Landscape LOD generator tes4ll

Reference Manual

This Manual refers to v4.10 of tes4ll.

The scope of this manual is how to reprogram tes4ll, if you just want to use tes4ll, you should take a look at the README.

I. Introduction

a. Syntax of the command

```
tes4ll [-x xpos -y ypos] -b batchfile -f "flags,flags,flags,..."
        -w worldspace [heightmap.bmp]
```

with:

x/ypos:	x/y position of the lower left cell,
batchfile:	the name of the batch script,
flags:	the flags which are propagated to the batch script,
heightmap.bmp:	the heighmap in 32 bit.

b. Units

Oblivion uses a cell grid. Each of the cells is divided into 4096x4096 units. Moreover, 32x32 cells define a "quad".

Details can be found here:
http://cs.elderscrolls.com/constwiki/index.php/Oblivion_Units

c. The LOD technique

The Oblivion game engine seems to place all quad meshes with all vertices, even if parts of the LOD are be replaced by the near cells. The LOD vertices, which are located in the near landscape area (by default 3x3 cells) are lowered and therefore burried.

Therefore, it is important that at the cell boundary the triangles are cut. In this case, when lowering the vertex inside the near grid cell, the edge of the cell triangles shares a boundary with a triangle of the LOD. Otherwise one see a void between the near and the far meshes.

d. Warnings

"Triangle with two equal points": Sometimes very small triangles are removed, because they share one vertex point. This appears if two vertices are so close, that they cannot be distinguished.

"Mesh is too dense": this happens when the density is so large at a single point that this local region would be overcrowded. Instead of trying to add more and more points at this place (which would be impossible because a minimum distance is required), the quad will be aborted.

II. General syntax of the batch and start sequence

a. Batch files

The batch files are just text files, therefore any text editor can be used for creating and changing these files.

Each batch file is structured in sections, as usual marked in squared brackets

`[sectionname]`

The part which is read by tes4ll must be marked with **`[tes4ll]`**. The other parts are for MPGUI, the GUI frontend for tes4ll.

Comments might be added to the batch file by "**`;`**" (like in ini-files) or with "**`#`**" (like in unix).

Each command might be followed by options. Options are starting with "-" and optionally assign a value with "=". The value itself *can* be included in quotation marks, this is a *must* if the value has spaces (like **`-name="a b c"`**).

The batch file syntax supports flags (i.e. variables). Via these flags the runtime behavior of the batch execution can be controlled, either with the GUI, or via a Windows-bat-file. The flags can be defined in the tes4ll command line with the -f argument.

Each command might be preceded by a test on a flag condition, which is invoked by an "**`@`**" followed by the flag itself. A simple test on the validity of a flag is:

```
@flagname MyCommand...
```

A test on a disabled flag is (negation) is indicated by an additional "!", like:

```
@!flagname MyCommand...
```

The test on a specific value of a flag can be realized as follows:

```
@flagname=value MyCommand...
```

Flags can be combined, like:

```
@flag1 @flag2 .....
```

In addition, all mods which are used (either with the -l argument or via **`ParseModList`**) are converted to flag names. Spaces in their names (which is not allowed for flags) are replaced with '_'.

E.g., this command is only executed if Elsweyr Aequina is installed and used in the mod list:

```
@ElsweyrAnequina.esp Panorama -x=7410 -y=-157980
```

Flag can also be used in commands: a flag with the name "**`flag1`**" can be used with a "**`$`**" (like in a unix shell script): "**`$flag1`**" will be exchanged by the value of the flag. Therefore, to use a real "**`$`**", it must be preceded by a backslash: "**`\$`**".

III. Command list

a. Options

General syntax:

SetOption [option list]

SetOption -mindistance=...

Sets the absolute minimal distance for any new vertex to the existing ones. This is only ignored by the **SetVertex** and **ReadDataFile** commands.

SetOption -createpedestals

With this option all meshes are generated with outer, vertically oriented triangles which form a pedestal below the landscape. If one looks carefully to the meshes from the CS one can see that they have not only an upper surface, but also triangles around the outer frame. The reason for this is that otherwise the mixing of meshes lead to gaps in the landscape. These gaps are filled with a vertical wall (which later can appear in-game as ugly "steps" between the original meshes).

If one generates always the complete meshes with tes4ll after any installation of landscape changers (i.e. when overwriting **\meshes\landscape\lod**), this option should be kept disabled. It saves triangles. But if LOD meshes should be overwritten or only partly regenerated with tes4ll, this option must be enabled.

SetOption -worldspace=...

Replaces the "60" in the NIF file name (the number of Tamriel) by a different number.

SetOption -installdirectory=...

Defines the installation path for the files. The keyword **$_gamedir** can be used to access the original game path.

SetOption -minheight=...

Set the height to this minimum. Must be defined before **GenerateHeightMap** is called.

SetOption -useshapes

Uses tri-shapes instead of tri-strips. With this option tes4ll generates the nifs much faster and can handle more triangles. However, the files are larger and the FPS could be slightly lower. Feel free to use this option and test the impact on the performance. If you go for more than 13000 vertices, this option is obligatory.

SetOption -use16bit

Use 16 bit numbers (quasi-shorts with a dynamic range of -122880 to 122880) instead of 32-bit

floats for the derivative matrices. Requires less memory consumption of the tes4ll executable. Use this only if you run out of memory. This is e.g. the case for the “Tamriel Heightmaps”.

SetOption -ddstool="..."

Selects the command for the “bmp to dds” conversion. Used by tes4ll for the normal map generation and the build-in TES4qLOD.

SetOption -lodshadows

Generates the normal maps with "faked terrain shadows" [normal maps only].

SetOption -zboost=...

Virtually elongates the z-direction before calculation the normal vector (default 1). Enhances the contrast.

For the "lodshadows" this parameter is used to steer the "north flip", i.e. making the valleys more dark.

SetOption -bmpscaling=...

Scales the bmp's before writing the files. The value must be a fraction of 1 (0.5, 0.25, 0.125, etc).

---

b. Miscellaneous

SetPath -value=...

Changes the executable patch (like chdir).

ParseModList

Reads the mod list from Plugins.txt or from the -l argument. The mod-flags are added to the list of flags. Spaces in flag names are converted to '_'.

CallTesannwyn

Calls **TESAnnwyn.exe** and reads the **-x** and **-y** position. TESAnnwyn must be installed.

CallTes4qlod -options="..." [-silent]

Calls the build-in tes4qlod.exe (**no external TES4qLOD is required**). The option are the same as described in the original TES4qLOD README, but *without* the worldspace name and the mod list. These two arguments are added by tes4ll.

For details about the handling of TES4qLOD, see the description there.

-options	TES4qLOD options as described in the TES4qLOD manual, but without the wordspace and the mod list.
-silent	Suppresses any text output.

E.g. to generate textures with alpha blending one has to use:

CallTes4qlod -options="-q 1 -a"

ReadHeightmap [-n=...]

Reads the bitmap with the heightmap information. The position is either taken from the TESAnnwyn command (see above) or from the -x and -y argument of the tes4ll command line.

-n	Scaling parameter (1 [default], 2 or 4). Used for normal maps only.

SetHeight -z=... [-gameunits]

Sets the height in a specific area (use a foci command before, see below). If **-gameunits** is set, the height is in game units, otherwise in heightmap units.

Filter -n=...

Calculates the derivatives of the heightmap by using a filter. This avoids that the search for the slopes is sensitive to local fluctuations.

-n	Filter radius (4 is recommended).

Exit

Ends the batch processing and quits tes4ll.

---

c. Vertex point placement

SetVertex -x=... -y=...

Sets exactly one single vertex at the given position. Hence it can be used to repair floating objects, bad parts landscape, and so on.

-x	X position in game units
-y	Y position in game units

ReadDataFile -filename=...dat

Works similar to **SetVertex**, but the **x,y** are stored in a data file, which contains the coordinates separated by spaces:
x1 y1
x2 y2
....
Comments (with "**;**" or "**#**") or section names (**[**...**]**) can be added in the data file.

-filename	The name of the data file.

SetGrid -x=... -y=...

This sets vertices with a fixed grid pattern.

Example:
SetGrid -x=4096 -y=4096
sets the grid points at all edges of the cells.

-x, -y	Distance of the grid (in game units).

BreakAtGrid -x=... -y=... -max=... -zmin=...

Works as "wall remover" in order to avoid the ugly steps between the near and the far LOD meshes.

Places a grid of virtual lines. Along each of these line the algorithm checks if the difference in height of the linear interpolation between the two grid points and the real heightmap is larger as the "**max**"-threshold. In this case, the grid line is split into two segments at the point with the largest difference. The "**mindistance**"-value is taken into account.

For the two new segments this procedure is repeated. This is done iteratively until no more space is left to place (within the mindistance-condition) new split points.

-x, -y	Distance of the virtual grid
-max	Maximal accepted height difference
-zmin	No action below **zmin**

Panorama -x=... -y=...

Puts a virtual observer at the given position (**x,y**) and calculates the highest visible point of the landscape (i.e. the horizon) in a 360 degree view mode around this observer.

Can be used to increase the quality of the horizon (the shape of the mountains) from a single points of view (e.g. a player home).

-x, -y	Position of the view point.

ContourLine -x=... -y=... -z=... [-offsetx=... -offsety=... -findmin -findmax -onlyintracell]

This command has 2 operation modi:

1. Called *before* the triangulation it searches for contours (also called "breakline"). Starting at the offset position, a virtual grid (with **x** and **y** as distances) is applied.

   In the following, for each outer line of the grid box, the algorithm checks if the heightmap crosses the defined contour height **z**. If this is the case a vertex is set with a position close to the breakline.

2. *After* the triangulation it searches for triangles which cross the height **z** and try to split them [VERY EXPERIMENTAL, leads to small gaps between the triangles]

-x, -y	Distance of the virtual grid (in game units).
-z	Height for the contour line.
-offsetx, -offsety	Start point for the virtual grid.
-findmin, -findmax	Options which look for the lowest/highest point between two breakline points. This helps to avoids direct connections between breakline points after the Delaunay triangulation.
-onlyintracell	Restricts the finding of the min/max points to be in the same single cell as with the breakline point.

CreatePolygon -name="..." -x1=... -y1=... -x2=... -y2=...

Creates a polygon with the name "**-name**". Polygons (which must be defined before the Delaunay triangulation) can be used to cut their shapes out of triangle net after the triangulation.

-x1, -y2, -x2, -y2	Position of the first initial two edge points of a polygon.
-name	Name of the polygon. Must be unique.

AddVertexToPolygon -name="..." -x=... -y=...

Adds the next vertex to a polygon.

-x, -y	Position of the next edge points of a polygon.
-name	Name of the polygon. Must be already existing.

ReadPolygonDataFile -filename=... [-name=...]

Reads the vertex points of a single polygon or a list of polygons from a data file which has the same form for a single polygon like for **ReadDataFile**:

```
x1 y1
x2 y2
....
```

or, if a list of polygons should be read:

```
[polygonname1]
x1 y1
x2 y2
....
[polygonname2]
x1 y1
x2 y2
....
```

-filename	File name of the data file.
-name	Name of the polygon *if the data file has no [...] sections and contains only one polygon*. Must be unique.

---

d. Focus

These commands define the valid area used for the following commands. This means that the commands or algorithms which comes after the focus are restricted to the region defined by the focus.

FocusAll

Use the entire worldspace.

FocusQuad -x=... -y=...

Use only one single quad.

-x, -y	Quad number. The numbering is consecutive, i.e. "**-x=0 -y=0**" is the center quad, whereas "**-x=1 -y=0**" is the quad "**32,00**".

FocusRec -x1=... -y1=... -x2=... -y2=...

Use only rectangular area.

-x1, -y1	Lower left point (in game units)
-x2, -y2	Upper right point (in game units)

NB: **x1** must be smaller as **x2**, and **y1** must be smaller as **y2**.

---

e. Density algorithms

These commands do not directly set the vertex points, but define only the density function. The vertex points are then set later using a combination of all your algorithms.

The foci can be used to define an algorithm to be valid for a specific region only.

More generally speaking, the algorithms form a list:

```
Alg1 ....
Alg2 ....
....
```

The common syntax is:

AlgXXX -multiply=... -add=...

The **multiply** and **add** values are used to define, how the different algorithms are merged.

The behavior is as follows: first, the value v_i of a specific algorithm is added to the running value v_r, after this operation the multiplication is done:

v_(r+1) = (v_r + c_add * v_i) * c_multiply * v_r

where c_add and c_multiply are the constants defined by **-multiply** and **-add**, respectively.

AlgFirstOrder

The value v_i is proportional to the first derivative of the height (the slope), see also "**Filter**".

AlgSecondOrder

The value v_i is proportional to the second derivative of the height (the curvature), see also "**Filter**".

AlgLayer -lowest=... -highest=... -minval=... -maxval=...

Enhances a layer (a band of the height) for **z**-values between **lowest** and **highest**. Inside this layer the value v_i is **maxval**, outside it is **minval**. If maxval is *lower* as minval, the density is

reduced.

-lowest, -highest	Height range of the layer.
-minval	Value outside of the layer.
-maxval	Value inside of the layer.

AlgLinear -lowest=... -highest=... -minval=... -maxval=...

Similar to **AlgLayer**, but uses a linear interpolation between **-lowest** and **-highest**. **-minval** is the value at **-lowest**, and **-maxval** the value at **-highest**.

If **-maxval** is larger as **-minval**, the density at higher points is larger, otherwise it is smaller.

AlgRadial -x=... -y=... -near=... -far=... -minval=... -maxval=...

Enhances a region around a point (**x,y**) with the value **maxval** for distances lower than **near**, and **minval** for distances larger than **far**. Between near and far a linear interpolation is done.

-x, -y	Center of the circle.
-near, -far	Inner and outer radial of the linear interpolation
-maxval	Value inside the inner circle.
-minval	Value outside the outer circle.

AlgPeakFinder -radius=... -scanradius=... -lowest=...
-maxval=... -minval=... [-linear]

This command can be used to improve the quality of mountain peaks. In each circle with the radius **scanradius** the highest point is scanned. The density will be improved in a circle with the radius **radius** around each found peak. The value inside this circle is **maxval** whereas outside it is limited to **minval**. E.g., "**maxval=1 -minval=0.2**" means that the mountain regions are improved by a factor of 5 (the ratio of this 2 values).

-lowest	Threshold. No peaks are searched below this height. This restricts the algorithm to the highest mountains, and not to small hills.
-linear	Option to use a linear interpolation between **maxval** and **minval** instead of fixed values.

---

f. Vertex calculation

These commands sets the random points based on the density defined by the above-mentioned algorithms. Do not go much above 10000 vertices per quad, the Oblivion game engine can only

handle one NiTriStrip/-Shape.

SetMaxVertices -n=...

Fill the maximum number of vertex points until -n vertices are reached in total [Only recommended for creating a mesh for the entire world].

SetMaxVerticesPerQuad -n=...

Fills the maximum number of vertex points until **n** vertices are reached in each quad. Quads which are only covered partly by the heightmap are getting a reduced number of vertices.

g. Triangle modification commands

MakeTriangulation

Calls the Delaunay algorithm and produces the triangles.

DivideGrid -x=... -y=...

Cuts the grid. Example:
DivideGrid -x=4096 -y=4096
cuts all triangles which share more than one cell directly at the cell boundary.

DivideAtPolygonBorder [-name="..."]

Divides all triangles at the border of the polygon shape "-name". If name is not used, all polygons are used for this operation.

-name	Name of the polygon. Must be already existing.

DivideAt -x=... *or* -y=...

Divides all triangles at a vertical (**-x**) or horizontal (**-y**) infinite line.

-x, -y	Position of the line.

DivideBetween -x1=... -y1=... -x2=... -y2=...

Divides along a line which starts at (x1,y1) and ends at (x2,y2). These 2 points must be existing and being defined as vertex points before the triangulation.

-x1, -y1, **-x2, -y2,**	Position of the start and end point.

StencilPolygon -name="..."

Removes a polygon shape from the mesh.

-name	Name of the polygon. Must be already existing.

InactivateAllVertices

First step for cleaning invisible triangles.

ActivateVisibleVertices -x=... -y=... -z=... [-radius=...]

Acivates all vertices, which are visible from a view point (**x,y,z**). If -radius is used in addition, also the displaced points (**x+radius,y,z**), (**x-radius,y,z**), (**x,y+radius,z**), etc... are used. If one of the displaced points is below the surface, its z-value is corrected.

-x, -y, -z	View point
-radius	Forms a virtual box around (**x,y,z**) with the edge length **radius**/2

RemoveInactiveTriangles

Last step for cleaning invisible triangles. Removes all triangles, which have not at least one active vertex.

h. Mesh generation commands

WriteQuad -x=... -y=... [-tex=...dds] [-ps]

Writes out a single quad with the number (x,y).

-tex	Adds a texture in addition. Do not use this mesh for LODs.
-ps	Writes out in addition a postscript-file with a map of the triangles.

WriteAllQuads

Writes all quads of the entire worldspace. Use this command to create the real in-game meshes.

```
WriteAll  [-tex="...dds" -name="...nif"
           -transX=... -transY=... -transZ=...]
```

Writes out a single world mesh [limited to one single NiTrStrip/-Shape object, i.e. 65536 triangles]. Can be used to produce “LOD” meshes for Immersive Interiors.

-tex	Adds a texture in addition.
-name	Name of the mesh (otherwise “wordspace”.nif is used.
-transX/Y/Z	Optional translation of the origin of the mesh

V. Remarks

Suggestions? Features missing?

Please contact me (gruftikus@tesnexus, @bethsoft, @tesalliance, @scharesoft)