Glyph 3d operating manual

Contents

Change tracker

1.0.0.0

- First release

1.0.0.2

- Added general mesh parameters.
- Added the option to calculate tangents for the generated meshes.

1.2.0.0

- Reworked the UI to better fit low resolution screens.
- Fixed error joining meshes when no materials were selected
- Added internal script to save created objects as prefabs
- Prepared internal structures for runtime building

Introduction

Welcome to the world of Glyphs – in 3D! This Unity^{tm} editor extension will allow you to import True Type Fonts (TTF) directly into Unity – in 3D, without going through a 3rd party 3D program, such as Blender or Maya.

Installation

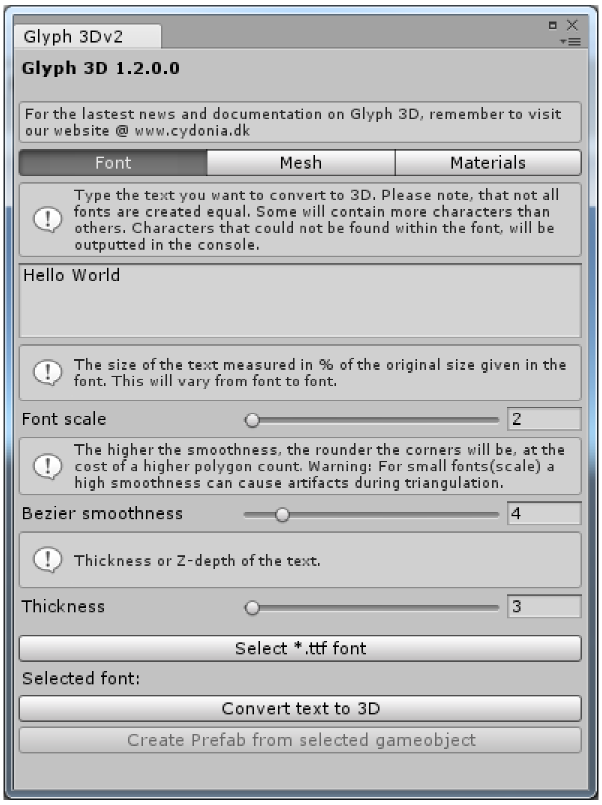
The provided DLL file should be placed in the Unity “Editor” folder. When Unity has loaded the file, a new menu item will appear in the “GameObject” menu with the title “Glyph 3D”.

Fonts

Currently, most TTF fonts are supported. However, only a small subset of OTF fonts is supported. Basically if you can use the font in Windows^{tm} it will work in Glyph 3D. If you are afraid if a font will work in Glyph 3D, you can send it to us, and we will tell you if it will work or not before you purchase Glyph 3D.

How to use it?

The interface for Glyph 3D is pretty straight forward. Every option has a small help text to help you. What follows now is a guided tour around the tabbed interface.


Glyph 3Dv2

Glyph 3D 1.2.0.0

For the lastest news and documentation on Glyph 3D, remember to visit our website @ www.cydonia.dk

Font | Mesh | Materials

Type the text you want to convert to 3D. Please note, that not all fonts are created equal. Some will contain more characters than others. Characters that could not be found within the font, will be outputted in the console.

Hello World

The size of the text measured in % of the original size given in the font. This will vary from font to font.

Font scale: 2

The higher the smoothness, the rounder the corners will be, at the cost of a higher polygon count. Warning: For small fonts(scale) a high smoothness can cause artifacts during triangulation.

Bezier smoothness: 4

Thickness or Z-depth of the text.

Thickness: 3

Select *.ttf font

Selected font:

Convert text to 3D

Create Prefab from selected gameobject

Text

The first thing you meet is the font text. Here you can input the text you would like to have converted into 3D. Tabs are not supported, but newline is. Default is ‘Hello World!’.

Font scale

All fonts come with a set size. Some are huge while others are small. This setting is a percentage scale of the original font’s size.

Bezier smoothness

Some fonts contain soft curves expressed in Bezier curves. This setting allows you to specify the number of steps a curve should contain. The more points, the softer the curve will look – and the more triangles will be produced. The setting 4 looks pretty good, while 10 looks awesome. Play with it and see how it looks – and how many polygons gets produced ☺

Thickness

Thickness, or character depth, allows you to specify the depth of the characters. You can always scale the letters on the z-axis afterwards if needed be.

Select TTF font

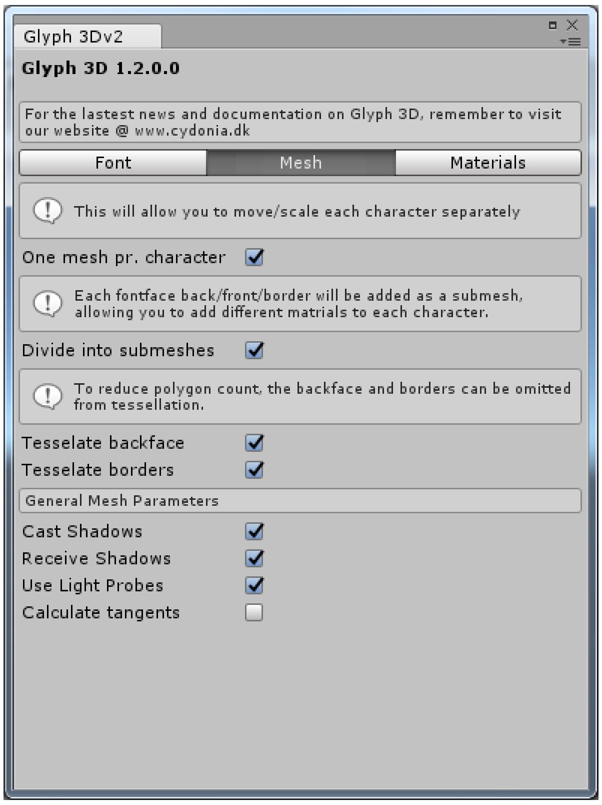
Glyph 3D does not use system fonts. You have to select a font file on disk. The font will **not** be included into your project. It will only be read during text processing. It’s also possible to select *some* OTF fonts.

Convert text to 3D

Press the button and watch the magic.

Create Prefab from selected gameobject

Select the created objects in the game hierarchy, and press this button to save them to disk as prefabs.


Glyph 3Dv2

Glyph 3D 1.2.0.0

For the lastest news and documentation on Glyph 3D, remember to visit our website @ www.cydonia.dk

Font | Mesh | Materials

This will allow you to move/scale each character separately

One mesh pr. character ✓

Each fontface back/front/border will be added as a submesh, allowing you to add different matrials to each character.

Divide into submeshes ✓

To reduce polygon count, the backface and borders can be omitted from tessellation.

Tesselate backface ✓

Tesselate borders ✓

General Mesh Parameters

Cast Shadows ✓

Receive Shadows ✓

Use Light Probes ✓

Calculate tangents ☐

One mesh per character

This is all about draw calls and what you would like to do with the characters afterwards. If this option is activated, each letter in your text will produce one mesh in Unity. This allows you to rotate/scale/position each character as you like.

Divide into submeshes

This will produce 3 submeshes per character - unless you omit the back face/border (see below). This will allow you to give each face of the letters its own material. If ‘One mesh per character’ is disabled, it will work on all the produced text.

General Mesh Parameters

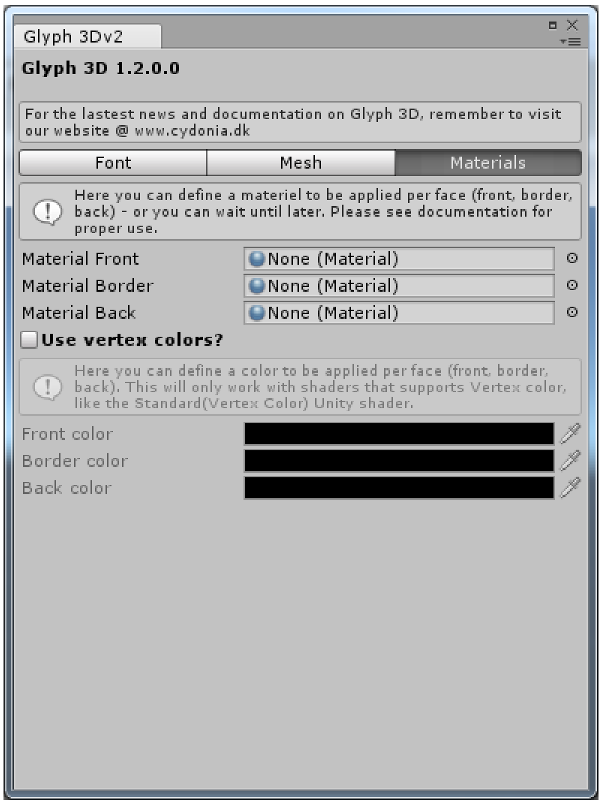
Settings selected here, will be applied to all generated meshes. These are standard Unity mesh properties.

Calculate tangets

Select if tangents should be calculated for the mesh. Tangents are usually only useful during bump mapping. If you don’t need them, leave this box unchecked.

Tesselate back face / borders

Let’s say you want to reduce the number of polygons in your text, and you know that nobody will ever see the back face of a letter – then why generate it? Save as many polygons as you can. Perhaps you only need the front face – disable the borders and the back face and save a lot of polygons.



Glyph 3Dv2

Glyph 3D 1.2.0.0

For the lastest news and documentation on Glyph 3D, remember to visit our website @ www.cydonia.dk

Font | Mesh | Materials

Here you can define a materiel to be applied per face (front, border, back) - or you can wait until later. Please see documentation for proper use.

Material Front: None (Material)
Material Border: None (Material)
Material Back: None (Material)
Use vertex colors? ☐

Here you can define a color to be applied per face (front, border, back). This will only work with shaders that supports Vertex color, like the Standard(Vertex Color) Unity shader.

Front color
Border color
Back color

Materials

Select the materials you would like applied to each letter face.

Use vertex colors

You are not really into all those textures – and would just like a simple color for your fonts. This is where you select a color per face. Just remember to select the correct shader for your materials in order for them to be visible. Unity has one build-in for just this purpose.

FAQ

Some of my letters are missing? Where did they go?

Not all fonts are created equal – or complete. What we are trying to say here, is that some font artists only create capital letters for their fonts, while others create both. Some omit the numbers, and others include special characters only available in their own country. So do yourself a favor, and try the font in a word processing program first, eg. OpenOffice, LibreOffice or even MS Paint. Just remember to install the font first.

I get a completely different set of characters on-screen than those I typed?

This will only happen, if your font has the wrong CMAP version. An error should have notified you about the problem in the console window. We only (currently) support fonts that used CMAP version 4.

My letters have incorrect spacing between them, what gives?

Two things could have happened. The first might be that your font is using an unsupported KERN table version, or a sub-table in the KERN table is using an unsupported version. If this happens, we fall back on the letters width plus a small margin. The second reason is usually, that the font does not contain any KERN table. Here we again fall back to the letter width.

My font characters are overlapping a bit, why?

Remember that these fonts are designed for 2D. Some of the font designers, usually those designing Calligraphy fonts, tend to overlap just tiny bit on purpose. You will have to manually shift the overlapping character a bit if you want to avoid it.

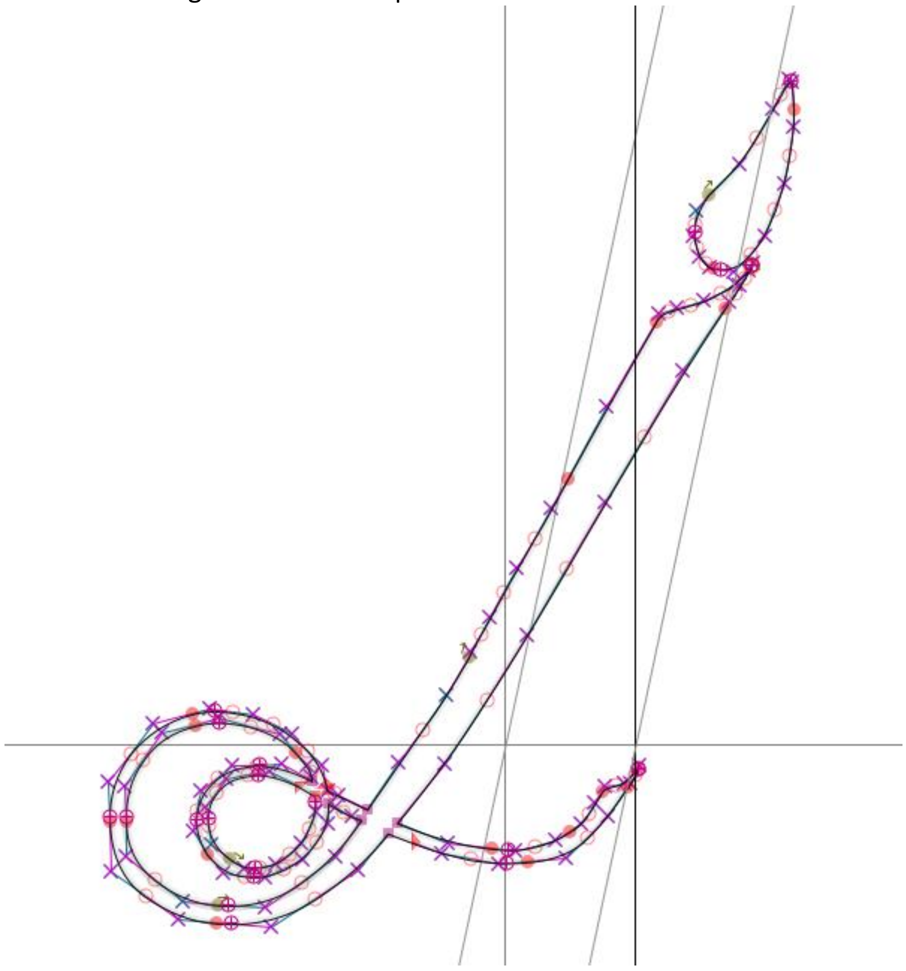
I have an OTF font, but I can’t open it?

True – currently we only support TTF fonts, and only a specific subset of those. See elsewhere in the FAQ. However, since OTF is based on TTF, **some** OTF fonts actually work just fine.

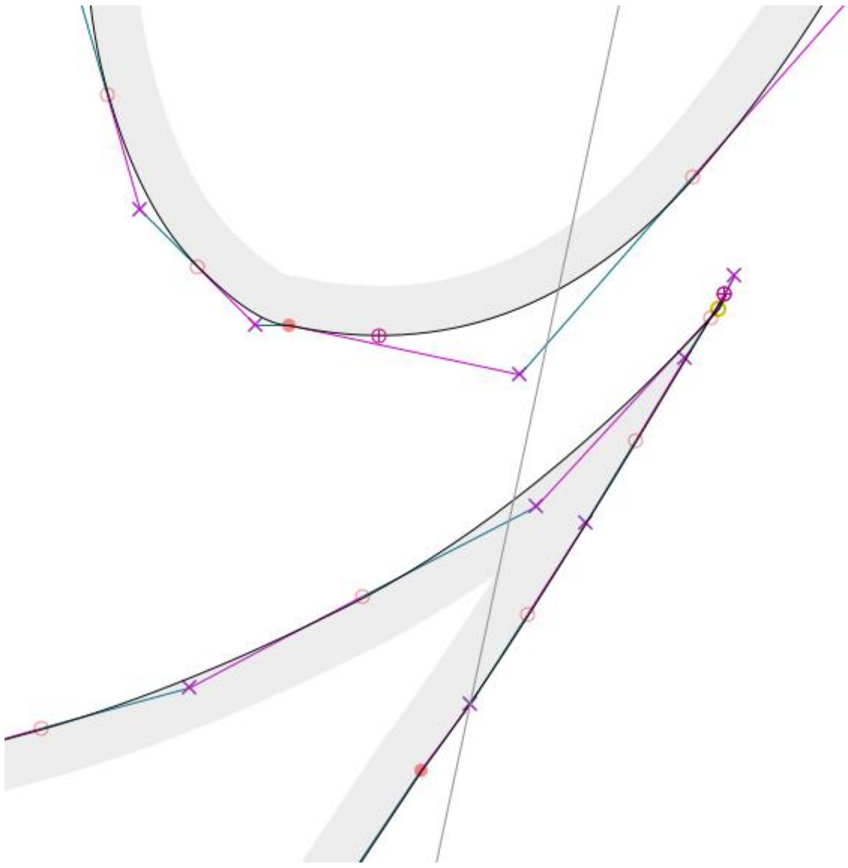
A letter is not rendered correctly/looks like gibberish – what's the problem?

It's usually caused by the font artist, creating the font in a bad way, or just being sloppy. Here is two examples:

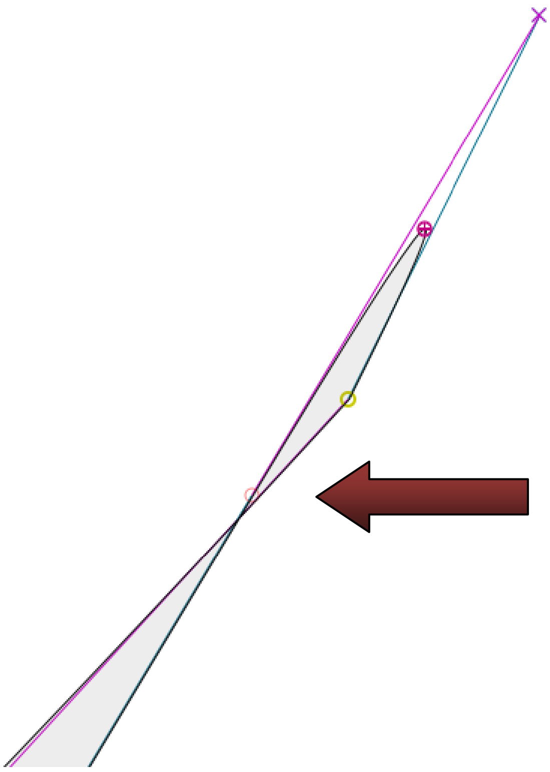
We are rendering the letter 'j' from the font "Queen of the moon" – found online. Here is how the font looks from inside FontForge - First the full picture:



This font will render incorrectly due to an overlapping contour. Overlapping? Where? Look at the top of the "i" – not the dot, but below it – let's zoom in real close.

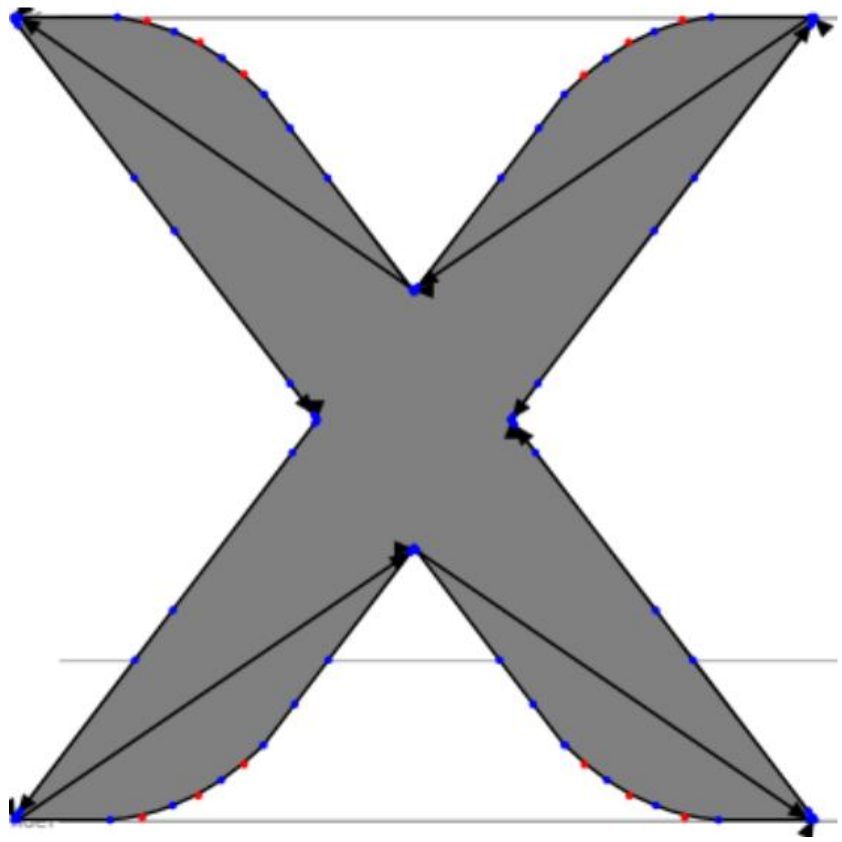


Still can't see it? Let's move in even closer.



If you follow the line from the left, you can see it cross the other line, before coming back around and going down. This little stunt will throw off the tessellation routine giving you a somewhat undesired result. However, it took less than two minutes to fix it in FontForge.

Another example we have often seen is the following.



This font is called “Maestro-Regular” and you are looking at the letter X – in 2D. This particular letter contains a stunning 17 contours – where only one was needed. We might be able to guess how to piece the letter together, but currently we take the contours in the way they appear in the file, usually in order. However, this font got its contours mixed up really good, dividing it into many, many pieces, making it impossible for us to tessellate correctly. The rest of the letters in this font renders perfectly – apart from the X. Currently we have no means to get around this problem. Remember – all these font were designed for 2D – not 3D ☺

So my font is fubar, what can I do about it?

There exists plenty of open source font editing programs you can download, and correct errors like the above, eg. FontForge (https://fontforge.github.io/en-US/) or BirdFont (https://birdfont.org/) to name a few.

Why would I want to set the font thickness/scale?

The simple reason is – if you already scaled it to the right dimension, Unity does not have to scale it again – saving you some CPU/GPU cycles.

I’m using a dingbat font – will that work?

It most likely will. However, dingbats often contain lots of contours – often in the wrong order, or overlapping contours. But give it a try.

I need to reduce the number of draw calls – is that possible?

Well – yes, sort off! If you deselect the “One mesh per character” option, we will only use up to 3 materials in total – batching letters together. It will not reduce polygon count. You can also try and remove the back face if you know you will never see it. That alone should reduce the number of polygons by 1/3.

Is my font included in my final build?

No, your (binary) font is NOT part of your final build. Your font file is only used while generating the 3D mesh.

Can I generate letters runtime?

Currently – no! However, this feature is planned for a future release.

Will it work on an Apple^{tm} computer?

Yes, it will. No OS calls are made to process the font. It should work on all devices that support the Unity editor.

Can I combine texture materials and vertex colors?

Yes, with the appropriate shader they can be combined.