version 1.1 20161224-0515 © 2015-2016 NT Entertainment

1. What is it?

HammUEr is an Unreal Engine 4 plugin that lets you build your levels with Valve's Hammer map editor or a Quake-based tool like Trenchbroom and imports them into your Unreal Engine project as a collection of meshes placed in your scene the way you want, where you want them with just a few simple clicks. Want to change something? Make your edits in your favourite tool, save, re-import, done.

During the importing process, you can assign materials already in your project to the original material names - or if you have the source images (TGA/BMP/PNG/JPG)/VTF/WAD(Quake1 & Halflife)/WAL(Quake 2), you can import them into UE as material instances based on a template material of your choice and let HammUEr automatically find and assign the right ones.

Want to start building a level in Hammer with your UE materials for true WYSIWYG level editing? HammUEr can do that, exporting your UE material textures to VTFs for easy use with a simple click.

Want to use models? You can now also import Source MDL files, which will automatically be placed in the right positions when importing a VMF map.

For the latest news, follow @hammUEr on Twitter.

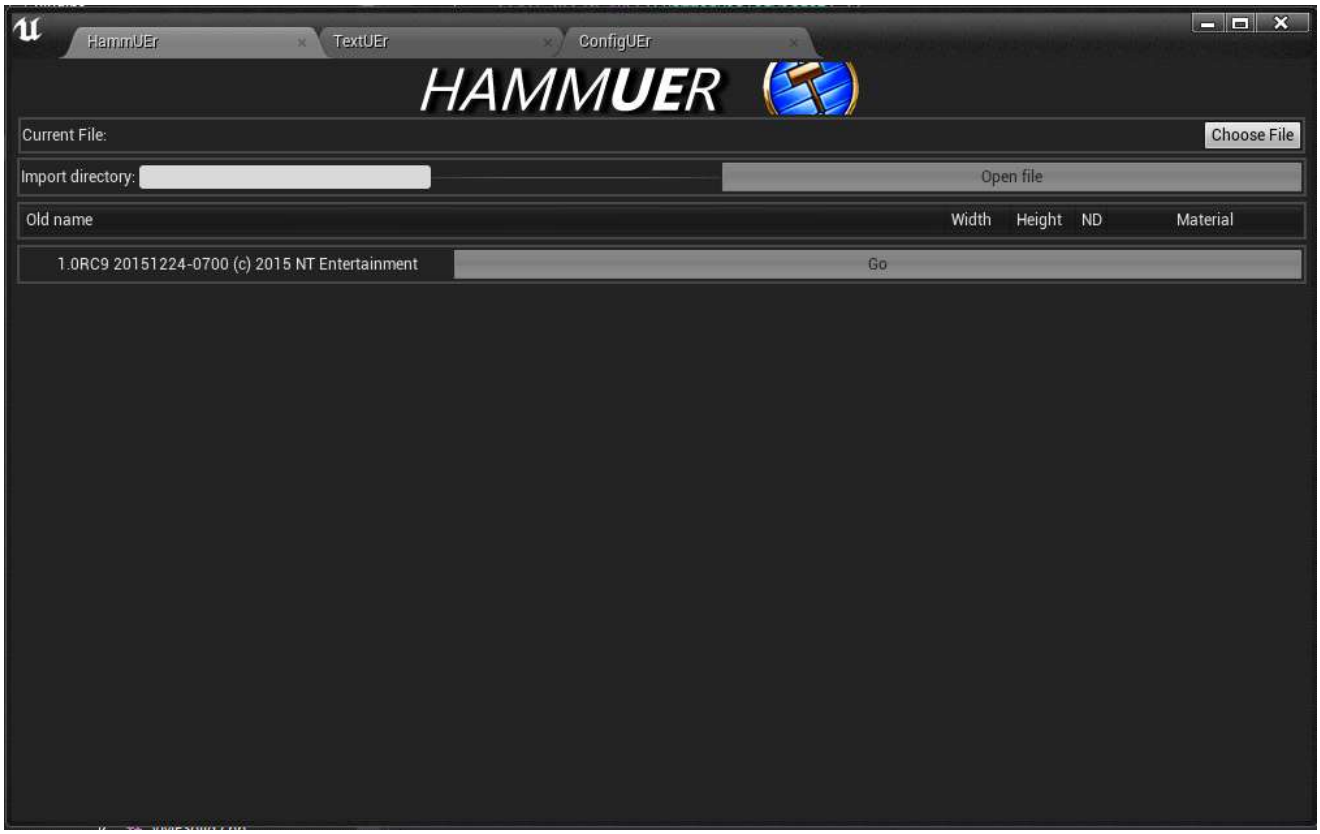
System Requirements:

- Windows system
- Unreal Engine 4.9 or higher (binary release only?)

Installation:

Create a Plugins directory under your project directory, then unpack the archive into it, so you end up with <project>\Plugins\HammUEr. If the HammUEr window does not open, enable the plugin from your Plugins menu.

2. The different tabs, and their options



2.1. HammUEr

This is the main tab, which contains the main level loading logic.

To import a MAP/VMF file, click "Choose file", then "Open file". This will (re-)load your master material list, which stores all known material information - either encountered in other files, or discovered in your project (see section 2.2) - and will try to find matches for the materials referenced in the file. If there are none, it will traverse your asset database to see if there's any matches there.

This material list will then be displayed below, allowing you to change the texture width & height, nodraw-flag (ND) and material to use for each one. The find material input box finds the first instance of whatever you typed in the input box.
Note that Source map UVs rely on the original texture width/height for calculation, so they need to match whatever you used inside Hammer.

The import directory gets automatically filled in based on the map file name, but you can edit this to whatever you want.

When you're ready, click "Go" and wait for it to do its business.

Can be shortened to only reimport certain meshes or entities by using the settings tab. (See section 2.3)

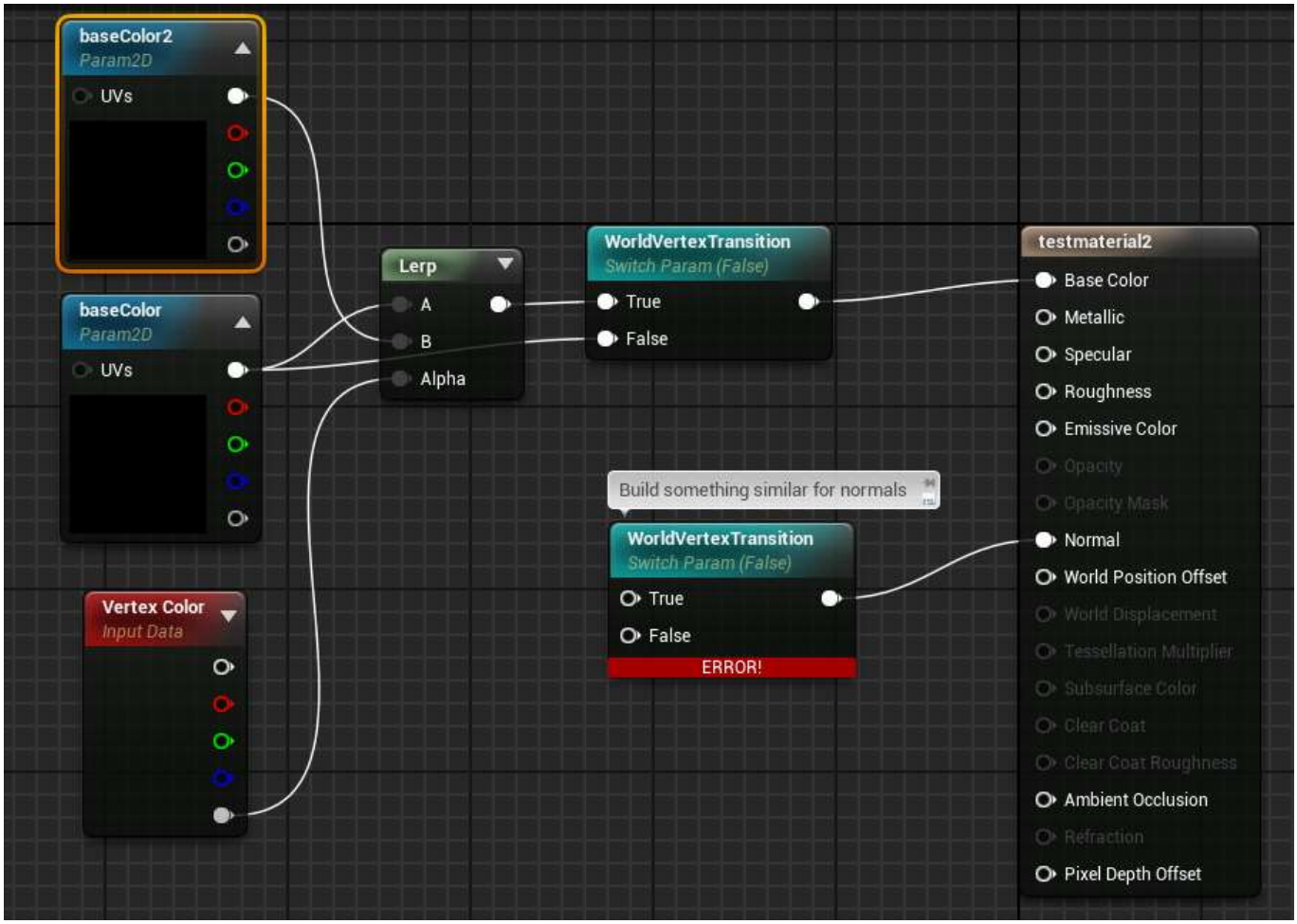
2.2. TextUEr

This tab is responsible for all material exporting/importing/discovery.

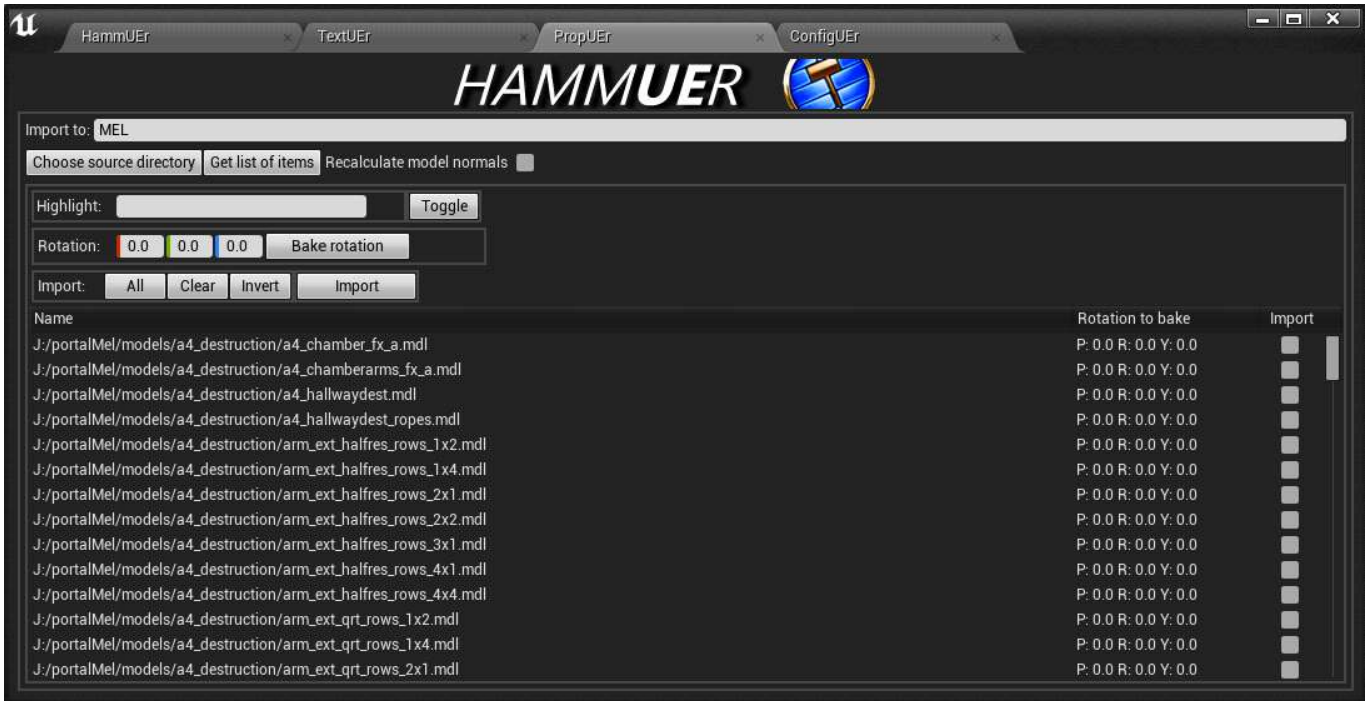
Export section: This exports every non-engine material in your project to a VMT/VTF combo in a directory you select by clicking "choose export directory". This will then select the main BaseColor texture of each and write it to a non-mipmapped VTF file for use in Hammer. You can finetune which textures are definitely not to be used by using the input boxes below. By default, textures ending in _M and _N (mask and normal) are ignored.

Discovery section: Goes through all materials in your project and adds the unknown/new ones to the master material list with the correct texture sizes. If "rebuild" is checked, it rebuilds the entire master list from scratch like this. Warning: This will invalidate all nodraw-flags you set previously.

Import section: This imports everything in a recognised format (JPG, PNG, BMP, TGA, VMT/VTF, WAD (both Quake and Halflife 1) and WAL) in the directory you choose (and its subdirectories) to the content browser under the directory specified in the inputbox below. It creates material instances based on the selected source material, by replacing the selected BaseColor/Normal/Roughness material chain parameters in it. (See 'Parameter' dropdowns in the image above) The actual names of the texture parameters don't matter, but you need at least a BaseColor parameter for a material to be eligible.



If the VMT is a WorldVertexTransition type and the source material contains a StaticSwitchParameter called WorldVertexTransition, this will be set accordingly, and the second, non-selected texture parameter in a chain will be used. (See example material)
When not part of a VMT Source material file, HammUEr will try to generate a material for leftover imported textures, except for the ones ending in the suffixes listed in the "Normal names" and "Don't build" input boxes.



2.3. PropUEr

This tab is responsible for all prop (Source MDL or SMD) importing.

Import to: This is the name of the directory in the content explorer that you want to store the imported meshes to.

Choose source directory: Directory to import. This includes all subdirectories, which will be re-created under the content directory you specified.

Get list of items: Finds all supported models in the source directory and fills the list below.

Recalculate model normals: Ignore normals stored in the model and generate new, smoothed ones.

Highlight: Finds all models in the list that contain what you typed in the input box, and pre-selects them.

Toggle: (Un-)Set selected models to import.

Rotation: Roll/Pitch/Yaw values to bake for selected models, gets set by clicking **Bake Rotation**.

All/Clear/Invert: Easily set all/none/the inverse of what's currently marked to import.

Import: Actually import the models checked to import, with baked rotations, if any.

2.4. ConfigUEr

Various settings and fine-tuning tools

2.4.1. General

Scale conversion: How many original units are in the standard UE unit of 1 meter/100 cm. The example value in the image above should be okay for most Source-based maps, for instance.

Light intensity factor: Given that UE lighting works in a totally different way, use this value to multiply the original Source light intensities. Not used for Quake map files.

Round points to nearest integer: This tries to minimise floating point errors to a minimum by *aggressively* rounding everything to an integer value. Probably not for general usage, but can be used to fix the weirder importing errors for complex geometry. (See section 3.1)

Error correction range: More floating point error catching. Within what range points in space should be considered the "same" while building a mesh. A default range of about 1.0f gives the best results, but this can be adjusted up or down as needed.

Intersect factor: Even more floating point error adjustment. A value of 100 *should* be enough for most maps, but if you have a lot of thin, low, almost flat pyramid shapes (to pick a totally random example, I swear), you might want to bump this up to 10000.

Decouple meshes from origin: By default, everything gets imported relative to 0,0,0. If this is checked, each mesh is rebuilt around its own personal origin and then moved to the correct location in the scene.

Adjust prop scale: When to apply scale conversion to Source models that get imported. Don't apply, apply it when importing the prop in PropUEr and bake it into the UE mesh, don't bake the scale but adjust it in-engine when building the map data.

Import level with offset: if checked, all the meshes will be offset by the following XYZ values

Rotate meshes after import: if checked, your entire map will be rotated by the following roll/pitch/yaw values

NoDraw brushes have two-sided shadowing: if checked, meshes that lost at least one side due to the user's NoDraw rules will be set to two-sided shadowing to stop lights leaking through 'open' meshes.

Lightmap size: The default lightmap size for each imported mesh. You probably don't want to go above 64, unless you enjoy interminable lighting builds.

Default texture size: The default texture width & height for unknown materials.

Do not use texture alpha on import: Many Source textures have erroneous alpha information stored in them, which makes walls and objects partially transparent where they should be solid. Ticking this will automatically remove all alpha information from textures on import, taking care of this problem.

Default channel for displacement alpha: The channel to encode the alpha blend values for a displacement map to, so it can be used in your texture blending material

Color to replace when importing HL1 textures: The color to replace with alpha when importing HL1 textures, which used a masking color (bright blue, by default)

Verbose debug messages to output log: Shows the inner workings of the plugin, with lots of debug information logged. Mostly useful for me if something goes wrong.

Do not generate a separate VMF material list file: When unchecked, this generates a .vmfmaterialList file in the directory where your vmf file is stored, where it dumps the materials, sizes, nodraw flag and UE material names for each in a readable format you can edit as you need. This file then gets loaded every time you open the file. *Probably* not really useful these days, with the material assignment straight in the editor and editable width/height/nd options, but it's still there for people that want to do it the oldskool way.

Save settings: Self-explanatory, really, I'd think.

2.4.2. Session specific

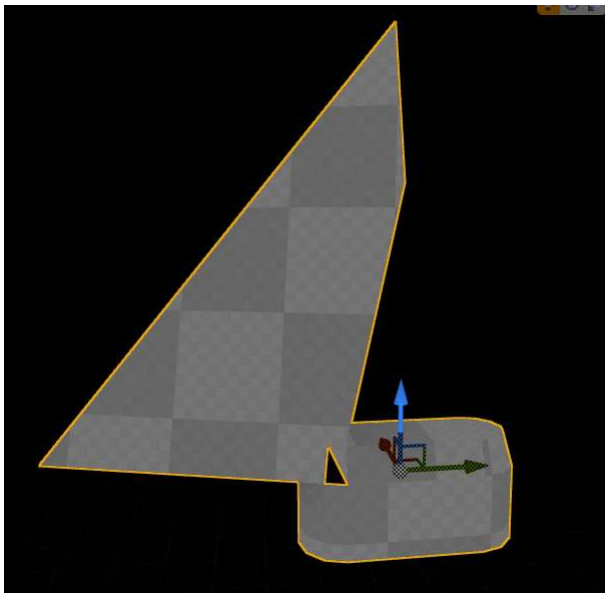
As alluded to above, these 2 options allow you to limit the amount of things to reimport if something happens during the import and you want to tweak the settings. The first is for singlemesh/groupmesh numbers (you can find these at the end of each one, e.g. c1a0_empty_singlemesh_24) in a comma separated list. The second input box works the same, but is specifically for entity numbers (again, at the end of of each name, c1a0 _empty_entitymesh_1043).

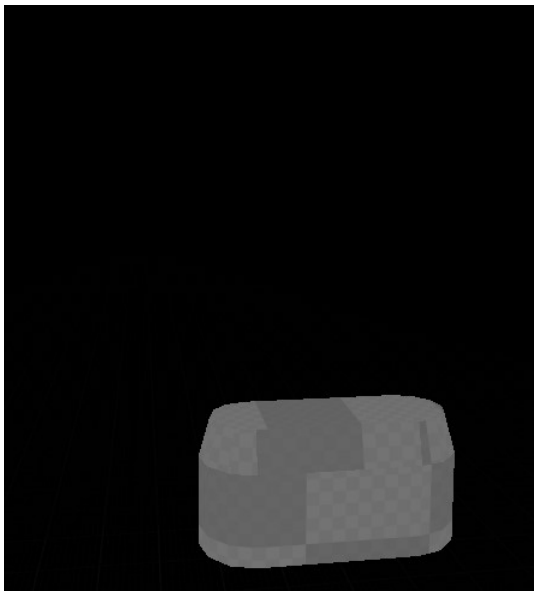
To *not* load a type (if you only want to reimport entities, for example), just enter a 0 in the relevant input box.

3. When things go wrong

3.1. Help, my mesh came out all spiky

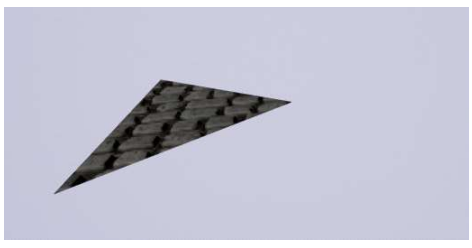
So you imported your map, and you're greeted by the horrible sight on the left. Don't panic. First, try reimporting the affected complex meshes (as per 2.3.2) with the "Round points to nearest integer" option selected. This should fix floating point errors like these.





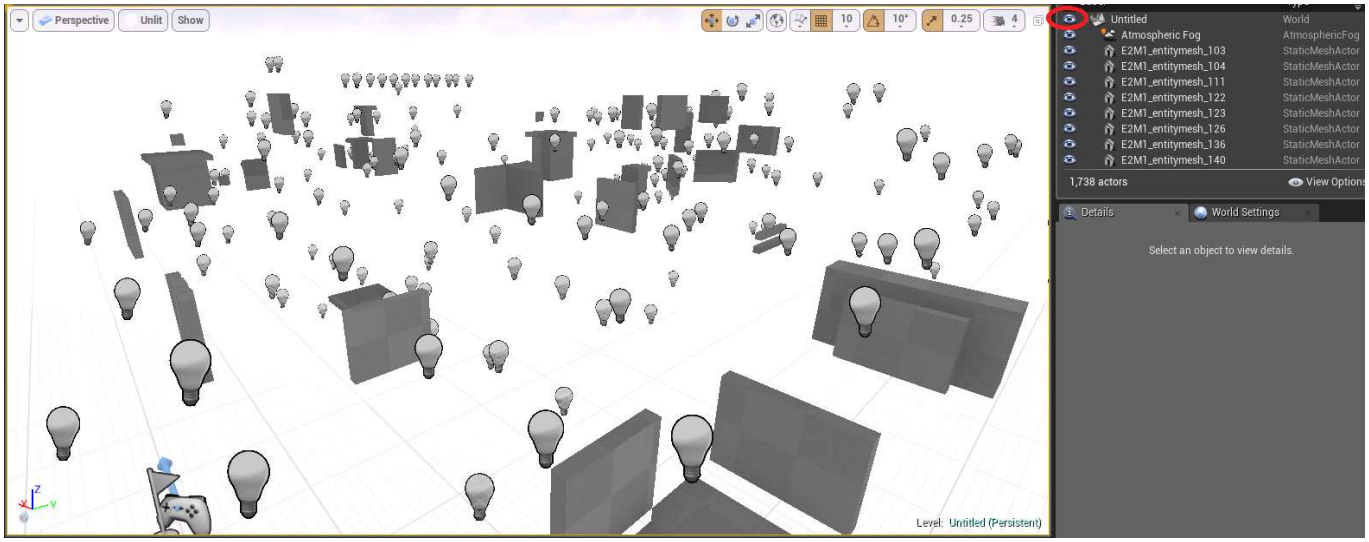
3.2. Help, my weird, thin shapes are broken

You've imported your map, but your more extremely thin and angular meshes look broken, missing most of their sides. Did you change the intersect value from 10000 to something lower at an earlier point in time? Reset it to 10000 and reimport the affected meshes to make the floating point error fixing incredibly lenient again.



3.3 Help, half my level is invisible

You've imported your map, but it looks more like a random collection of meshes than your map, as in the image below? This happens rarely, normally, but don't worry. Just click the circled eye icon at the top of your world outliner twice, and everything should appear.



3.4 Help, I've imported textures/models, but they're invisible in the content browser

This can happen sometimes, especially when you've imported a lot of them in one go. The best way to handle this is to just close the editor after you're done importing, and reopen it, after which they'll all be visible.