MMTiler Utility – Quick Start-

(Questions Not Covered? Email support@freeterrainviewer.com)

- **Preparing Data For Use In The Runtime Engine:**

The MMTerrain engine streams data as needed to render a virtually unlimited database; as a result, there are some ideal tile sizes and settings that work better than others.

For Windows users, a basic tiling tool is included that generates ideal output from different input sources. The tool is designed to work with geotiff data (using lat/lon and UTM), but also has an option for generic sets of indexed tiled images, with a consistent naming convention.

The output of the tiling tool is a set of folders with tiled imagery and elevations, neatly re-sampled and organized into levels of details. A text file is generated that can be used in the default Unity setup to run the terrain. Further modifications to this basic setup are possible depending on the application.

NOTE: The tiler tool is provided as an added convenience, and is not required for the runtime engine. The format for tiled databases, and associated metadata, is described in the document <name>. The core engine will run with the appropriate data regardless of how/where it is prepared.

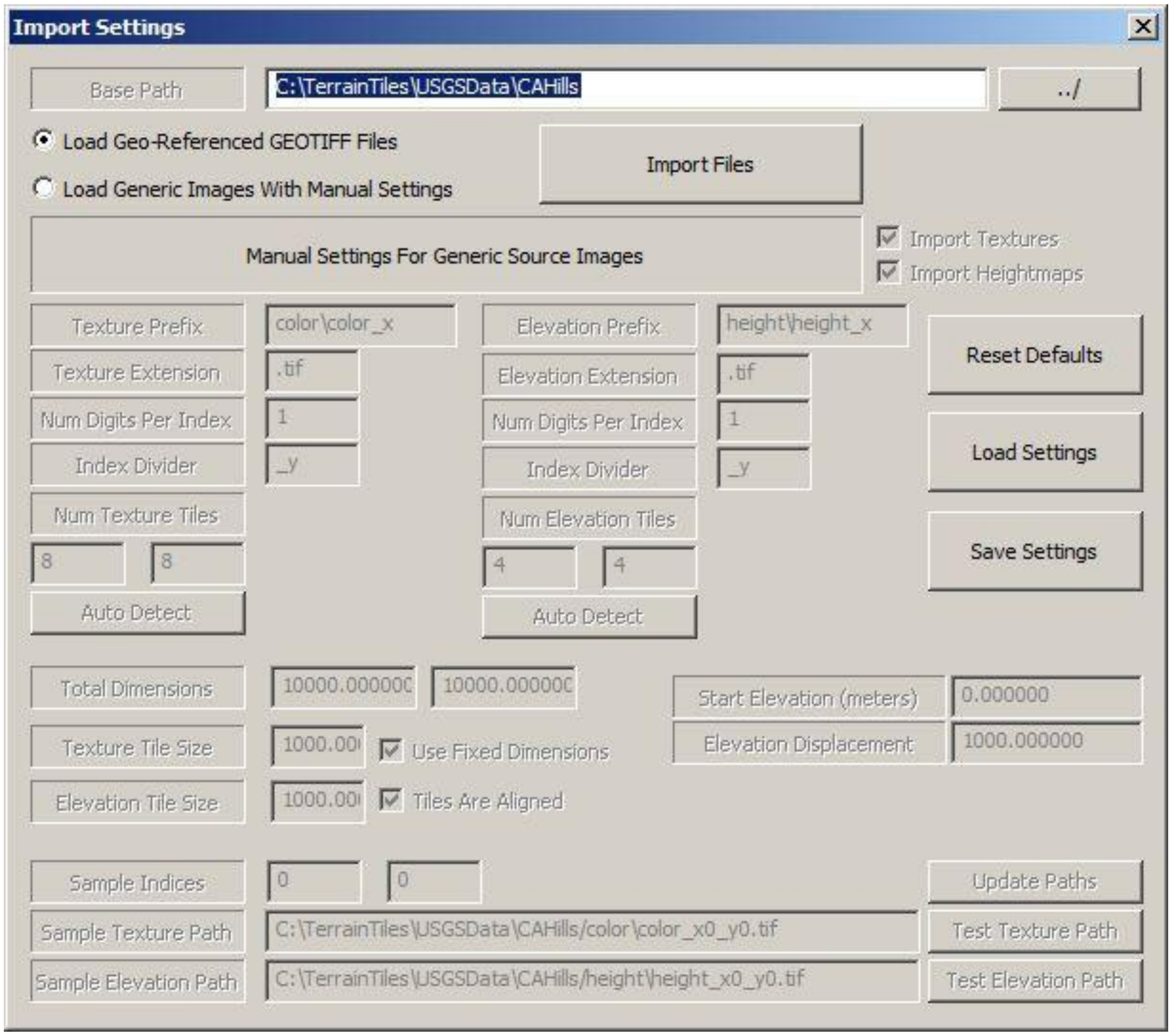
In A Nutshell – Input Data:

Geotiffs already come with coordinate data that defines the region and alignment. For geotiffs, there is very little user input required, basically just the input and output paths.

For generic indexed images, the naming convention and paths to the tiles must be defined. The number of tiles can be auto-detected (assuming a regular grid with no missing spots). The world size can be defined by absolute size, or tile size. Elevation tiles can be a different resolution and world size than textures. Elevation/displacement must also be defined.

- **Import Settings:**

The tiler tool starts with the Import Setting dialog box:


Import Settings

Base Path: C:\TerrainTiles\USGSData\CAHills

[✓ Load Geo-Referenced GEOTIFF Files] Load Generic Images With Manual Settings

Manual Settings For Generic Source Images

Import Textures ✓ | Import Heightmaps ✓

Texture Prefix	color\color_x	Elevation Prefix	height\height_x
Texture Extension	.tif	Elevation Extension	.tif
Num Digits Per Index	1	Num Digits Per Index	1
Index Divider	_y	Index Divider	_y
Num Texture Tiles	8, 8	Num Elevation Tiles	4, 4

Total Dimensions	10000.00000C, 10000.00000C
Texture Tile Size	1000.00
Elevation Tile Size	1000.00
Use Fixed Dimensions	✓
Tiles Are Aligned	✓
Start Elevation (meters)	0.000000
Elevation Displacement	1000.000000
Sample Indices	0, 0
Sample Texture Path	C:\TerrainTiles\USGSData\CAHills/color\color_x0_y0.tif
Sample Elevation Path	C:\TerrainTiles\USGSData\CAHills/height\height_x0_y0.tif

NOTE: Settings for all fields are automatically restored from the previous session (last folder browsed, etc.). Settings can also be manually reset (“Reset Defaults”), or saved/restored (“Load/Save Settings”);

At the top is the Base Path for the source data. This could be a folder of geo-referenced geo-tiff files, or the root path to some set of generic indexed images.

When the geotiff option is selected, most of the options are disabled, as no user settings are required.

Using Manual Settings:

Textures and/or heightmaps can be loaded from most standard formats. The naming convention and pathing options must conform to the available options (this is generally not a problem). The generated paths can be tested at any particular XY indices. See the screenshot for an example of how the paths might be constructed.

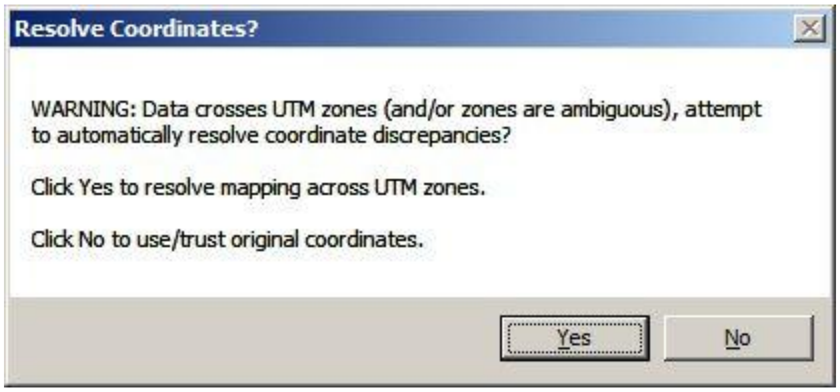
The number of tiles can be auto-detected for both textures and heightfields. The world dimensions can be defined in absolute extents (“Fixed Dimension”), or explicit tile size. If both the heightfields and textures cover the same total area (ie. no blank areas or overlap along any edge), then “Tiles Are Aligned”, and the tile sizes will automatically be constrained to match the total dimensions / the number of tiles.

The elevation displacement defines the minimum elevation in world units, and the total displacement. Geotiff data typically uses actual data relative to sea level, but artificial data can use virtually any custom elevation range. Values will be relative to the tile size and total dimensions.

Once the paths and settings are in place, clicking “Import Files” will start the process.

- **Importing Files:**

Ideally, there is nothing wrong with the data, and the conversion is straightforward. However, many datasets cross UTM zones and/or may be missing coordinate data. The most common case is that data from multiple UTM zones must be projected to a single dominant zone (in order to fit within a consistent coordinate spread across a rectangular grid). When converting geotiff data, you may see a dialog box like this:


Resolve Coordinates?

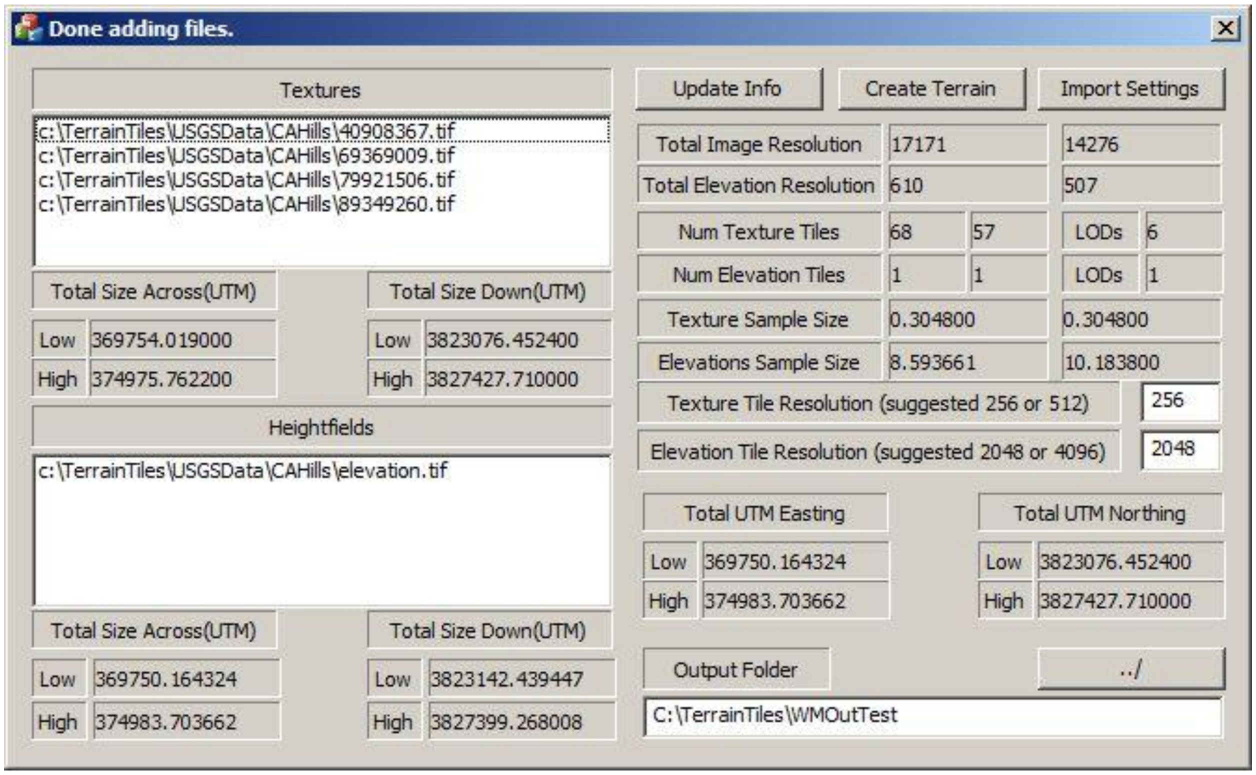
WARNING: Data crosses UTM zones (and/or zones are ambiguous), attempt to automatically resolve coordinate discrepancies?

Click Yes to resolve mapping across UTM zones.

Click No to use/trust original coordinates.

In some cases, the coordinates may be fine, and there is a simple ambiguity or discrepancy. However, in cases where the conversion doesn’t go as expected, using the auto-correct option may be the best option for some datasets that otherwise don’t map to a rectangular grid. NOTE: There are no user settings available for auto-correction; everything is simply re-projected to the dominant UTM zone.

- **Export Settings:**


Done adding files.

Update Info | Create Terrain | Import Settings

Textures
c:\TerrainTiles\USGSData\CAHills\40908367.tif
c:\TerrainTiles\USGSData\CAHills\69369009.tif
c:\TerrainTiles\USGSData\CAHills\79921506.tif
c:\TerrainTiles\USGSData\CAHills\89349260.tif

	Total Size Across(UTM)		Total Size Down(UTM)
Low	369754.019000	Low	3823076.452400
High	374975.762200	High	3827427.710000

Heightfields
c:\TerrainTiles\USGSData\CAHills\elevation.tif

	Total Size Across(UTM)		Total Size Down(UTM)
Low	369750.164324	Low	3823142.439447
High	374983.703662	High	3827399.268008

Total Image Resolution	17171		14276	
Total Elevation Resolution	610		507	
Num Texture Tiles	68	57	LODs	6
Num Elevation Tiles	1	1	LODs	1
Texture Sample Size	0.304800		0.304800	
Elevations Sample Size	8.593661		10.183800	

Texture Tile Resolution (suggested 256 or 512): 256
Elevation Tile Resolution (suggested 2048 or 4096): 2048

	Total UTM Easting		Total UTM Northing
Low	369750.164324	Low	3823076.452400
High	374983.703662	High	3827427.710000

Output Folder: ../
C:\TerrainTiles\WMOutTest

The final dialog box requires very little user input, and simply displays the estimated output. This is the time for a sanity check. The number of tiles across and down indicates the number of images that will be created at the specified output resolution.

On the right side, the total resolution of the image and elevation data (from all combined source files) is shown. This can be compared with the known input data as a verification step (especially with geotiff data). The number of tiles, and expected levels of detail, are shown. The exact disk cost can't be calculated due to variability in compression, but the total output can be estimated from the numbers.

The extents of the data are displayed as UTM easting/northing. For artificial data, these values will reflect the absolute world size manually entered in the Import Settings.

- **Converting The Terrain Data:**

Clicking "Create Terrain" starts the process. This can take anywhere from a few minutes, to a few hours, to several days, depending on the size of the data, and where it is stored (we typically run large conversions on external USB drives, which slows down the process a bit).

Provided that all data is available and the settings are correct, the rest of the process is automatic. The output folder will contain multiple sub-folders with the tiled elevation and imagery (with auto-generated levels of detail), and a text file that can be imported into the Unity engine as a text asset.

The textures and elevations are both stored as 24-bit RGB jpg images. These can be fed directly into the engine as-is, using the appropriate metadata settings. The elevation heightmaps are encoded for RGB (vs. plain grayscale), this splits up the divisions between channels so that R covers the bottom 1/3rd of the elevation range, G covers the middle, and B covers the top. This triples the precision of standard grayscale maps at virtually no extra performance cost.

- **Text File With Metadata:**

Once the files are generated, the text asset is attached to the basic *.cs script, and the script sets up the engine from there. NOTE: While the output file contains many settings, only a few of them are actually used in the runtime script. In the case of things like lat/lon and UTM information, the data is available for those that need it, but for the most part it can be ignored.

For convenience, the format of the text file is an actual block of standard c# variable declarations, exactly as they might appear in a normal *.cs file. This means that the variables can be directly pasted into scripts (as needed). It is assumed that a final implementation for any application will have some custom method of setting metadata.

- **Metadata:**

The following settings are from the auto-generated text files, and are the critical variables needed to control the core engine (see the *.cs file for details). Some settings are purely optional (such as values for lat/lon, UTM, meters, etc.).

Although the tiler tool is specifically designed to work with the Unity engine setup, it is not a required part of the engine, and the image tiles (and metadata) can be defined in any number of ways. The following settings are the output from the tiler tool, not necessarily the absolute inputs required for the runtime engine. See tech doc for more info.

NOTE: Unless the settings are being manually defined, the tiler will generate ideal output values for all settings. In the case of manually created data, the text file can be used as a template, and/or the actual script can be modified so that settings are defined directly in code.

All Exported Settings:

The following settings are used by the main *.cs script to generate paths to each image file. The script provides the Texture2D objects to the core engine on demand. The core engine itself has no knowledge of paths, it simply requests tiles by index and detail.

public string DEFAULT_LOCAL_ROOT_PATH = "C:/TerrainTiles/CAHillsNormal";

public string DEFAULT_REMOTE_ROOT_PATH = "http://www.myserver.com/CAHillsNormal";

public int maxTextureRowsPerFolder = 295;

public int maxElevationRowsPerFolder = 20001;

The following values are used to setup the runtime engine parameters for textures…

```
public int TEXTURES_textureTileResolution = 256;

public int TEXTURES_numTextureLODS = 6;

public int TEXTURES_texturesNumXTiles = 68;

public int TEXTURES_texturesNumYTiles = 57;

public float TEXTURES_textureTileSize = 76.963814f;
```

This is an internal value used to consolidate multiple tiles into a larger texture composed of varying LODs (these are assembled at runtime, during rendering). The best value is generally going to be 4x the texture resolution (no greater than 4096). This value is determined automatically by the tiler, but can be modified.

```
public int TEXTURES_maxConsolidatedTextureResolution = 1024;
```

The following values define similar parameters for the elevation tiles.

```
public int ELEVATIONS_elevationTileResolution = 2048;

public int ELEVATIONS_numElevationLODS = 1;

public int ELEVATIONS_elevationsNumXTiles = 1;

public int ELEVATIONS_elevationsNumYTiles = 1;

public float ELEVATIONS_elevationTileSize = 5233.539551f;

public double ELEVATIONS_displacement = 681.704468;

public double ELEVATIONS_smoothingRange = 1.0f;

public double ELEVATIONS_blendTransitionRange = 2616.769775f;
```

The following values define optional measurements that may be used in the code (or not).

```
public double ELEVATIONS_minLatitude = 34.541759;

public double ELEVATIONS_maxLatitude = 34.580463;

public double ELEVATIONS_minLongitude = -118.419444;

public double ELEVATIONS_maxLongitude = -118.363055;

public double ELEVATIONS_minElevation = 714.794678;

public double ELEVATIONS_maxElevation = 1396.499146;
```

public double ELEVATIONS_elevationRange = 681.704468;

public double numSourceSamplesPerElevationTileX = 610.000000;

public double numSourceSamplesPerElevationTileY = 722.872183;

public double numSourceSamplesPerTextureTileX = 252.514706;

public double numSourceSamplesPerTextureTileY = 252.514706;

public double originalTextureSampleSizeInMetersX = 0.304800;

public double originalTextureSampleSizeInMetersY = 0.304800;

public double originalElevationSampleSizeInMetersX = 8.593661;

public double originalElevationSampleSizeInMetersY = 10.183800;

public double metersPerTextureTile = 76.963814;

public double metersPerElevationTile = 5233.539338;

The following values define the detail level of the smooth-morphing geometry mesh. The values define the smallest detail size (base tile size), the distance before the detail is halved, and the number of times this occurs. The levels of detail will double in size (effectively halving the detail at each stage) for as many levels as specified. This is usually more than enough to extend a reasonable horizon.

NOTE: Unity mesh limits are 65535 vertices, so there is a cap on the possible detail. Detail should be allocated as needed within the desired horizon and viewing limits, a tighter viewing distance means that more detail can be applied locally. The settings can be experimented with, if the number of required vertices exceeds 65535 then Unity will fail to create the mesh.

public float baseTileSize = 4;

public float highDetailRange = 100;

public int numDetailLevels = 12;

The following settings are purely optional, and not used by the runtime script. Some data has been included in the exported text file for posterity only. NOTE: Depending on whether geotiff data uses meters or feet, there may be some ambiguity. It is assumed that geotiffs coordinates will be in meters.

public double feetPerTextureTile = 252.505959;

public double feetPerElevationTile = 17170.405201;

public double feetAcross = 17170.405201;

public double feetDown = 14275.779984;

```
public double milesAcross = 3.251970;

public double milesDown = 2.703745;

public double kmAcross = 5.233539;

public double kmDown = 4.351258;

public string dominantUTMZone = "11N";

public int useRGBEncodedElevations = 1;

public int useGrayscaleElevations = 0;
```

- **Integrating Data Into The Runtime Engine In Unity:**

The basic setup for Unity includes a wrapper script that uses the text file to define all metadata.

The text file created from the tiler tool can be used as a text asset for the wrapper script "BasicTiledTerrain.cs".

The wrapper script sets all internal data and supplies the core engine with image tiles as needed.

The full setup process is referenced in the document "MMTerrainQuickStart.pdf".

NOTES ON CONVERTING DATA FROM USGS:

USGS is no longer hosting geotiff data (they are currently using other formats), however, they have provided conversion scripts that generate geotiffs that can be used with the tiler.

Although USGS usage is out of scope for the engine, as a courtesy, the exact steps for downloading and converting USGS into geotiff format (including pre-bundled applicable scripts) can be downloaded here:

www.freeterrainviewer.com/USGSHelperStuff.rar

This file contains all USGS conversion scripts, steps, and screenshots necessary to download and process USGS.

BEWARE: Data from from USGS is not guaranteed to work in the engine unless it has been properly formatted and prepared to run in the Unity MMTerrain engine. There is virtually no support for issues related to USGS or data preparation (in terms of the runtime engine), but feel free to email with questions, we are happy to assist in any way possible within reason.

An example of properly formatted data can be seen in the test database ("CAHills", reference in multiple places), it is recommended that a solid pipeline be established before attempting to run unknown or unverified source data.