Quick guide for the use of the last version of the scripts

This is only a quick guide. There are several ways of using the scripts. Open Basics_poster.ppt to have a wider view. More details can be found in details.doc

If you want to use multiple tracks on your project, you should read also **Multi tracks scripts.doc**.

NOTE: In this document it is always assumed when using octave commands that the path of the scripts has been previously added to octave's path:

> ***addpath('f:\project\scripts')***

For this version of the scripts you need Octave **3.2.3**

a) Download the last folder's structure (last version of octave scripts is included)

http://btbtracks-rbr.foroactivo.com/tutoriales-f31/zaxxon-s-method-tutorial-t131.htm#171

Step 0. Import kml

b) Use importakml script to import the kml.

Copy the kml file (e.g. ejemplo.kml) inside **s0_import** folder.

Open octave and execute:

> **cd f:\project\s0_import**
> **importakml('example.kml')**

The direction of the kml can be reversed by editing importakl.m and changing a parameter found in the first lines of this file.

A file named f:\project**mapeo.txt** will be created. This files establishes a relationship between the terrestrial coordinates and the BTB coordinates for two points of the terrain. The correspondence could be visually checked using Google Earth and the graphs generated by importakml.

c) Before closing the graphs it is recommended to write down the minimum and maximum coordinates of the track (minimum and maximum in both dimensions, horizontal and vertical, **x** and **z** respectively). Those limits will be useful if we are going to use Google Earth to get elevation data.

Step 1. Getting elevation data

We can retrieve elevation data from **d1)** the seamless server (the best option for USA) or **d2)** from Google Earth through 3DR Builder, BTBlofty or raise_kml.

d1) Once we have downloaded the data from the server, we copy the .hdr and .flt files inside s2_elevation folder and we type (using the right file names, of course):

> cd f:\project\s2_elevation
> **leer_gridfloat('ned_03263284.hdr','ned_03263284.flt');**

The result of this action will be a file called **lamalla.mat** inside s2_elevation\salida folder. lamalla.mat depends upon mapeo.txt, so if mapeo.txt is changed, lamalla.mat should be generated again.Jump to section e)

d2) We will generate a coordinates grid whose elevation is going to be retrieved with 3DR Builder or BTBLofty (values should be changed to fit each track):

> cd f:\project\s2_elevation
> **make_grid(-1800,1800,-1500,1500,25,5000)**

this will create a grid of points separated 25m with a extension from x=-1800 to x=1800 and from z=-1500 to z=1500. The road should be completely covered by that grid. It is recommended to use a safety margin (100m, for example) to avoid problems.

If you are going to use BTBLofty, it will be necessary to split the grid points into several files. That can be made adding another parameter to make_grid specifying the number of points per file. 30000 is the limit for BTBLofty, but it is better to use 5000 points because bigger values make the process slow down. If you are using the free version of 3D Route Builder 500 points/file is the limit. Inside the "salida" folder a few files named **gridXXX.kml** have been created. Open them with 3D Route Builder or BTBLofty to get the elevation of the points. You can also run give them elevation calling **raise_kml** script from s2_elevation, but read *raise_with_python* document before trying to use it. If you use BTBLofty/3DRB, once you have the data save it inside "salida" folder as **gridXXX_relleno.kml**.

Now we run:

> **read_grid**

A file called **lamalla.mat** will be created inside s2_elevation\salida.lamalla.mat depends upon mapeo.txt, so if mapeo.txt is changed, lamalla.mat should be generated again. Just call read_grid again, that's all.

The same steps must be done inside s2_elevation_b, using a bigger grid, enough to

cover all the terrain you plan to create. Separation parameter for the points for this grid can have a bigger value if compared with the one created inside s2_elevation.

NOTE: the last version of importakml (from revision 32) checks if ..\mapeo.txt exists. If not, ..\mapeo.txt is created. Otherwise the existing one is used.

NOTE: as far as I know BTBLofty doesn't work with Vista nor Windows7.

Step 2. Road anchors generation

e) Run:

> cd f:\project\Venue
> **btb06(5,1)**

The parameter for btb06 is the road width in meters. Second parameter must be 1 the first time you call btb06. Files **nac.mat** and **anchors.mat** have been created inside f:\project\ folder, and **nodos.mat** and **porcentajes.mat** inside f:\project\Venue folder.

Step 3. Raising the road

dar_altura is a script that raises the road trying to fit existing terrain elevation data.

corregir is a script that changes terrain elevation data trying to fit the road elevation profile

f1) If we want to raise the road according to the terrain elevation data we should execute:

> cd f:\project\s3_road
> coge_datos
> creartrack1
> dar_altura(25,0.15,-0.15)

25 is a smoothing factor (always odd. The bigger, the smoother in height the road will be) and the two values 0.15 and -0.15 are the maximum and minimum slopes allowed to the road. It is recommended to test different values, observing in the graphs how the road will fit the mountain. A 4th parameter can be used to change the spacing of the elevation points used for setting definitive road elevation (if not indicated 25m will be used). The bigger, the smoother. If 4th parameter is 0, the smoothing method from previous versions is used.

The output of this step is f:\project\s3_road\salida**nodes.xml**

dar_altura creates a elevation profile for the track, but with the last version of the scripts you can change it to fit real elevation changes just clicking with the mouse. Watch tutorial inside documentation folder.

f2) If elevation data available is not so good, you can create a road with a smooth elevation profile and then change elevation data to make the mountains fit that road.

First we create the smooth road:
> coge_datos
> creartrack1
> dar_altura(53,1,-1,100)

Then we change the elevation data for the terrain

> corregir

And we create a new road that fits the new elevation data.
> dar_altura(21,0.25,-0.25)

Of course you can use the parameters you want, but the idea should be creating an elevation profile you like with the first dar_altura and changing the terrain to fit that profile.

If we don't like the result, running coge_datos again will get original elevation values. After that step, run again creartrack1 and dar_altura

g) We accept the new elevation profile for the road:
> cd f:\project\Venue
> **btb06**

Step 4. Creating the terrain mesh

h)
> cd s1_mesh
> mallado_regular(12,3)

Parameters for mallado_regular are the width of driveable terrain on both sides of the road and the number of "panels" on each side.

mallado_regular creates a file called f:\project\s1_mesh\salida**anchors_carretera.geo**. This file is the base for creating the terrain with gmsh. If you want you can add a grid showing the limits of available elevation data:

> **addgrid(1,1)**

Where parameters are the number of horizontal and vertical divisions of the. The meshes you create should not exceed shown rectangular limits.

i) Mesh generation:

Open anchors_carretera.geo with gmsh and create the surfaces for the mesh.
It is mandatory to create a surface on the start and end of the road. Otherwise the script that creates the invisible protection walls will crash.

Create a Plane Surface for the non-driveable zone (with the driveable zone as its internal boundary).

Close gmsh.

Open anchors_carretera.geo with a text editor and define the Physical Surface 222 with the non-driveable surface. For the Physical surface 111 use the phys111.txt contents and complete the list with the surfaces created on each end of the road.

Save the file.

Open again anchors_carretera.geo with gmsh. Create the mesh (Mesh->2D) and save it (File->Save Mesh).
Open anchors_carretera**.msh** with gmsh and check that the mesh is complete (including the ends of the road).

Extract nodes data (nodos.txt) and polygons data (elements.txt) from anchors_carretera.msh:

> trocea_malla

Two files, **nodos.txt** and **elements.txt**, have been created.

Step 5. Creating the terrain

j) Raise the terrain

> cd f:\project\s4_terrain

> **coge_datos**
> **procesar_nodostxt**

Now we have 2 options: accept the mesh as it is (k1) or use MeshLab to reduce the poly count (k2). Usually it further reducing the poly count is unnecessary, so option k1 has to be chosen

k1) If your mesh is already optimized (read gmsh_threshod.pdf), now you just need to accept it:

> **accept_mesh**

k2) **(deprecated)** The next step is simplifying the mesh to reduce the amount of polygons (triangles).

We split the mesh in three parts (conducibles.ply, noconducibles.ply and intocables.ply):

> **simplificar**

We use MeshLab to simplify the ply files or to remove unreferences vertex in them:

- noconducibles.ply: simplify
- conducibles.ply: remove unreferences vertex
- intocables.ply: remove unreferences vertex

Resulting meshes should be saved inside "salida" folder with names i.ply, n.ply y c.ply

We put the meshes together:

> **juntar_mallas**

Step 6. Protection wall

This step will fail unless you created a mesh where there is always a driveable triangle in the boundary between the driveable and the non-driveable zone. If it fails you can just delete the ill-formed walls using BTB tools.

n) Invisible protection wall creation:

Run:

> **cd f:\project\s7_walls**

> coge_datos
> poner_muro

Two files will be created inside the "salida" folder: **muros.txt** and **muros_invertidos.txt**. One of them should work ok as a set of walls that prevent the car from falling outside of the limits of the driveable zone. poner_muro accepts an optional parameter: the LOD out value for the walls. If it is not specified, LOD out of 5m will be used.

Step 7: splitting terrain and road

q) Road and terrain splitting

It is recommended to split the road in to several segments. After doing that, the terrain must be regenerated to adapt the TerrainFaces to the new situation.

> **cd f:\project\s10_split**
> **coge_datos**
> split_track(8)

where the parameter for split_track is the number of segments the road will be splitted into.

A new figure will be created showing the nodes selected for splitting the road. The numbers of those nodes will be saved in the file “pos_nodes.txt”. Looking at the graph we can decide if we want to accept those splitting points or if we want to edit by hand ‘pos_nodes.txt’ to change those points. When we are finished, we go on with the process

> **partir_track**
> **procesar_elementstxt_mt(10,10,0)**

partir_track will read “pos_nodes.txt” and those splitting points will be used.`procesar_elementstxt_mt parameters are the dimensions of the grid uses to split the terrain and a 3rd parameter that forces the scripts to blend terrain with background images (if set to 1).

Step 8: Join the parts

o) Join the parts

Run:
> cd f:\project\s9_join
> join_all

Now copy f:\project\s9_join\salida\Venue.xml and try to open it with BTB.

p) Copy the just created Venue.xml inside the BTB project folder, copy WP.zip inside the XPacks subfolder, and open the project with BTB.