Tom's Terrain Tools

Contents and Instructions

This package includes scripts and example data for Unity 3D.
For full details of the contents and how to use them, check original archived docs:
https://web.archive.org/web/20200724223550/http://lemuria.org/Unity/TTT/
**These docs are also copied into this file already.*

This package contains some terrain & tree assets and textures from Unity Technologies, used with permission per the Unity Assets License.
The full package of these assets can be downloaded from
https://assetstore.unity.com/packages/3d/vegetation/terrain-assets-6
For these assets, the license included in the full package applies.

Support

Use github page for reporting issues & feature requests
https://github.com/unitycoder/TomsTerrainToolsUnity

OFFLINE GUIDE

Tom's Terrain Tools

Tom's Terrain Tools is an editor script that makes it a lot easier to bring realistic terrains and landscapes into Unity 3D.

The Terrain Tools take distribution maps you have created elsewhere, e.g. in Terragen, L3DT, World Machine, etc. and apply them to the terrain in Unity. They do **not** do actual procedural/fractal generation like FractScape or the Terrain Toolkit do. Instead, they allow you to harvest the incredible power of the tools dedicated to this purpose and bring their results into Unity.

In detail, the Terrain Tools can:

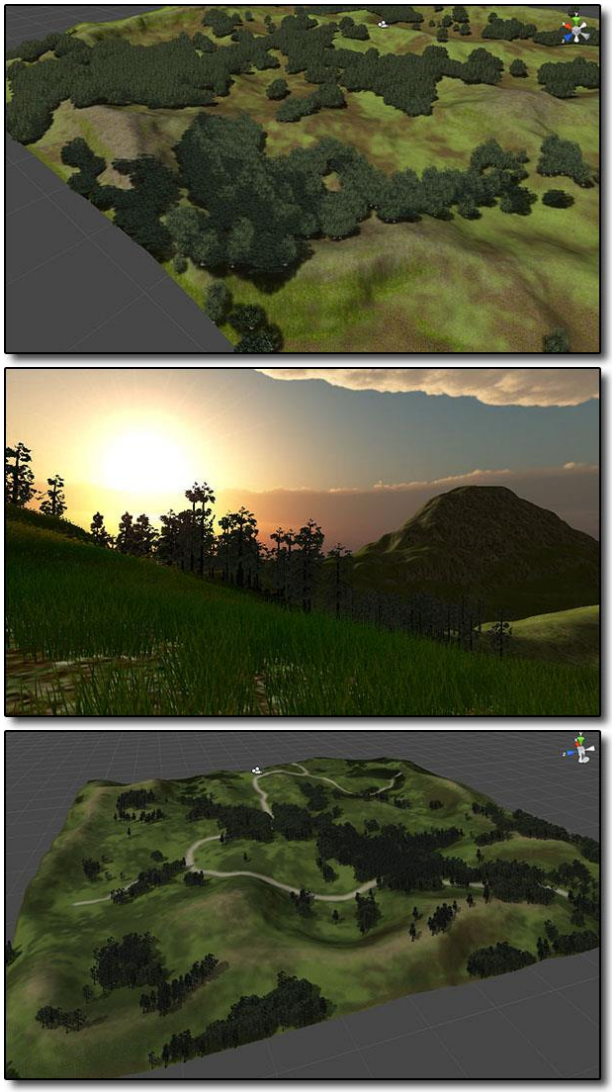
- Apply terrain textures (splatmaps)
- Mass-place trees (many thousands at once)
- Mass-place grass and detail meshes over the entire terrain
- Add overlays, which can be used for paths, roads, rivers, etc.
- Add splatmaps individually or even build up a completely new terrain cover incrementally (with the incremental splatmapping tool)

In addition, there are a few extra scripts thrown in that you may find useful, these allow you to:

- Place selected objects on the terrain surface
- Use a spawn map to spawn objects — much better than spawn points as you can not "camp" a spawn map
- Adjust terrain quality settings
- Access individual trees to modify them, for example to make trees destructable (a must-have if you have tanks, large monsters, etc. in your game)

You will receive full, documented source code for all scripts, so you can adapt them to your needs if you want to.

Using these tools, you can easily and quickly create lush and beautiful outdoor scenes like these these three example scenes, which are also included in the download, including their Terragen 2 source files:



Documentation

- Manual — full documentation
- AutoMagic — a cool new feature for even quicker terrain generation
- Incremental Splatmapping — add splatmaps to existing terrain or build up a new one in layers
- Extras — additional content
- FAQ and known problems — look here if you are stuck
- Terragen Workflow — how to create the various maps in Terragen 2
- L3DT Workflow — how to create maps in L3DT
- World Machine 2 Workflow World Machine 2 workflow
- Terrain Software — list of tools you can use to create maps

Manual

Preparations

All the following assumes that you already have done:

- Created a new terrain
- Imported a heightmap
- Imported all the splatmaps, treemaps, etc. you wish to use into Unity as RGB 24bit textures.
- If using Unity 2.6, all the textures/maps you intend to use must be set to "readable" in the import settings.
- If using Unity 3, all the textures/maps must have their ‘‘Texture Type"‘ set to "Advanced" and then checked the "Read/Write enabled" option.

Once you've installed the Terrain Tools, you will find them in **Window => Terrain Tools** in the Unity editor. You will probably want to dock that window somewhere.

Interface

To the right you can see what the interface looks like. Its individual sections can be collapsed if you don't need them. The topmost object field can be used to specify which terrain you want to work on, if you have several in your scene. If you leave this empty, the tool will automatically use the currently active terrain. So if you only have one terrain object in your scene, this field is optional.

Splatmaps

Precondition: You need to have set up textures for the terrain before applying the splatmap. 4 textures for 1 splatmap, 8 for 2 splatmaps. Additional textures can be set up, but won't be used by the script.

This is the easiest part. Simply drag your splat map(s) to the texture fields and hit the "Apply Splatmap(s)" button.

The image will be processed and terrain textures applied to the terrain according to the colours in the splatmap. Textures will automatically be blended and colour values normalized so you don't have any dark or overbright spots.

Splatmaps must be square, in a power-of-two size and imported as RGB24. They do not have to have the same resolution as the heightmap.

Trees

Precondition: You need to have set up at least 3 trees for the terrain.

The most configurable and powerful script.
First, you need to drag the tree distribution map to the "Tree map" selector. Again, power-of-two, etc.

Tree Density and **Threshold** control the amount of trees that will be placed. Density can not be above 1.0 which means every dot on your treemap will result in a tree being placed. The right value for this depends on your treemap, the size of your terrain and the scale of your trees. The default value works fine for FPS-size maps. Be careful with higher values, they can quickly create too many trees. Threshold is a kind of "sharpness" value. It determines how much colour your map requires at least for tree placement to happen. At high values, only the brightest spots in the treemap will have trees. At 0.0 you could have individual trees everywhere. The valid range is 0.0 to 1.0

Tree Size can be used to scale all trees up or down in order to fit them to your scene. Usually, the default value of 1.0 will work just fine.

Size Variation gives a bit of randomness and thus more realism to tree sizes. The default values work fine for most cases, if you want to play around with them, I suggest changing them +/- 0.1 at a time.

Grass and Bushes

Precondition: You need to have set up at least 6 grass textures or detail meshes.

This dialog takes two distribution maps. One for grass, the other for bushes. At least that is the idea, you can use them, mix them, in any way you like. Nothing says you can't have 2 grass textures and 4 bush types. The number is simply because of the RGB structure of your map images. The principle is the same as for tree maps — each colour stands for one type of grass texture or detail mesh, black means empty areas.

Grass Density and **Bush/Detail Density** are simple multipliers for the density of grass and bushes. Usually, you want grass to be dense while bushes are a bit more scattered about, which is what the default values accomplish.

Grass Clumping is a value that makes it less likely that you get isolated, single grass billboards, which usually looks awkward. It works as a chance that around any grass placed, there will also be other grass nearby, even if the grassmap does not mention it. The default value works fine for most cases, if you want to change it, valid values are between 0.0 and 1.0

Overlay Map (optional)

Precondition: Nothing

The overlay script allows you to add additional terrain textures to your map, again according to a distribution map. The most common case that you will probably use it for is to add roads, paths, rivers, lakes and other well-defined features. There is a lot you can do with this script. Since it is not always used, it is collapsed by default. Click the triangle in front of it to expand it.

First, you need to drag your overlay map into the slot provided. You also need to select a terrain texture and drag it into the **Overlay Texture** slot. This texture will be added to your terrain textures array with the tile size values you give here.

Threshold again allows you to cut out soft edges and ignore any low noise your map might have.

When you apply this script, it will add your selected overlay texture to the map according to the overlay map. That is, it will lay down a path or road according to the road map that you've provided. Not only is this simpler than drawing the road inside Unity, it also allows you freedom in the creation of the map image. You could use bezier curves, for example, or use varying width. You could even take it from an actual map if you are re-creating a real-world environment.
The script will also automatically blend the overlay texture with the existing splatmap and colour-correct the existing splatmap textures.

Clear Trees and **Clear Grass** will automatically remove trees and grass from the overlay parts, so that you don't have trees in the middle of your river or grass on your road. If your overlay map has soft edges, this removal will consider them, softly thinning out the grass around your path. Please note that removal of trees is very slow. It is often better to use an image manipulation program to simply substract the overlay map from the tree distribution map before applying that in the Trees script, which is why this option is by default disabled.

Finally **Change Terrain** is an incredibly powerful feature that will change the heightmap of the terrain as well. This enables you to use the Overlay Map script to actually carve rivers or lakes into your terrain (make sure they have soft edges!). Or you could use it to have a slightly elevated road, or maybe train tracks? The possibilities are endless. The value you put in here is in absolute measurement units, so it depends a lot on your terrain size, especially the Y value. Negative values cut into the terrain, positive values elevate it.

AutoMagic

AutoMagic is a revolutionary new feature in Terrain Tools 2, after version 2.1

Overview

AutoMagic allows you to create a whole terrain with a few clicks. It will create the terrain, load the heightmap, set up terrain textures, tree prefabs and grass, apply splat-, tree- and grassmaps, all automatically.

Preparations

In order for AutoMagic to work, you have to create one of each heightmap, splatmap, treemap and grassmap. They **must** follow a simple naming convention: (arbitrary name)-(mapname).ext:

- (somename)-heightmap.raw
- (somename)-splatmap.(tiff|png|jpg)
- (somename)-treemap.(tiff|png|jpg)
- (somename)-grassmap.(tiff|png|jpg)

and they must all be found in the same folder in your Unity assets.

Using AutoMagic

- Open Tom's Terrain Tools
- Open the AutoMagic foldout
- Drag any of the four maps, it doesn't matter which, to the "Hint Object" drop.

- Change any of the default textures, trees or the default grass as you wish. The defaults assume you have the free Unity Terrain Assets (download here or from the Asset Store) installed.
- Click "Run AutoMagic" and wait a few seconds
- done

You can also check out this video for a demo:

Details

AutoMagic will use the settings that you make in the Terrain Tool settings, meaning you can modify the tree size and density, etc. as you are used to.
It will also add the maps you used to the proper object fields in TTT, so you can easily update or fine-tune anything, or re-apply if you made manual modifications

Examples

To get you started and as a reference point, there are a few example height-, splat-, tree- and grassmaps in the TTT package, check the **AutoMagic Examples** folder.

Incremental Splatmapping

This additional tool (available in TTT 2.4 and newer) allows you to incrementally build up a texture covering, or add terrain textures with splatmap control to an existing terrain. You can also use it if you need more than 8 different textures, and don't want to modify the main script.

Using it is extremely simple. Just like in the main tool, you need a splatmap with R,G and B channels representing three different textures. In the dialog window, drag your terrain into the terrain slot and your splatmap in the splatmap slot. Drag the textures for R,G and B into the respective fields, and you can also set the tile size here directly. Note that all three slots are optional, so if you want to add only two textures and they're in the G and B channels, leave R empty.

Note that in this tool, black means no texture and will leave the terrain texture at this point unchanged. You can, of course, use shades of colours to indicate less coverage (so a dark red would add the red texture, but at less strength). The alpha channel is not used.

Finally, there is one slider to set that controls how the new textures are blended with existing textures where they overlap. At the default setting 1.0 the textures will be blended with equal strength. Slide it to the left to give preference to the existing textures, and to the right to give preference to the new texture.
If, for example, you are adding roads, stone paths or fields to an existing terrain, you want it pretty high so the original terrain is completely covered. But if you add foot paths or patches of green to make more variety (say, from a noise texture), you want it more low so the old texture dominates and the new influences the final look only slightly.
Experiment with this slider to get a feeling how it affects the final visuals of your terrain.

Extras

There are some additional scripts in the Terrain Tools package that you might find useful.

Crater Maker

The Crater Maker script allows you to, well, make craters in your terrain. With some noise (if you want), smooth edges, change of height and texture. You can define the radius and depth.

To use it, attach the Crater Maker script to your terrain, then call the Create() function every time you want to make a crater. The script Create Crater On Impact shows you an example of how you can accomplish the effect of a bomb or missile hitting the terrain.

All units are world-units.

Two **Warnings**:

- craters made into your terrain are permanent, they will not revert to normal when you stop the editor playing. This is due to the way Unity handles terrain, all changes are stored into the terrain data, which is not a runtime copy. In your final game, when the terrain data on disk is not changed by the game, this won't be a problem, it only affects the editor.
- this script is not good for realtime, as Unity recalculates the terrain mesh collider whenever heights are changed. This will almost certainly lead to stuttering in your framerate.

Random Object Map (optional)

Precondition: Nothing

This script allows you the random-but-guided-by-a-distribution-map placement of any game object whatsoever. It works much like the tree, grass, etc. map scripts, except that it will interpret the map you give it as a grayscale "probability" of placing an object.

Max Number is the maximum number of objects the script will place. The actual number can be lower if your distribution map is largely black and the script has trouble finding valid spots, especially if you use a high *spacing* value (see below). For most reasonable numbers (i.e. a few hundred objects or less), the actual number of objects will usually be very close to the max number.

Spacing is the amount of space (in map units, not world units!) that you want to leave around each object. Use this to ensure that objects are not spawned so close that they intersect with each other. The default value of 1 should be good for most smaller objects, but if you use the script to place huge rocks, buildings or other large objects, you may want to increase it.

Above Ground is the number of world units that objects will be offset from the terrain (or other ground object). Use this for floating objects, or use a negative value to put objects partially into the ground (e.g. rocks, etc.). Note that this offset is relative to the unit center. A sphere will be placed half into the ground if *Above Ground* is zero, since that is where the object center is.

Use Layer allows you to control which layers are raycast against when finding the Y (height) value to place objects. You can use this to ensure that objects are only placed on terrain, for example, not on any objects that might sit on top of it already — or on the contrary, to make sure that your randomly spawned objects always sit on top of everything else.

Terrain/PlaceSelectionOnTerrain

This will simply lower/raise the currently selected editor object so that its origin sits square on the terrain. Very useful to place stuff on your terrain. Can have multiple game objects selected.

Scripts/PlaceOnSurface and Scripts/KeepOnSurface

Much like the editor script above, except that these are intended for use at runtime. One will place the object it is attached to on top of the terrain (or other objects) in Start(), the other will keep placing it in Update(). This is an easy way to animate walking people, driving cars, etc. without worrying about the Y axis.

Scripts/SpawnInArea

This allows you to add yet another procedurally generated element. This script will take a *spawnmap* and spawn/position something randomly inside that spawnmap on your terrain.

I use this to generate maps that exclude the corners of the map, the very edge, and anything where the steepness of the terrain is too high. On maps with water, you can also use spawnmaps to make sure that the player (or objects, etc.) isn't spawned over/in water.

Scripts/TerrainQualitySettings

A simple script that will change the terrain quality in response to the game's quality settings. You should check and modify the values included depending on your particular game, and you need to integrate this into your game so that it is attached to a game object but also called whenever the quality settings are changed.

Scripts/TreeExplosion

A simple script I released before on the Unity forums. It shows how you can access individual trees, for example if you want to make your forests destructable. It's simple, but really nifty.

FAQ

Splatmap Error

There is a rare case where you need to fix the splatmap setup. It can happen if there is an error during splatmap processing. If you get a fatal error complaining about an index being out of bounds, check if the 2nd splatmap field is really empty ("none") or if there is a blank (all black) splatmap there. This black splatmap is generated internally during processing and usually cleaned up at the end. But sometimes, it gets stuck. Simply click on "select" and set the 2nd field to "none" and things should be all fine.

Terragen Workflow

The various maps for use with the TerrainTools can be created in various ways. One possible way is to use the rendering abilities of Terragen 2.

Setup

Start with any terrain you like. It can be a heightfield or an entirely procedural terrain. This gives you the option of starting with a designed layout, for example by using L3DT and its "design map" feature to draw a rough outline of your terrain.

One way or the other, have a terrain loaded up.

Now set up a camera for all your renderings. This must be an orthographic camera, with a -90/0/0 rotation. Disable shadows, set the sunlight to a 90 elevation (i.e. straight up) and voila, you've got your top-down camera.

Heightfield

My method is described here: http://forums.planetside.co.uk/index.php?topic=7223.0

Note that heightmaps must be x^2+1, e.g. 513, 1025, etc.

Save as .exr and convert into .raw in Photoshop or whatever else you use. Make sure to save as a non-interleaved file, and that your byte-ordering is identical in Photoshop and Unity (i.e. if you save as Mac, you need to import as Mac, if you save as PC, import as PC).

Splatmap

Now set up surface shaders in four colours: Red, Green, Blue and Black. Set them up in whatever way you want your texture covering. I usually have one for strong slopes where rock should show through, and the others for features, heights or just random/fractale distribution. But use whatever you want.

One thing to keep in mind is that ‘‘Surface Layers'‘ do not blend in TG2. That means you have clearly defined textures. If you want them to smoothly blend not only into each other at the edges, but also within the areas (which if done right can give you great visual effects and make people believe you used a lot more different textures than you did), use shader overlays (i.e. no "apply low colour") or functions (e.g. "add color").

Then render from the same camera as your heightmap, this guarantees that heightfield and splatmap fit together.
Remember that you need x^2^ here, e.g. 512, 1024, etc.

Tree-, Grass- and Bushmaps

These follow the same basic principle as the splatmaps — set up shaders so that they display in primary colours where you want each tree/grass/bush type to be, use blending or additive mixing for areas where you want multiple types (e.g. white/gray = all types equally).

One difference is that "black" means "no trees/grass/bushes". On almost every map you will have areas without vegetation. Use a simple powerfractal blendshader to accomplish a "scattered woods" or "grass areas" effect. Reduce coverage to thin out vegetation equally.

Spawnmap

This is a ton easier than it appears to be. Simply add a surface layer where you exclude areas you don't want spawns to occur on. This will usually be steep slopes, but you can use it to exclude water areas. If your waterline is at 100, simply set a minimum altitude of 100 (or 120 with 20 soft zone, whatever suits you).

In addition, you can use a spherical distance shader as a blendshader to restrict spawn areas to the middle of the map, or at least exclude the very edges. Since you will usually have barriers or borders there, that makes a lot of sense.

For the colour, simply give it a white colour. In the spawnmap, white (or gray) means "spawn possible" and black means "don't spawn here". You can use grayscales if you want different spawning probabilities.

Overlays

Usually, you'll draw these in a painting program.

L3DT Workflow

This is from RoyS, posted to the Unity forums in this thread, with tiny changes to update the workflow to the new tool version:

BTW, here is my Tutorial for using Tom's Terrain Tools (the first version) with L3DT (I use Pro)…

Here's the steps I use to get my terrain into Unity.

1 — Generate your L3DT map via the Wizard if you're unfamiliar with it. Then…

```
File-->Export map-->(select) Heightfield
File format - (pulldown) select "RAW"
-type in a File Name
-check "resize for export"
-1025 size (default)
Operations-->Alpha Map-->Alpha Express (png)
```

The raw is your heightmap and the png is your splatmap. Move those to your Assets folder.

2 — Make sure "Tom's Terrain Tools" is installed per his instructions.

3 — In Unity

```
Terrain-->Create Terrain
```

Add your 4 textures in Unity as normal (you can get some nice terrain textures from http://cgtextures.com)

```
Terrain-->Import Heightmap RAW
--Bit16
--Windows
```

Now you should see your heightmap in your first colour chosen. If the mountains are too high, then go Terrain—>Set Resolution and change the 600 value. Just change those numbers until you get the desired height of your mountains and valleys.

Now for the rest of the colour of your terrain.

In Unity, select your png texture in the Project folder and in Inspector change it to RGB 24 bit, set the "Read/Write enabled" box and hit the Apply button. Now you won't get an error message to change it to 24 bit RBG as you've just done it. Now to add the splatmap…

open `Window-->Terrain Tool` and drag the splatmap into the first splatmap field.

If you're sand is in your mountain area and your grass under the sea, then just change the order of your textures until it's right (ie, the terrain paintbrush textures as if you were painting one from scratch — the same ones you loaded above).

Terrain Software

Highly subjective list based on my experience with the various tools.

Terragen 2

- http://www.planetside.co.uk/
- For OS X or Windows

TG2 is a node-based tool used mainly to render photo-realistic landscape views. With the correct setup (see Terragen Workflow) it can be used to render all the maps required for use in the Terrain Tools. It is extremely powerful, but requires a bit of a learning curve. Most terrain needs for a game can be realized without going into the node network too much, which makes it a lot easier to pick up. It's my personal favourite due to familarity and because it runs natively on OS X.

L3DT

- http://www.bundysoft.com/L3DT/
- For Windows

A graphical wizard-based tool that guides you cleanly through the process of generating maps. Its unique feature is the *design map*, which allows you to specify the target landscape in a rough sketch, which is then used by the fractal generator as a basis. It seems to have a node network behind the scenes, but I've never used that. Great for coming up with a basic map, I very often use the design map feature to get a heightmap to my specifications and then continue work in TG2 (L3DT can export to Terragen's heightmap format). RoyS posted a Workflow for L3DT.

World Machine 2

- http://www.world-machine.com
- For Windows

A node-based tool that has a very good node network visualisation. It supports shapes and other editing functions. I've only used it a few times myself, so I don't really have much of an opinion.

GeoControl 2

- http://www.geocontrol2.com/
- For Windows

The only tool that is controls-based, no node network in sight here. I only used the trial for a while. It seems powerful, especially because it has some features that all the other tools lack, like river generation.

Terrain Composer

- http://www.terraincomposer.com/
- for Unity3D

The first generator tool for Unity. It looks very impressive, I haven't used it yet, so I don't know what exactly it does and doesn't aside from what you can see in the videos. Very powerful, also fairly complex.

Limited support at
https://github.com/unitycoder/TomsTerrainToolsUnity

Check other assets:
https://assetstore.unity.com/publishers/3747?aid=1101lGti