

# Monotype AR/VR Text Plugin for Unity

## Version 2.0 - Quick Start Guide



## Notices

Copyright © 2018 Monotype Imaging Inc.

The software described in this document is furnished under license agreement from Monotype Imaging Inc. (“Monotype”) and may be used and copied only in accordance with the terms of such. This Quick Start Guide, the software described herein, and/or any copies thereof may not be provided or otherwise made available to any other person except as explicitly permitted by Monotype in writing.

This Quick Start Guide is provided “AS IS” and Monotype makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties or merchantability and fitness for a particular purpose. Monotype shall not be liable for errors contained herein for incidental or consequential damages in connection with the furnishings, performance, or use of this material. This document contains proprietary information protected by copyright. All rights are reserved.

Monotype Imaging Inc.  
600 Unicorn Park  
Drive Woburn, MA  
01801 phone  
781.970.6161 fax  
781.970.6001  
[monotype.com](http://monotype.com)

## Support

Contact your  
Monotype Technical Support Engineer  
or email: [AR/VR.Support@monotype.com](mailto:AR/VR.Support@monotype.com)

---

© 2018 Monotype Imaging Inc. All rights reserved. No part of this document may be photocopied, reproduced, or translated to another language without prior written consent from Monotype Imaging Inc.

Monotype™ is trademark of Monotype Imaging Inc. registered in the United States Patent and Trademark Office and may be registered in certain jurisdictions. TrueType is a registered trademark of Apple Computer Inc. in the U.S. and/or and other countries. Microsoft, Windows and OpenType are registered trademarks of Microsoft Corp. in the U.S. and/or other countries. Unicode and the Unicode logo are trademarks of Unicode Inc. and are reproduced with permission. Unity™ is a registered trademark of Unity IPR ApS in the U.S. and/or other countries. All other trademarks are the property of their respective owners.

## About This Document

Monotype AR/VR Text Plugin for Unity  
Quick Start Guide  
2.0(doc rev 1)  
August 21, 2018

## Contents

About This Document .....	2
Contents .....	3
Welcome to Monotype AR/VR Text Plugin .....	4
Getting Started .....	4
Prerequisites .....	4
Supported Platforms .....	5
Installing the Plugin.....	5
Updating the Plugin.....	5
Using the Plugin.....	6
Text Components .....	6
Text Component - Controls.....	7
Using Fonts .....	10
Smart Atlas Management.....	11
Rich Text Support.....	12
Transparency Support.....	17

## Welcome to Monotype AR/VR Text Plugin

The Monotype AR/VR Text Plugin is a plugin for rendering text in Unity environments that

- Renders text that is high quality, legible, and aesthetic
- Uses adaptive distance fields to render high quality text during run time
- Has native support for global languages - The shaping of complex languages (ex. Indic, Thai), large character sets, and any language layout requirements (ex. Arabic)
- Enables dynamic tuning, and customization of text
- Is resource (footprint, memory, battery consumption) optimized



This is beautiful!  
这很漂亮！

This guide introduces the developer to the features of Monotype AR/VR Text Plugin, how to install it and how to get the most out of it in terms of performance and text quality. It also points how to get more detailed information available in other documents and web sites and from technical support.

*TIP - For product brochures and more information on fonts and text layout solutions go to: <http://www.monotype.com/products-and-services>.*

## Getting Started

### Prerequisites

The plugin package - the plugin is packaged as a single. unityPackage file.

This can be downloaded from Monotype's [Developer Portal](#) using the credentials you received, or from the Unity Asset store.

# Monotype

## Supported Platforms

- Unity Development Platform: -
  - Windows 10 (32/64 bit)
  - Windows 7 (32/64 bit)
  - MAC (64 bit)
- Target Platform: -
  - Android
  - iOS
  - Windows
  - MAC

You will also need Unity editor installed in your system. The supported versions of Unity editor are –

- Unity 5.6.x
- Unity 2017.1.x
- Unity 2018.1.x

## Installing the Plugin

To install the plugin:

1. Open your Unity project in the Unity editor.
2. Double click the plugin file and choose your project from the Unity project selector window.
3. Alternatively, select the plugin from the Assets>Import Package>Custom Package menu of your Unity project and import the plugin.

## Updating the Plugin

To upgrade the plugin that's in your project:

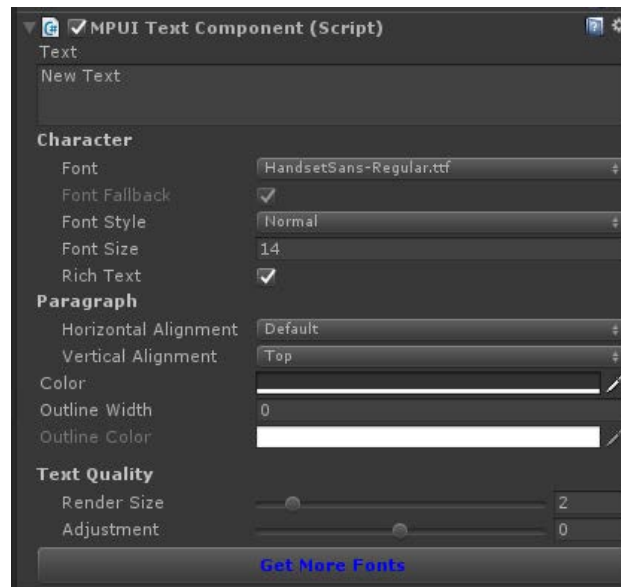
1. Save your project.
2. Create a new empty scene.
3. Close the project by closing the Unity editor.
4. Open the project once again. It should open the blank scene.
5. Import the plugin in your project.
6. Open your scene that you saved earlier.

## Using the Plugin

### Text Components

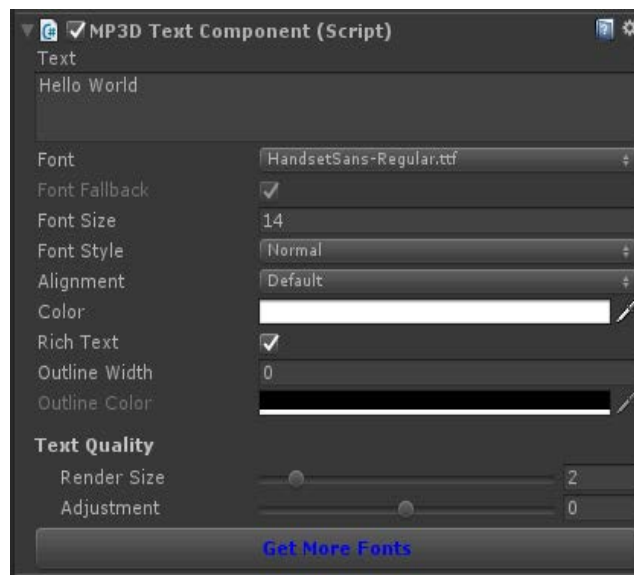
The Monotype Text Plugin provides three kinds of GameObjects to display texts in scenes.

1. Monotype-Text: Available under UI in GameObject menu.



Monotype-Text

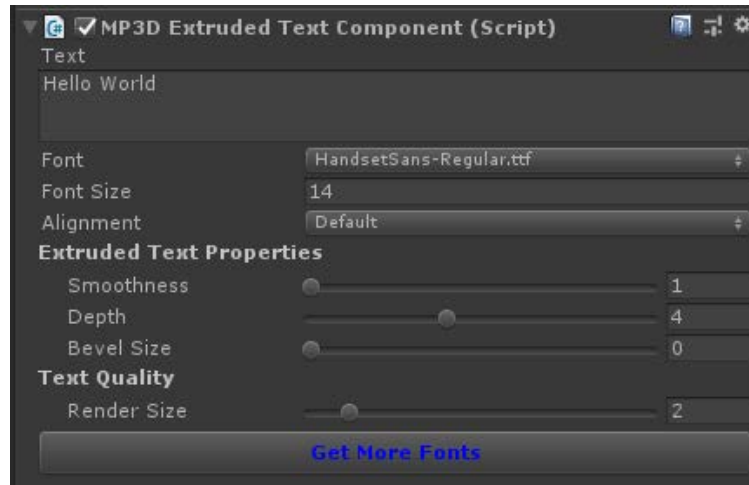
2. Monotype-3D Text: Available under 3D Object in GameObject menu.



Monotype-3D Text

# Monotype

3. Monotype-3D Extruded Text: Available under 3D Object in GameObject menu.



Monotype-3d Extruded Text

## Text Component - Controls

The plugin supports a powerful and extensive set of controls to help you configure and customize the text in your scenes.

### Character:

1. **Text:** The text string to display in the scene.
2. **Font:** This controls in which the text string is shown.  
This control will list all font files present under StreamingAssets/Monotype/Fonts path of the selected Unity project. Text will be displayed using the font selected from this list. Refer to the [Using Fonts](#) section for more details.
3. **Font Size:** The size at which text will be displayed in the scene.
4. **Font Style:** The font style (Normal/Bold/Italic) in which text will be displayed.

### Paragraph:

5. **Horizontal Alignment:** The horizontal alignment of the text.
6. **Vertical Alignment:** The vertical alignment of the text. This is not supported for 3D text.
7. **Color:** The color in which the text will be displayed.
8. **Rich Text Support:** Turn on/off Rich Text feature. Rich Text feature allows you to apply different attributes to different parts of a text object. See Rich Text Support section below for details.
9. **Outline Width:** The width of the outline applied to the text.
10. **Outline Color:** The color of the outline applied to the text. This control is disabled when the Outline Width is set to zero.

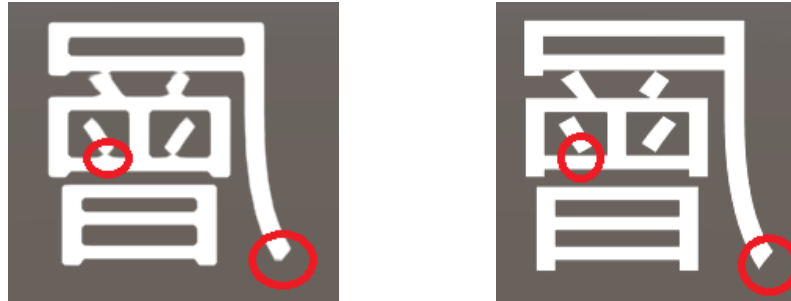
# Monotype

**Text Quality:**

11. **Render Size:** The relative size at which glyphs will be rendered internally.

Increasing this value will increase the quality of the text, make it sharper, crisper, but fill the internal memory faster.

Following is text at *Render Size* 2 and 10 respectively. You can see the difference in the sharpness of the edges.



Monotype's plugin renders text using distance fields – this adjusts the rendering size of the distance field. Higher numbers will result in increased text quality as the glyphs will be rendered by the engine at a larger size and consume more space in the text atlas. Version 1.0 of the Monotype Text Plugin provides Smart Atlas Management which means you will never run out of buffer space. Refer to the [Smart Atlas Management](#) section to learn more.

**Note:** While the *Font Size* control scales the text (changes the visible text size) without affecting the quality, *Render Size* impacts the text quality without affecting the display size.

12. **Adjustment:** This will adjust the thickness of the glyphs, without changing the spacing between glyphs.

Monotype's plugin uses adaptive distance fields to render text, so you can change the look, and feel of the text on the fly. This control can essentially create different weights of a font from thin to bold as shown below without having to package multiple weights! You should adjust this control after updating the *Render Size* for the best text quality.







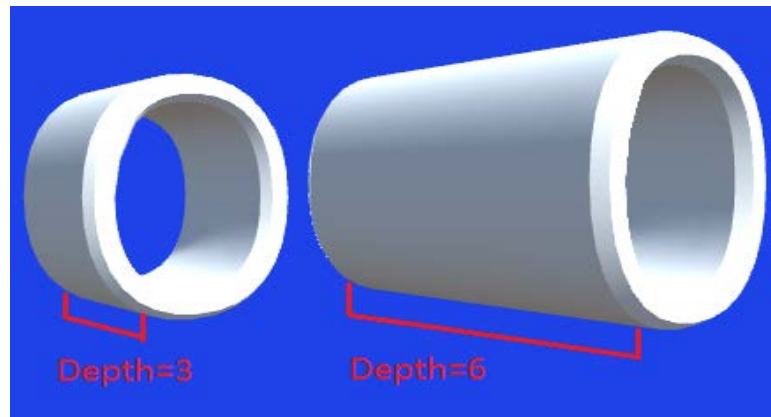
13. **Get More Fonts.** This will take you to a repository of 475+ fonts (covering 100+ languages) that are optimal for AR/VR. From there user can choose fonts as per their choice from a range of fonts optimized for AR/VR platforms. Please refer to the [Using Fonts](#) section for more details on how to use fonts.

#### **Extruded Text Properties (Supported by Monotype-3D Extruded Text Component):**

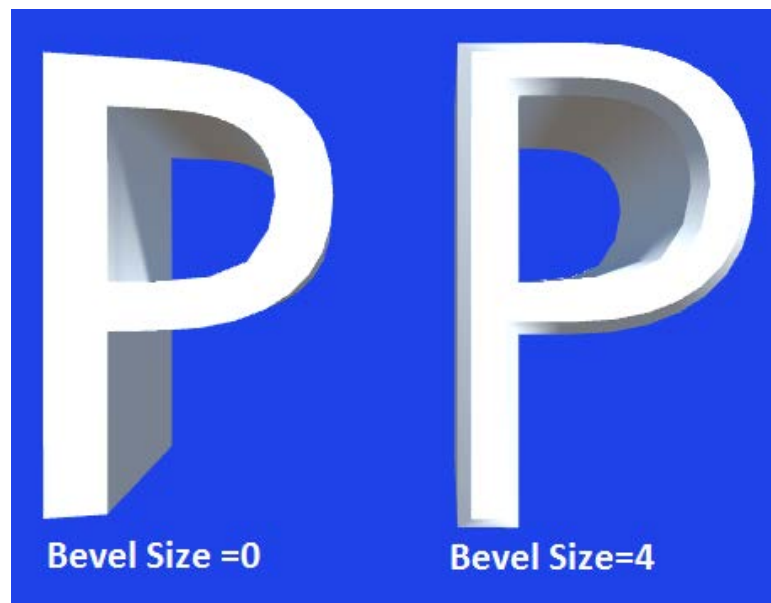
14. **Smoothness:** This feature helps in smoothing of edges of drawn text to improve their appearance and readability. Monotype Text Plugin gives user an option to adjust Monotype-3D Extruded text smoothness for enhanced readability and better appearance in your Scene. Below is the text rendered with Smoothness value set at 1 and 7. As the smoothness value increases, edges tends to get smooth and text appears to be of high quality.



15. **Depth:** Monotype-3D Extruded text supports Depth effect so that user can adjust extruded text depth effect and make text appear more interactive in real life applications. Depth effect can be set by sliding through values 0 – 10.



16. **Bevel Size:** Bevel is the control over the edges for an extruded object. Monotype Text Plugin provides *Bevel Size* option to control extruded text edges. Monotype Text Plugin supports Bevel Size starting from 0 till maximum value of 4.



17. **Render Size:** The relative size at which glyphs will be rendered internally. Increasing this value will increase the quality of the text, make it sharper, crisper, but fill the internal memory faster.

## Using Fonts

The Monotype Text Plugin supports the following font formats –

1. ttf
2. otf
3. ttc

# Monotype

**Using Fonts from Unity project:**

- To use fonts, you need to include the font file in the StreamingAssets/Monotype/Fonts folder of the Unity project.
- Upon adding the font file, you will be notified about the font licensing for the particular font.
- Once user accepts the license, you will be able to use the font from the font selection list in his Unity project.
- You can use the font for any Monotype text element used in the project. The font files present under StreamingAssets/Monotype/Fonts will be packed inside the application built from the project.

**Using Font Fallback:** Font fallback provides a mechanism to fall back on a list of fonts if a glyph isn't available in the selected font.

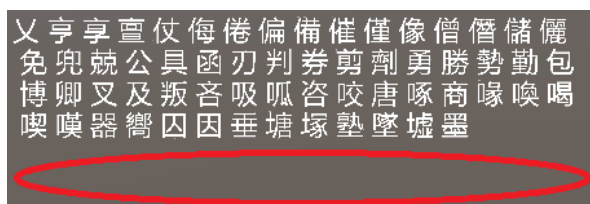
You can specify fallback fonts in the Unity project. If a glyph isn't found in the selected font (from the font selection control in the editor), it will fall back to other font files present in StreamingAssets/Monotype/Fonts or fonts specified in the fallback list. You can configure the fallback font list by specifying absolute path of the font files in the following file –

- StreamingAssets/Monotype/WindowsFontFallback.txt – For Windows system
- StreamingAssets/Monotype/OSXFontFallback.txt – For MAC system
- StreamingAssets/Monotype/AndroidFontFallback.txt – For Android system
- StreamingAssets/Monotype/IOSFontFallback.txt – For IOS system

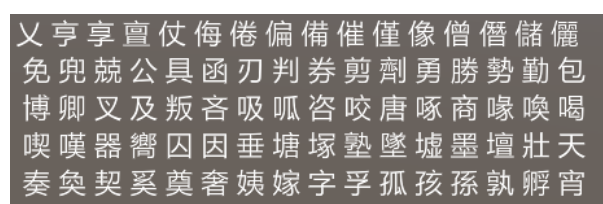
**Smart Atlas Management**

The Monotype Text Plugin comes with smart atlas management that ensures that you never experience missing text.

When the Atlas runs out of space, text will not be shown (see the figure on the left); Smart Atlas management ensures that you don't run out of space, so text will always be shown (right).



Unity Text



Monotype Text

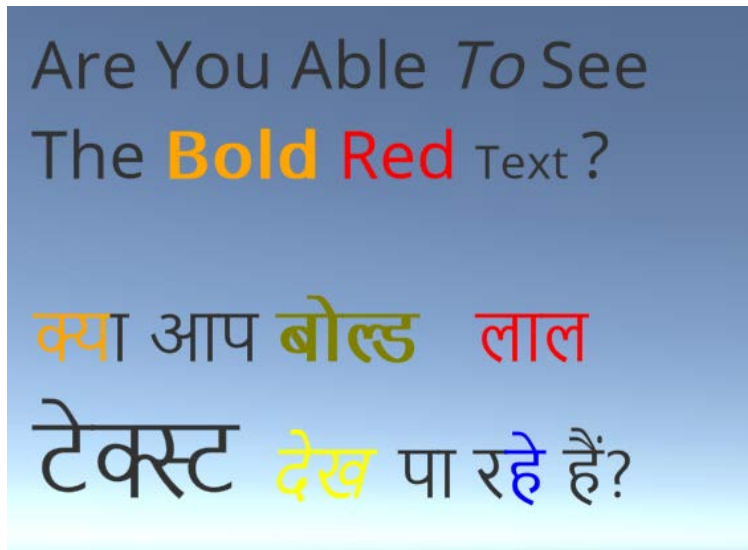
The plugin creates a Text Atlas for each font and *Render Size* combination. It also creates a new Text Atlas when an existing one becomes full thus ensuring that no character gets dropped.

# Monotype

**Note:** You don't need to do anything additional to enable the smart atlas management. This is all handled automatically, behind the scenes.

### Rich Text Support

The Rich Text feature allows you to apply different font attributes such as *Font Style* (bold/italic), *Font Size*, and *Font Color*, etc. to different parts of a Monotype Text object by providing markup tags in the text.



### Markup Elements

The markup system is similar to Unity's Rich Text support but isn't intended to be an exact replica. The basic idea is that a section of text can be enclosed inside a pair of matching tags. The tags are not displayed but indicate style changes to be applied to the text.

We are <b>very</b> excited!

As the example shows, the tags are just pieces of text inside the “angle bracket” characters, < and >. The text inside the tag denotes its name (which in this case is just b). Note that the tag at the end of the section has the same name as the one at the start but with the slash / character added. The b tag used in the example above applies bold to the word “very”, so the text will appear on screen as: -

We are **very** excited!

A marked up section of text (including the tags that enclose it) is referred to as an element.

### Nested Elements

## Monotype

It is possible to apply more than one style to a section of text by “nesting” one element inside another

We are `<b><i> definitely very</i></b>` excited!

The `i` tag applies italic style, so this would be presented onscreen as

We are ***definitely very*** excited!

Note the ordering of the ending tags, which is in reverse to that of the starting tags. The reason for this is perhaps clearer when you consider that the inner tags need not span the whole text of the outermost element

We are `<b>absolutely <i>definitely</i> very </b>` excited

which gives

We are **absolutely *definitely very*** excited

### Tag parameters

Some tags have a simple all-or-nothing effect on the text but others might allow for variations. For example, the color tag needs to know which color to apply. Information like this is added to tags by the use of parameters: -

We are `<color=green>green</color>` with envy

Note that the ending tag doesn't include the parameter value. Optionally, the value can be surrounded by quotation marks but this isn't required.

Note - In case any rich text element in a Monotype Text object is not well formed, no rich text will be applied on that text object.

### Supported Tags

- Tag - `b`

**Description** - Applies pseudo-emboldening to text.

# Monotype

**Example** - We are <b>very</b> excited!

- **Tag** - i

**Description** - Renders the text in italic.

**Example** - We are <i>very</i> excited!

- **Tag** - size

**Description** - Renders text in the mentioned size (in pixels). It accepts integer parameter as value of size. If no value is specified, the *Font Size* value set in the editor will be used.

**Example** - We are <size=50>largely</size> unaffected.

**Note** - When size tag is applied with parameter, the *Render Size* value set in the editor is ignored and the *Font Size* value set in the editor or mentioned in the size tag is used as *Render Size*. This may lead to degradation in the text quality at *Font Size* smaller than 40 (This may vary with font/language).

A simple workaround would be to increase all font sizes (set in editor and mentioned in size tags) by a factor such that the minimum *Font Size* in the Monotype Text object is 40 and then reduce the transform scale by the same factor.

- **Tag** - color

**Description** - Sets the color value according to the parameter value. If no parameter is specified, the color set in editor will be used.

The color can be specified in the traditional HTML format i.e.

#rrggbbaa







where the letters correspond to the pairs of hexadecimal digits denoting the red, green, blue and alpha(transparency) values for the color. For example, cyan with full opacity will be specified by #00ffffff.

Another option is to use the name of the color. This is easier to understand but naturally, the range of colors is limited and full opacity is always assumed.

For example, <color=cyan>

The available color names are given in the table below.

Color name	Hex value	Swatch
aqua (same as cyan)	#00ffffff	
Black	#000000ff	
Blue	#0000ffff	
Brown	#a52a2aff	
cyan (same as aqua)	#00ffffff	
Darkblue	#0000a0ff	
fuchsia (same as magenta)	#ff00ffff	
Green	#008000ff	
Grey	#808080ff	
Lightblue	#add8e6ff	
Lime	#00ff00ff	
magenta (same as fuchsia)	#ff00ffff	
Maroon	#800000ff	
Navy	#000080ff	
Olive	#808000ff	

Orange	#ffa500ff	
Purple	#800080ff	
Red	#ff0000ff	
Silver	#c0c0c0ff	
Teal	#008080ff	
White	#ffffffff	
Yellow	#ffff00ff	

**Example** - This is <color=#00ffffff>cyan</color> color.

- **Tag** - quad

**Description** - Renders an image in line with the text. It takes parameters that specify the material to use for the image, the image height in the pixels, and a further four that denote a rectangular area of the image to display. Unlike the other tags, quad does not surround a piece of text and so there is no ending tag - the slash character is placed at the end of the initial tag to indicate that it is “self-closing”.

Renderer’s Material List for Monotype 3D Text - Monotype 3D Text doesn’t support material fallback as supported by Unity 3D Text. This means when materials are added by user to Monotype 3D Text renderer’s material list, no effect will be seen on the text. These materials can only be used for quad tags. Monotype Text’s material management may lead to creation of multiple materials for a single text object at run-time. This may lead to change in the position of materials added by the user. Thus Monotype Text ensures that all the materials created internally (named as Monotype/DistanceField) are always on the top of renderer’s material list and are not modifiable by the user. This means that user can add custom materials at the bottom of the list.

## Monotype



Also the material index to be specified in quad tag needs to be the position of custom material in the renderer's material list (ignoring the internally created materials). For example, if the material list has following materials

Monotype/DistanceField
Monotype/DistanceField
Monotype/DistanceField
Default-Diffuse
Default-Skybox

then, the index of Default-Diffuse would be 0 and that of Default-Skybox would be 1. This also means that user won't be able to use internally created materials for quad tag.

#### Example -

```
<quad material=1 size=20 x=0.1 y=0.1 width=0.5 height=0.5 />
```

This selects the user specified material at position in the renderer's material array and sets the height of the image to 20 pixels. The rectangular area of image starts at given by the x, y, width and height values, which are all given as a fraction of the unscaled width and height of the texture.

#### Note -

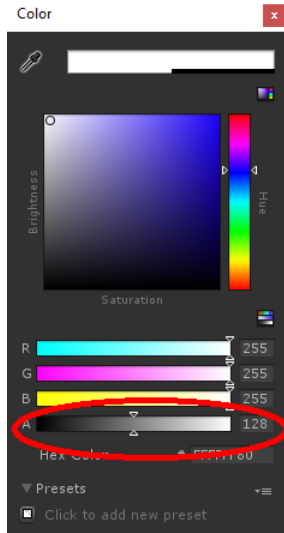
1. The minimum width of quad is fixed to 1 pixel. This may lead to distorted texture when the value of size parameter is less than the height parameter.
2. The negative values of parameters such as size, height and width may lead to undefined output.

**Note** - Monotype Text currently doesn't support material tag supported by Unity.

### Transparency Support

You can make Monotype Text object transparent by changing alpha value in color selector dialog that opens up on clicking color property in inspector.

# Monotype



See following Monotype text with alpha (A) value 255 and 128

Hello World

Hello World

You can also make text transparent by providing proper alpha value in the Rich Text color tag. Using `<color=#ff0000ff>` will give you 100% opaque red color while `<color=#ff000080>` will give you 50% transparency same as alpha value 128 as shown above.