TYPE3D

3D FONT COLLECTION
v1.1.2

Created by Ripcord Development
info@ripcorddev.com

This project has been thoroughly tested for bugs before being sent to the Unity Asset store. If you do find an issue with this package, please contact us before posting a negative review on the Unity Asset Store. We are more than willing to help solve any issues you may encounter.

Type3D is a collection of 3D font characters which can be arranged on their own, or can be programmatically using the TypeGenerator tool. Through the TypeGenerator you can control the size, spacing, and alignment of the Type3D characters, as well as preview materials and animated effects. When you're happy with the result, you can save the Type3D object as a prefab for use in your project.

PREFABS

The prefabs folder is divided into 4 different sections, Characters, Fonts, Timers, and Scoreboards.

- **Characters** – The individual character prefabs that make up a Type3D font. They are organized by font style.
- **Fonts** –These objects contain a reference to each character within the font that can be accessed through the TypeGenerator.
- **Timers** –Premade time display objects using Type3D. Just send your time value to the attached script and the object will display that value using Type3D characters.
- **Scoreboards** –Premade score display objects using Type3D. Like the timers, just send your score value to the attached script and the object will display that value using Type3D characters.

MODELS

Currently each **Type3D** font contains 83 characters which consist of uppercase and lowercase letters, as well as numbers, and many of the most common symbols and punctuation. The pivot point of each character is centered at its bottom. The entire character set for each font is contained within its own FBX file.

MATERIALS

The materials are broken down into two categories: Generic, and Font-Specific. Generic fonts are intended to be applied to any of the **Type3D** characters, while Font-Specific materials are optimized for the UV map coordinates on the character meshes. Feel free to apply your own materials, or modify any of the ones included to get the right look for your project.

SCRIPTS

Type3D contains a number of scripts that can help generate the perfect 3D text for your project. Depending on your needs you may not even need to use any of these if all you're after are the 3D models in this package. Below is an overview of each script and what it does.

- **TypeGenerator** – This script can create a set of Type3D characters based on the selected input. It gives full control over the various properties of the font and allows you to convert your object to a prefab. It's been built in a way that you can use it to create your own stand-alone Type3D objects, or the generator itself can be integrated directly into your project.

  - **Font3D List** – A list of all the 3D font sets available to the TypeGenerator.
  - **Font3D Index** – The index of the currently visible Type3D font from the font list.
  - **Input String** – The string of characters that will get converted into Type3D objects.
  - **Type3D List** – A list of the Type3D characters currently in the scene.
  - **Kerning** – The amount of space between each Type3D character.
  - **Alignment** – The position of the Type3D characters relative to the Type3D container.
  - **Type Effect** – The animated effect that will play on each character in the Type3D container in sequence.
  - **Loop Type** – If and how the animated effect will behave once it has affected each of the characters in the sequence.
  - **UI Components** – Each of these properties corresponds to a piece of the TypeGenerator UI. You can use as many or as few of these as you wish. Each component features a Boolean to toggle whether the feature is used or not, and a reference to the UI component that will control that feature.

This script requires the free iTween package to function properly, see below

- Font3D – Controls the character set for the font, as well as any materials associated with it.
  - **Font Name** – The written name of the font as it will be displayed in the TypeGenerator.
  - **Characters** – The list of character prefabs associated with the font.
    - Please note that the index of each element in the list corresponds to the Unicode Decimal value for that character. (Ex. Element 65 is always uppercase letter ‘A’, Element 99 is always lowercase letter ‘c’).
    - There are gaps in the list where no prefab is specified. These gaps line up to special characters that were not included in this package, because they’re not as commonly used. They may be included in a future update.
  - **Replacement Character** – If the user types a character that is not supported by a Type3D font, that character will be represented with this object instead.
  - **Materials** – A list of all the materials that can be applied to the object through the TypeGenerator.
- TypeEffects – This script controls all the settings for the animated effect on the character prefabs.
  - **Characters** – All the characters contained in the Type3D container, listed in order.
  - **Type Effect** – The animated effect that will play on each character in the Type3D container in sequence.
    - None – The Type3D characters will not have an animated effect
    - Bounce – Each Type3D character will move up and down a bit on its Y axis
    - Flip – Each Type3D character will rotate 360 degrees on its X-axis
    - Scale – Each Type3D character will scale up and back down
    - Shake – Each Type3D character will rotate back and forth a little bit on its Z-axis
    - Spin – Each Type3D character will rotate 360 degrees on its Y-axis
  - **Loop Type** – If and how the animated effect will behave once it has affected each of the characters in the sequence.
    - None – No animated effect will play unless specifically called through script.
    - PlayOnce – The animated effect will play once only when the object first spawns.

        - OrderedLoop – The animated effect will loop through all characters starting at the beginning of the list all the way to the end.
        - RandomLoop – The animated effect will loop through all characters in a random order. Once it plays on each character it will start over with another random order.
        - PingPong – The animated effect will loop through each character from left to right, and then back from right to left.
    - **Effect Amount** – A multiplier to modify the range of motion on the animated effect. Does not affect all effect types.
    - **Effect Delay** – The number of seconds before the effect acts on the next character in the sequence. Set this value to 0 to affect all characters at the exact same time.
    - **Effect Frequency** – The number of seconds before the effect repeats on the first character in the sequence.

This script requires the free iTween package to function properly, see below

- Timer3D – This script converts a float-based timer into a collection of 3D characters representing the time.
    - **Numbers** – The mesh objects that will represent each of the numbers from 0-9.
    - **Timer Digits** –The GameObjects that display up each individual character of the timer.
- Score3D – This script converts an int-based number into a collection of 3D characters representing that number.
    - **Numbers** – The mesh objects that will represent each of the numbers from 0-9.
    - **Score Digits** – The GameObjects that display up each individual character of the score.

3rd Party Scripts

- **iTween**

The free iTween package must be installed for this package to function properly

This package makes use of the freely available iTween package for Unity, created by Bob Berkebile (pixelplacement). To learn more about iTween or to help support its development, you can check out the documentation here:
http://www.pixelplacement.com/itween/index.php

iTween can be downloaded from the Unity Asset Store here:
https://assetstore.unity.com/packages/tools/animation/itween-84

To install iTween simply import its asset package. Nothing further needs to be done.

UNICODE VALUES

Below is a list of the Unicode values used by **Type3D**. This list stops at the end of the Basic Latin character-set, though there's no reason why you couldn't expand the character list for your own project.

Value	Glyph	Description	Value	Glyph	Description	Value	Glyph	Description
032		Space	064	@	At Sign	096	`	Grave Accent
033	!	Exclamation Mark	065	A	Uppercase A	097	a	Lowercase A
034	“	Quotation Mark	066	B	Uppercase B	098	b	Lowercase B
035	#	Number Sign	067	C	Uppercase C	099	c	Lowercase C

036	$	Dollar Sign	068	D	Uppercase D	100	d	Lowercase D
037	%	Percent Sign	069	E	Uppercase E	101	e	Lowercase E
038	&	Ampersand	070	F	Uppercase F	102	f	Lowercase F
039	‘	Apostrophe	071	G	Uppercase G	103	g	Lowercase G
040	(	Left Parenthesis	072	H	Uppercase H	104	h	Lowercase H
041	)	Right Parenthesis	073	I	Uppercase I	105	i	Lowercase I
042	*	Asterisk	074	J	Uppercase J	106	j	Lowercase J
043	+	Plus Sign	075	K	Uppercase K	107	k	Lowercase K
044	,	Comma	076	L	Uppercase L	108	l	Lowercase L
045	-	Minus Sign	077	M	Uppercase M	109	m	Lowercase M
046	.	Period	078	N	Uppercase N	110	n	Lowercase N
047	/	Slash	079	O	Uppercase O	111	o	Lowercase O
048	0	Zero	080	P	Uppercase P	112	p	Lowercase P
049	1	One	081	Q	Uppercase Q	113	q	Lowercase Q
050	2	Two	082	R	Uppercase R	114	r	Lowercase R
051	3	Three	083	S	Uppercase S	115	s	Lowercase S
052	4	Four	084	T	Uppercase T	116	t	Lowercase T
053	5	Five	085	U	Uppercase U	117	u	Lowercase U
054	6	Six	086	V	Uppercase V	118	v	Lowercase V
055	7	Seven	087	W	Uppercase W	119	w	Lowercase W
056	8	Eight	088	X	Uppercase X	120	x	Lowercase X
057	9	Nine	089	Y	Uppercase Y	121	y	Lowercase Y
058	:	Colon	090	Z	Uppercase Z	122	z	Lowercase Z
059	;	Semicolon	091	[	Left Square Bracket	123	{	Left Curly Bracket
060	<	Less-than Sign	092	\	Backslash	124	|	Vertical Bar
061	=	Equal Sign	093	]	Right Square Bracket	125	}	Right Curly Bracket
062	>	Greater-than Sign	094	^	Circumflex Accent	126	~	Tilde
063	?	Question Mark	095	_	Low Line			

For a full list of Unicode values and their corresponding characters, please see this page on Wikipedia: https://en.wikipedia.org/wiki/List_of_Unicode_characters

GETTING STARTED

There's two different ways to get started with Type3D. The first method is to simply take any of the character prefabs and arrange them into your own words. It's basic, but it gives you full control over the final look. The second (and more fun :D) method is using the TypeGenerator.

Using the TypeGenerator

First, open up the typeGenerator scene found in the DemoScenes folder and hit Play in the editor. From here you'll have access to all the features that make up Type3D. Start by entering some text then press the "Make Type3D!" button. You've made your first Type3D text! Simple eh?

With your 3D text on the screen, start clicking through the various options to see how you can modify it. You can change the font style, size and spacing of the text, as well as modify the material or add animated effects. When you're happy with your creation, hit the "Save Type3D!" button to create a prefab in the Type3D/Resources folder.

Once you have created your Type3D prefab, feel free to drag it into your scene and modify it further with your own materials, objects, etc.

Notes:

- *If you change any of the animated effect settings while an animation is in mid sequence, there's a chance it can cause the characters to not revert to their original positions once the sequence completes. If this ever happens, just press the Make Type3D! button to reset the characters.* ☺
- *Pay attention to the Delay and Frequency settings. If the delay multiplied by the number of characters is greater than the frequency, the animation will start the sequence again before the previous sequence has finished. This could have undesired effects on the overall animation.*

```
if (delay * number of characters > frequency) {
        return potentially bad animation;
}
```

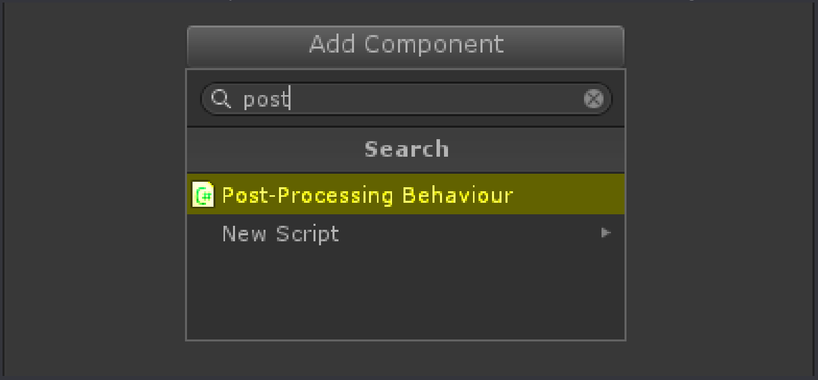
SCENE SETUP

The following steps are only required if you want to use the full Type3D demo as seen in screenshots and the WebGL build. If you want to just get right to using Type3D for your own project you can skip this section.

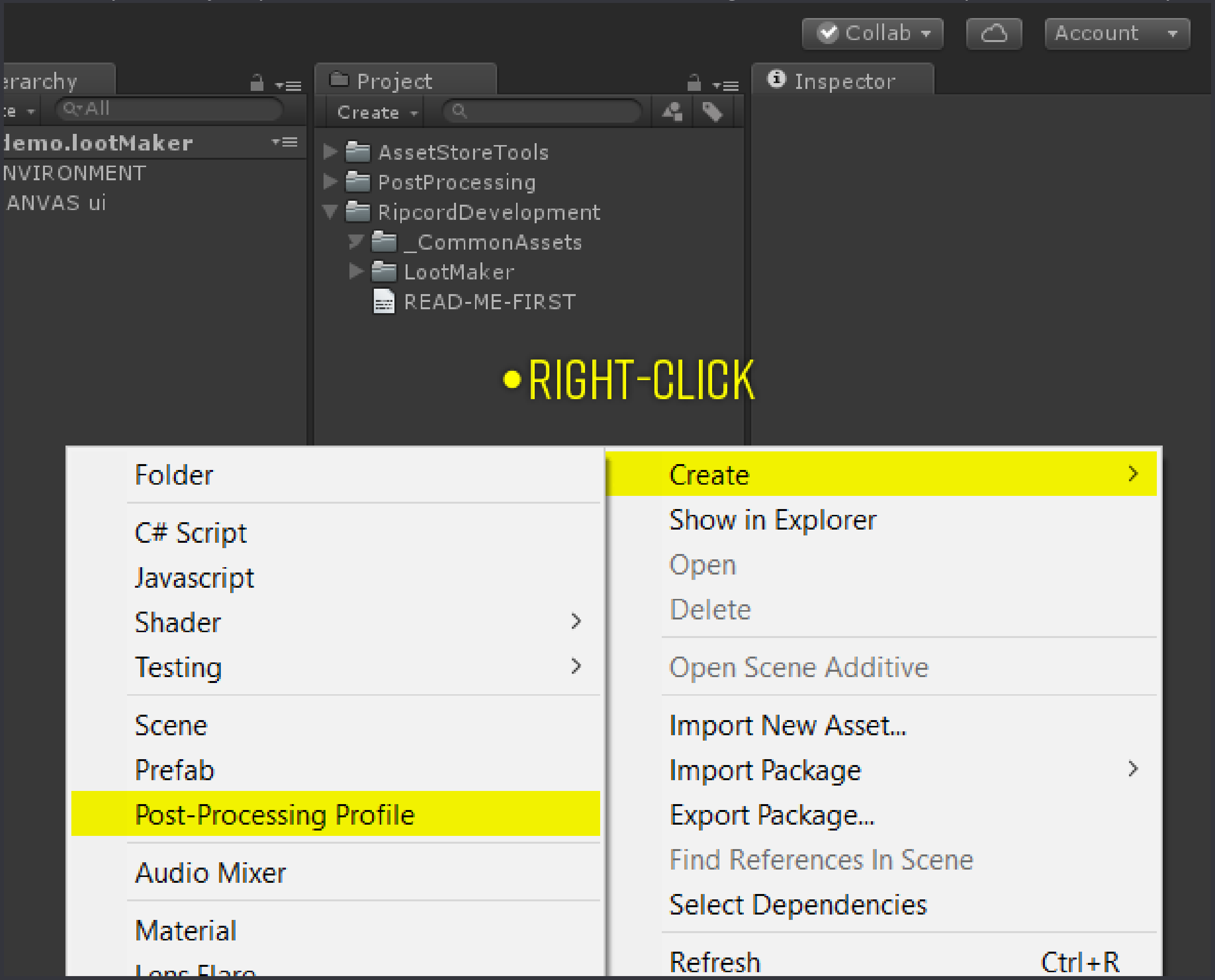
The demo scenes in this package make use of a free asset package provided by Unity. To replicate the exact look and functionality of the demo scenes, you'll need to import this package.

Download this package from the Unity Asset Store and import it into your project - Post Processing Stack

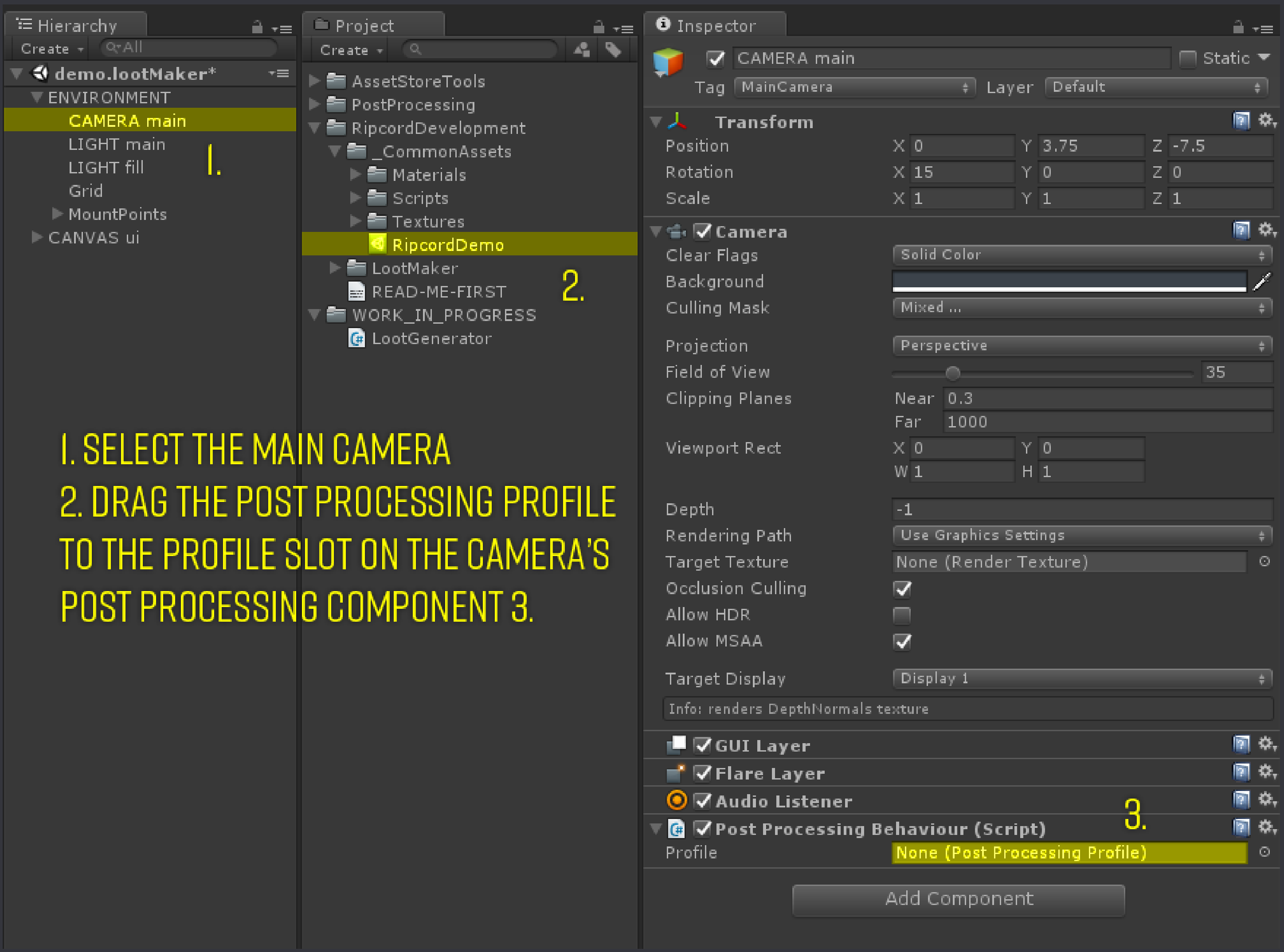
Select the camera in your scene and add the Post-Processing Behaviour component:



Right-click in your Project panel and select Create > Post Processing Profile. Name the profile whatever you like:



With your Main Camera selected, click and drag the new Post-Processing profile to the Profile slot on the Post Processing Behaviour component on the camera.



1. SELECT THE MAIN CAMERA
2. DRAG THE POST PROCESSING PROFILE TO THE PROFILE SLOT ON THE CAMERA'S POST PROCESSING COMPONENT 3.

☑ CAMERA main □ Static

Tag MainCamera Layer Default

Transform

Position X 0 Y 3.75 Z -7.5

Rotation X 15 Y 0 Z 0

Scale X 1 Y 1 Z 1

☑ **Camera**

Clear Flags: Solid Color

Background

Culling Mask: Mixed ...

Projection: Perspective

Field of View: 35

Clipping Planes: Near 0.3, Far 1000

Viewport Rect: X 0 Y 0 W 1 H 1

Depth: -1

Rendering Path: Use Graphics Settings

Target Texture: None (Render Texture)

Occlusion Culling: ☑

Allow HDR: □

Allow MSAA: ☑

Target Display: Display 1

Info: renders DepthNormals texture

☑ **GUI Layer**

☑ **Flare Layer**

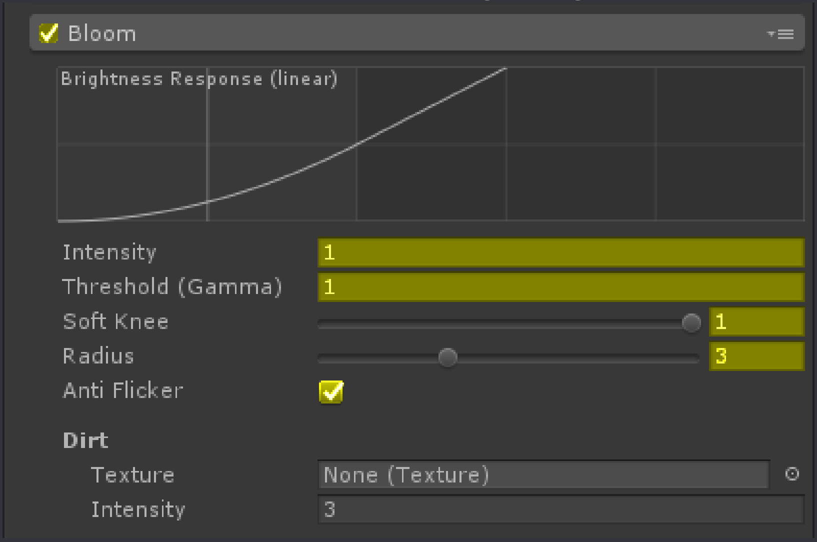
☑ **Audio Listener**

☑ **Post Processing Behaviour (Script)**

Profile: None (Post Processing Profile)

Select the Post Processing Profile in the Project panel and apply the following settings:

Bloom Post Processing settings



☑ Bloom

Intensity: 1

Threshold (Gamma): 1

Soft Knee: 1

Radius: 3

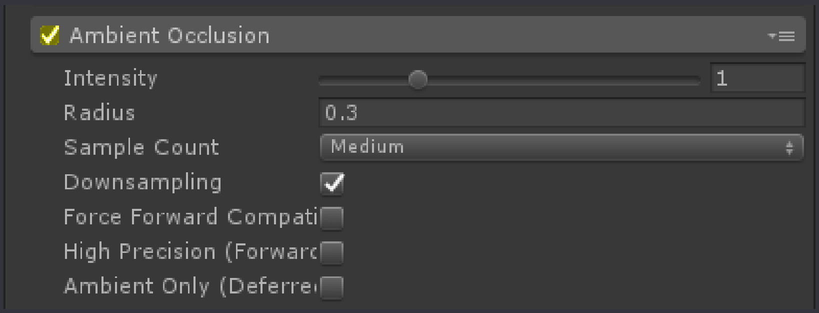
Anti Flicker: ☑

Dirt

Texture: None (Texture)

Intensity: 3

Ambient Occlusion Post Processing settings



☑ Ambient Occlusion

Intensity 1

Radius 0.3

Sample Count Medium

Downsampling ☑

Force Forward Compati □

High Precision (Forwarc □

Ambient Only (Deferre □

WRAP UP

Type3D provides the tools to easily generate your own dynamic 3D font objects using the characters supplied. It also provides a framework for developing and implementing your own 3D font.

The code has been heavily commented and modularized to make this package as easy to understand as possible. If you have any questions or comments, please don't hesitate to reach out.

If you find this package useful, please don't forget to leave positive feedback on the Unity Asset Store**. If you have any issues, please contact me with as much information about the issue as you can and I will get back to you as soon as possible.**

Thank you!

RIPCORD //
DEVELOPMENT

www.ripcorddev.com