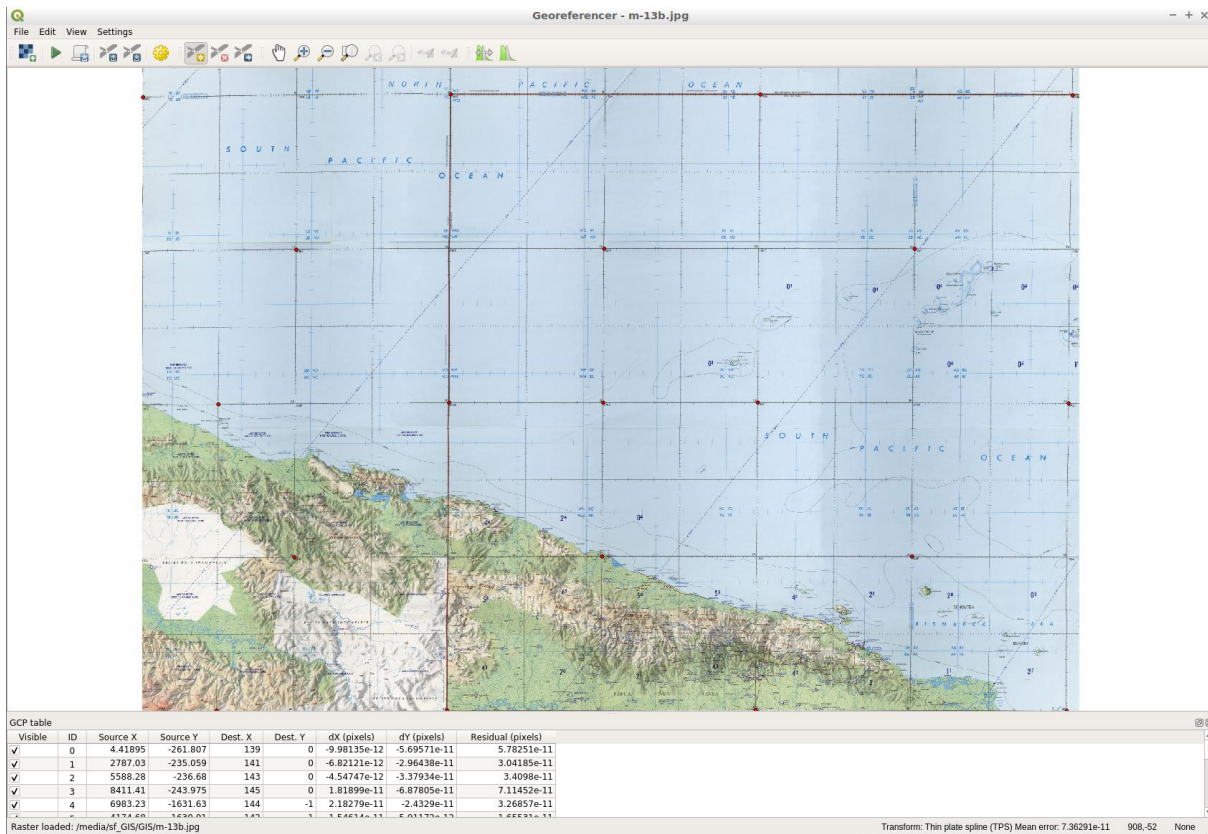
Enable the GeoReferencing Pluging
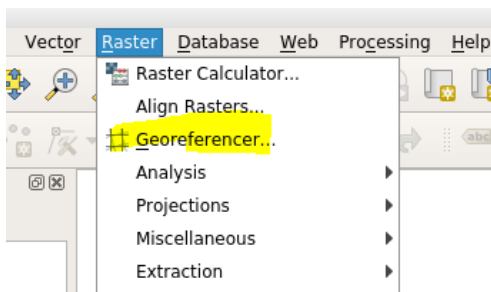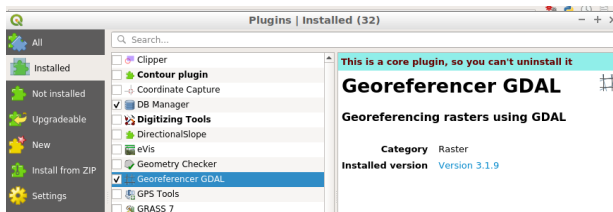
In Georeferencer set the Transformation Setting per map:

Georeferencing… Add a RasterMap from the menu (the jpg map image)

Add Points and assign the coordinates

When finished adding reference points; Use File – Start Georeferencing to complete

The map should appear in the GeoDesktop as new layer.

Arrange the z-order to get an appropriate overlapping where needed (slide layers up/down)

Use the XYZ Tiles plugin to create the tiles for the server.

To make an image from the complete map see below
I found that it has it's limits slightly beyond the output image size used in the example (with the QGIS Desktop V 3.4 on that VM image – may be not an issue with a more current version.

In GeoDesktop create a combined GeoTIFF:

Make it all visible then File – Export – Map to Image

Set Resolution to 200..300 dpi, Draw on Canvas and select the output area.
Then check that the output width and height is about the sum of the input map files.
Save with a new name – one should have a georeferenced TIFF file that is ready for slicing and tiling.

Lower DPI when the prog complains about not enough memory.
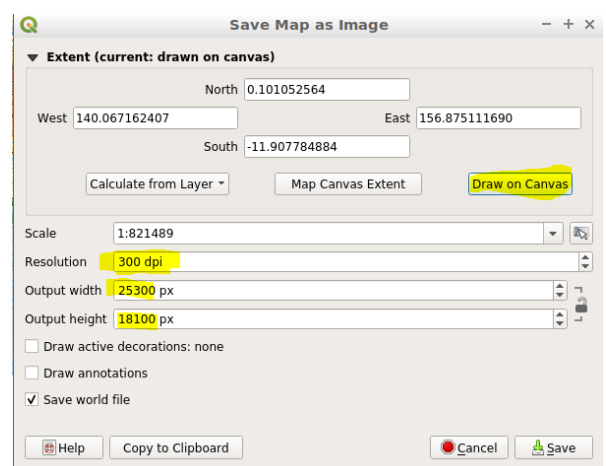
XYZ Tiles Plugin

In QGIS Desktop Plugin Manager find and load XYZ Tiles plugin

Set the desired map Scale e.g. 1:750'000 or smaller depending on the area and source map

Set the Magnifier to e.g. 15% to have the complete map visible

Show the Toolbox – XYZ Tiles is may be the last entry in the list

Double click to open it.
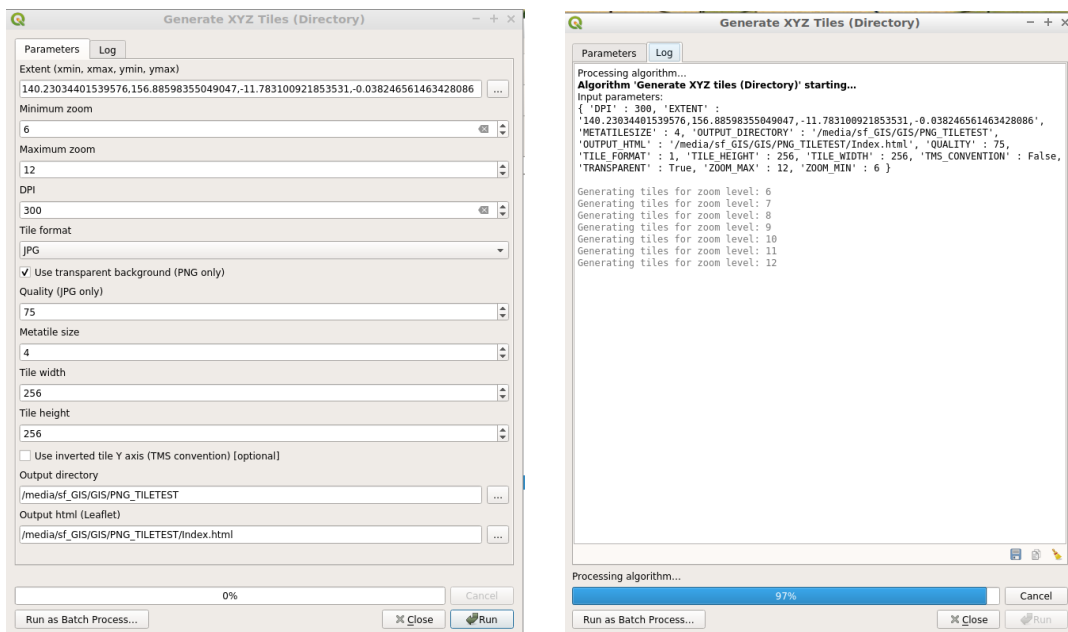
Select the canvas extent of the mapping area

Select the desired Zoom levels (~6..12) depending on the resolution of the input material

Select the DPI (300) depending on the input material

Select the output file type JPG provides much smaller tiles than PNG

Select an output folder to save the tiles to

Hit Run and wait for completion.

COPY and paste the Extent to a file as it is used for the tessera config file later

metadata.json:

```
{
    "tilejson": "1.0.0",
    "name": "PNG Map",
    "description": "PNG from Tactical Pilotage Maps",
    "version": "1.0.0",
    "scheme": "xyz",
    "minzoom": 6,
    "maxzoom": 12,
    "bounds": [ 140.170,-11.821,156.960,0.126 ],
    "format": "jpg"
}
```

Zoom levels as created

Bounds: Xmin(West), Ymin(South), Xmax(East), Ymax(North)

Format: jpg else tessera assumes PNG

MapSlicer:

Alternative to XYZ Tiles with much better rendering but also more work is required to do the Job..

Hit Rendering and wait…

Level 12 and 11 takes quite a while to complete, creates >35'000 tiles…

Note:

The created viewing html does not work – outdated or otherwise wrong.

Changed the proposed detail from 3..10 to **6..12** (more detailed >12 creates HUGE sets, less detailed from 3..5 does not make any sense as no detailed are to be seen anyway).

MapSlicer produces a tile set with TMS orientation – upside-down what Google Maps (XYZ type) is using.

So some trickery must be applied when using the tiles and tile services in order to cope with the inverse orientation.

Or apply the script below to get a XYZ Tileset: https://gist.github.com/kannes/ebfe021458f96e4f30b5

Note: don't use that script as is as it does overwrite files around the equator – use the one below

```
#!/bin/bash
# rename TMS tiles to the XYZ schema
# no quoting, since all files have simple numeric names
# do not run this anywhere else than INSIDE your tiles directory

# run it like this: find . -name "*.png" -exec ./tms2xyz.sh {} \;

filename=$1
```

```
tmp=${filename#*/}        # remove to first /
z=${tmp%%/*}              # remove from first /

tmp=${filename%/*}        # remove from last /
x=${tmp##*/}              # remove to last /

tmp=${filename##*/}       # remove to last /
y=${tmp%.*}               # remove from first .

extension=${filename##*.}

let newy="2**$z-$y-1"   # calculate the xyz tile
#echo $z $x $y $newy $extension
newfile=./$z/$x/$newy.$extension
# echo $newfile
# if the file already exists - rename to *.NEW else we overwrite unproc. ones at worst
if [  -e $newfile  ]
then
        mv ${filename} ${newfile}.NEW
        echo Cleanup ${newfile}.NEW
else
        mv ${filename} ${newfile}
fi
```

To Cleanup rename all *.NEW files by removing the .NEW extension

appconfig.js for reference

```javascript
"use strict";
// ---------------------------------------------------------
//
// (c) 2021 Martin Burri
// MIT License - no warranties whatsoever...
// https://github.com/bm98/FSimPngTiles
//
// This file is to change the configurable settings.
//   overwrites ./core/config.js settings here
//
// ---------------------------------------------------------

// The server for any raster tiles -
// Set to the IP where the service is running - either your PC, or a NAS IP when running as Docker image
MapTileService= "http://192.168.1.69:8081/"

// -- Map settings -----------------------------------------
// The maps initial zoom level, 0 - 16, lower is further out
DefaultZoomLvl  = 10;

// Aircraft marker distance rings (default only)  [set true or false]
AircraftIcon = true; // true to start with show icon (only shown if the center marker is shown)
SiteCircles = true; // true to start with show circles (only shown if the center marker is shown)

// Circles around the aircraft icon shown in the map
// Numbers are the Radius and in nautical miles
// default: SiteCirclesDistances = new Array(2, 5, 10, 20);
SiteCirclesDistances = new Array(2, 5, 10, 20);
// Outline color for distance rings
OutlineRingColor = '#202000';
// Outline color for aircraft icon
OutlineAircraftColor = '#000000';

// Provide a Bing Maps API key here to enable the Bing imagery layer.
// You can obtain a free key (with usage limits) at
// https://www.bingmapsportal.com/ (you need a "basic key")
//
// Be sure to quote your key:
//   BingMapsAPIKey = "your key here";
//
```