ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

Photoscenery for Realistic Scene Generation and Visualization in Flightgear: A Tutorial

Srikanth A[1], Indhu B[2], L Krishnamurthy[1], VPS Naidu[3]

Dept. of Mechanical Engineering, NIE, Mysore, India[1]

Dept. of Instrumentation and Control, NIT, Trichy, India[2]

Multi-sensor Data Fusion (MSDF) Lab, CSIR-NAL (National Aerospace Laboratories), Bangalore, India[3]

ABSTRACT: Realistic scene visualization is an essential part of flight simulation and training. In this tutorial, an approach for generation of photo realistic scenery for visualization in the FlightGear (Flight Simulator) has been presented. The realistic scene integrates orthoimagery and the terrain data obtained from a digital elevation model (DEM). The orthoimagery is overlaid as a texture on the terrain to obtain the PhotoScenery. Using this approach highly realistic scene can be accomplished.

KEYWORDS: FlightGear, Flight Simulation, Scene Visualization, Photo Realism, PhotoScenery

I. INTRODUCTION

FlightGear Flight Simulator (FGFS) is an open source flight simulator that has been used in a range of projects across the academia and industry. FGFS has been used as a tool for simulation, modelling and visualization of aerial vehicles. FGFS has been used for training the pilots on aerial vehicles under different simulated scenarios. FGFS can also be used in the development of autopilots for testing and validation. The default FlightGear installation includes scenery for few airports and locations and is not realistic (it renders only generic texture) and hence an attempt is made to develop a realistic scene. This can be achieved by overlaying the *orthoimage* on terrain (obtained from World Scenery). The objective is to establish a standard routine for generating *PhotoScenery*, which should be both open and easy to use. PhotoScenery [1] is a scenery in which orthoimagery is placed over the terrain.

In this tutorial, generating PhotoScenery for use with FlightGear knowing the latitude and longitude of any location of interest has been described. The generic scenery is obtained for the geographical area of interest from the *World Scenery 2.0* [2] project. The scenery data files can be obtained [3] by clicking on the 10x10 degree graphical interface for the location of interest. World Scenery already integrates the worldwide elevation data from *Shuttle Radar Topography Mission* (*SRTM*) and worldwide airport scenery, thereby simplifying the steps required. They need to be installed into the $FG_SCENERY directory [4].

II. REALISTIC SCENE GENERATION

In order to generate the realistic scene, one has to know the geographic co-ordinates of the location of interest. For example, the geographic co-ordinates of National Aerospace Laboratories (NAL) are `12.9610589°N` and `77.6532198°E`. Using the python script "`fg_latlon_to_tile.py`" (see appendix I); corresponding tile index and tile bounds of the FlightGear tile has been calculated by entering the following command at the terminal.

```
> python fg_latlon_to_tile.py 12.9610589 77.6532198
```

Executing the above script, the following information will appear in the console as:

```
Tile number:  4217277
Tile bounds:  [12.875, 13.0, 77.625, 77.75, 12.9375, 77.6875]
```

The tile number is used as the filename for saving PhotoScenery, the procedure for which will be described in the following section II.A. The tile bounds indicate [minLat, maxLat, minLon, maxLon, centerLat, centerLon] values.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

Once the tile bounds are obtained we need to obtain an orthoimage spanning exactly the bounding box defined by the corners (minLat, minLon) i.e., `(12.875, 77.625)` and (maxLat, maxLon) i.e., `(13.0, 77.75)`. The resolution of this orthoimage will dictate the quality of the rendered PhotoScenery. The resolution of the orthoimage has to be a power of two (i.e., `2048x2048`, `4096x4096`, etc), but the maximum value is only limited by the capability of the graphics hardware.
The realistic scene generation process can be described as a three step procedure:

1. Obtain orthoimage
2. Convert orthoimage into PhotoScenery
3. Integrate PhotoScenery with FlightGear

A. Obtain orthoimage

According to Wikipedia – An orthoimage is an aerial / terrestrial photograph which is geometrically corrected (orthorectified) so that the scale is uniform and the image is free from lens distortion and camera tilt nonlinearities. An orthoimage can be used to measure actual distances, since it offers a spatially accurate representation of the Earth's surface. Some pre-processing such as removing haze and shadows might be required depending on the source used to obtain the orthoimage [1].

The following method has been established to obtain the orthoimage using the ArcGIS REST API [5]. It has to be noted that ArcGIS REST API limits the maximum size of the image that can be retrieved to `4096x4096`. However, individually retrieving portions of a given tile at `4096x4096` and later stitching them together to form a single tile gives higher resolutions to that tile. This can be done by using any raster graphics editing software like GNU Image Manipulation Program (GIMP) [6]. For the case of our example we obtained the orthoimage for the NAL area using ArcGIS REST API from the following URL:

`http://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/export?bbox=77.625%2C+12.875%2C+77.75%2C+13.0&bboxSR=4326&size=4096%2C+4096&imageSR=4326&format=png24&transparent=false&f=image`

B. Convert orthoimage to PhotoScenery

The orthoimage, which is in the Portable Network Graphics (PNG) format, should be converted to DirectDraw Surface (DDS) format for use with FlightGear. DDS is used since it stores data in a compressed format, reducing the storage space required. DDS can be decompressed in real-time by most Graphics Processing Unit (GPU) hardware, thereby having no performance impact.

GIMP [6] image editing program has been used for converting the image from PNG format to DDS format. The GIMP DDS plugins [8] must be installed before the conversion begins. The following procedure is adopted:

1. Start GIMP by clicking on its icon.
2. Open the orthoimage obtained by following the procedure described in “section II.A”. [Select File -> Open]
3. Flip the orthoimage in the vertical direction. [Select Image -> Transform -> Flip Vertically]
4. Scale the image if required. The image size must be a number which is a power of two [Image -> Scale Image -> Enter Width and Height as 4096 or higher number that is a power of two]
5. Export the image to ‘.dds’ format. [Click Export as -> Select DDS image (*.dds) from the dropdown list at the bottom right]
6. Save the exported image with the filename as the tile index i.e ‘`4217277.dds`’ (The filename should be the tile index which is obtained from the python script described in “section II, paragraph 1”).

C. Integrate PhotoScenery with FlightGear

FlightGear will not support PhotoScenery by default. As both SimGear and FlightGear-data has to be patched for PhotoScenery support. To do this, one needs to build FlightGear from source [9]. Store the ‘`.dds`’ image obtained by following the steps in “section II.B” under the FlightGear Scenery folder along with the ‘`.stg`’ file of the same name. For this example, store the file at “$FG_SCENERY\Terrain\e070n10\e077n12\4217277.dds”.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

Finally we can verify that the PhotoScenery is being loaded by FlightGear and overlaid on the scene. Start FlightGear with the following command. When FlightGear is started, it automatically loads this PhotoScenery and overlays it on the scene.

```
fgfs --fg-root==path_to_fgdata-2.8.0 --lat=12.9610589 --lon=77.6532198 --altitude=5000 --fdm=external
```

III. RESULTS

The results of the FlightGear Flight Simulator with PhotoScenery are illustrated in the following screenshots. The following screenshots have been taken from the simulated camera view at an altitude of 2000 m. Snapshots in Fig-1 are taken with field of view (FOV) set to 24.2°, while snapshots in Fig-2 and Fig-3 are captured with FOV set to 55°.
Fig-1 shows the area around NAL. As it can be seen in Fig-1a the scenery only consists of some generic land cover textures. Fig-1b shows the same area with the actual PhotoScenery generated from the satellite orthoimagery. We can observe in Fig-1c that using a higher resolution of `8192x8192` gives us a much clearer image.

The terrain elevation data from SRTMv3 is integrated with the PhotoScenery which provides a resolution of 90m for most of the world. It can be observed in Fig-2a that the default scenery shows a road on the scene. But in reality the road does not exist in the actual scene as can be perceived in Fig-2b.

In Fig-3 the default airport layout scenery is still rendered on top of the PhotoScenery as can be envisaged in Fig-3b which may not be desirable.

Fig-1a FlightGear default scenery for NAL (tile size 4096x4096)

Fig-1b PhotoScenery for NAL (tile size 4096x4096)

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

Fig-1c PhotoScenery for NAL (tile size 8192x8192)

IV. ADVANTAGES AND LIMITATIONS

The described approach has several advantages as follows:

- PhotoScenery shows the real satellite imagery, rather than trying to emulate the real world using generic textures.
- The elevation data is smoothly integrated with the PhotoScenery improving the visuals.
- Areas using PhotoScenery blend realistically, instead of a hard cut between urban and forest, or between water and land.

Some of the limitations of this approach are:

- Higher resolutions of orthoimagery are required for low altitude flying, 1m/pixel can be considered the absolute minimum. But it requires decent graphics hardware.
- Orthoimagery is often hard to obtain, especially under an open license.
- The default airport scenery is rendered over the PhotoScenery, which may not be desirable as shown in Fig-3a & 3b.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

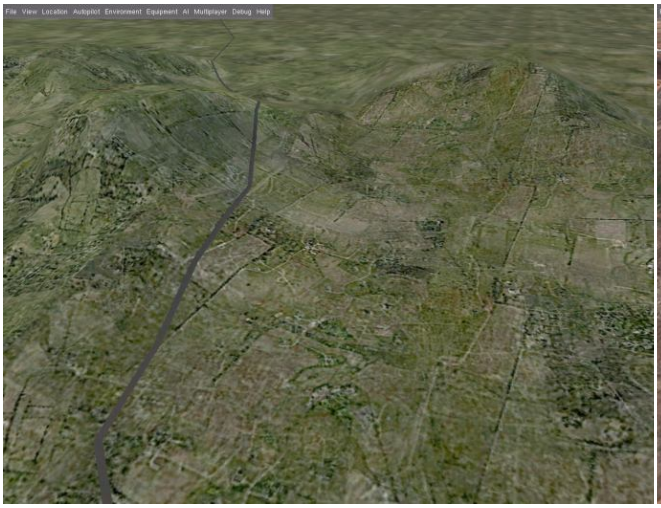

Fig-2a FlightGear default scenery for Nandi Hills (tile size 4096x4096)

Fig-2b FlightGear PhotoScenery for Nandi Hills (tile size 4096x4096)

Fig-3a FlightGear default scenery for HAL Airport (tile size 4096x4096)

Fig-3b FlightGear PhotoScenery for HAL Airport (tile size 4096x4096)

V. CONCLUSION

A routine has been developed for generating realistic scenery for use with FlightGear (Flight Simulator). The scenery using a tile size of `4096x4096` pixels provides a spatial resolution of approximately 3.42 m/pixel which is good enough for medium and high altitude flying. But for smaller FOV as well as lower altitudes when flying drones, MAV or UAV a resolution of `8192x8192` or higher could be used which will provide sharper scene at a minimum resolution of 1.71 m/pixel. The maximum size is only limited by capabilities of the graphics hardware. Further research needs to be carried out before the visualization of the scenery can be implemented at a large scale.

ACKNOWLEDGMENT

Many credits go to the FlightGear project developers working from different parts of the world for developing such a wonderful piece of software. We should also thank Esri (Environmental Systems Research Institute) for providing open access to satellite imagery through its ArcGIS REST API without which this project would not have been successful.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

REFERENCES

[1] Photoscenery - FlightGear wiki, source: http://wiki.flightgear.org/Photoscenery accessed on October 9, 2014.
[2] FlightGear World Scenery 2.0, source: http://www.flightgear.org/tours/world-scenery-2-0/ accessed on October 9, 2014.
[3] Getting the World Scenery for FlightGear, source: http://www.flightgear.org/download/scenery/ accessed on October 9, 2014.
[4] How to Install scenery – FlightGear wiki, source: http://wiki.flightgear.org/Howto:Install_scenery accessed on October 9, 2014.
[5] ArcGIS REST API – ArcGIS Services – Export Map Reference, source: http://services.arcgisonline.com/arcgis/sdk/rest/02ss/02ss00000062000000.htm accessed on October 9, 2014.
[6] GIMP 2.8, source: http://www.gimp.org/ accessed on October 9, 2014.
[7] ArcGIS ExportMap (WorldImagery), source: http://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/export accessed on October 9, 2014.
[8] GIMP DDS plugin, source: https://code.google.com/p/gimp-dds/ accessed on October 9, 2014.
[9] PhotoScenery for Realistic Visualization in FlightGear: A Tutorial, MSDF Report.
[10] fgphotoscenery, source: http://fgphotoscenery.square7.ch/ accessed on October 9, 2014

Appendix I [fg_latlon_to_tile.py] (Adapted from [10])

```
#!/usr/bin/python

import sys

def tileWidth(lat):
      tiletable=[[0,0.125],[22,0.25],[62,0.5],[76,1],[83,2],[86,4],[88,8],[89,360],[90,360]]
      for i in range(len(tiletable)):
          if abs(lat)>=tiletable[i][0] and abs(lat)<tiletable[i+1][0]:
              return float (tiletable[i][1])

def coordinatesFromTileIndex(tileindex):
     base_x    = (tileindex>>14) - 180
     base_y    = ((tileindex-((base_x+180)<<14)) >>6) - 90
     y         =  (tileindex-(((base_x+180)<<14)+ ((base_y+90) << 6))) >> 3
     x         =  tileindex-(((((base_x+180)<<14)+ ((base_y+90) << 6))) + (y << 3))
     tilewidth = tileWidth(base_y)
     return [(base_y + 0.125 * y), base_y + 0.125 * (y+1), base_x + x * tilewidth, base_x + (x+1) *
tilewidth, 0.5 * (base_y+0.125*y + base_y + 0.125*(y+1)), 0.5 * (base_x + x * tilewidth + base_x + (x+1) *
tilewidth)]

def tileIndexFromCoordinate(lat, lon):
     import math
     base_y    = math.floor(lat)
     y         = int((lat-base_y)*8)
     t ilewidth = tileWidth(lat)
     base_x    = math.floor(math.floor(lon/tilewidth)*tilewidth)
     if base_x<-180:
        base_x=-180
     x         = int(math.floor((lon-base_x)/tilewidth))
     t ileindex = int(((int(math.floor(lon))+180)<<14) + ((int(math.floor(lat))+ 90) << 6) + (y << 3)
+ x)
     return tileindex

def main(argv):
     lat = float(argv[1])
     lon = float(argv[2])
     print 'Tile number is: ', str(tileIndexFromCoordinate(lat, lon))
     print 'Tile bounds are: ', str(coordinatesFromTileIndex(tileIndexFromCoordinate(lat, lon)))

if __name__ == "__main__":
     main(sys.argv)
```