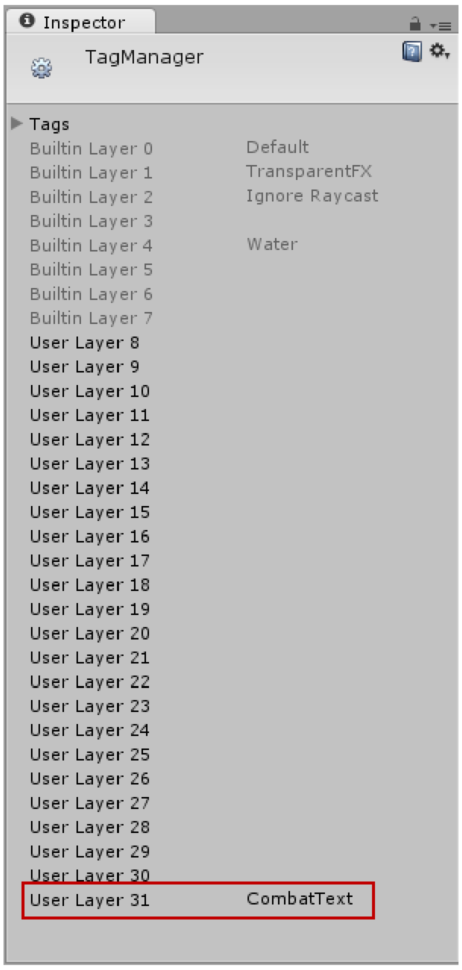
Combat Text Manual

Note: The names BattleText and CombatText are used interchangeably in the text, this is due to the original name being BattleText, but it was changed to CombatText before release.

Initial Setup

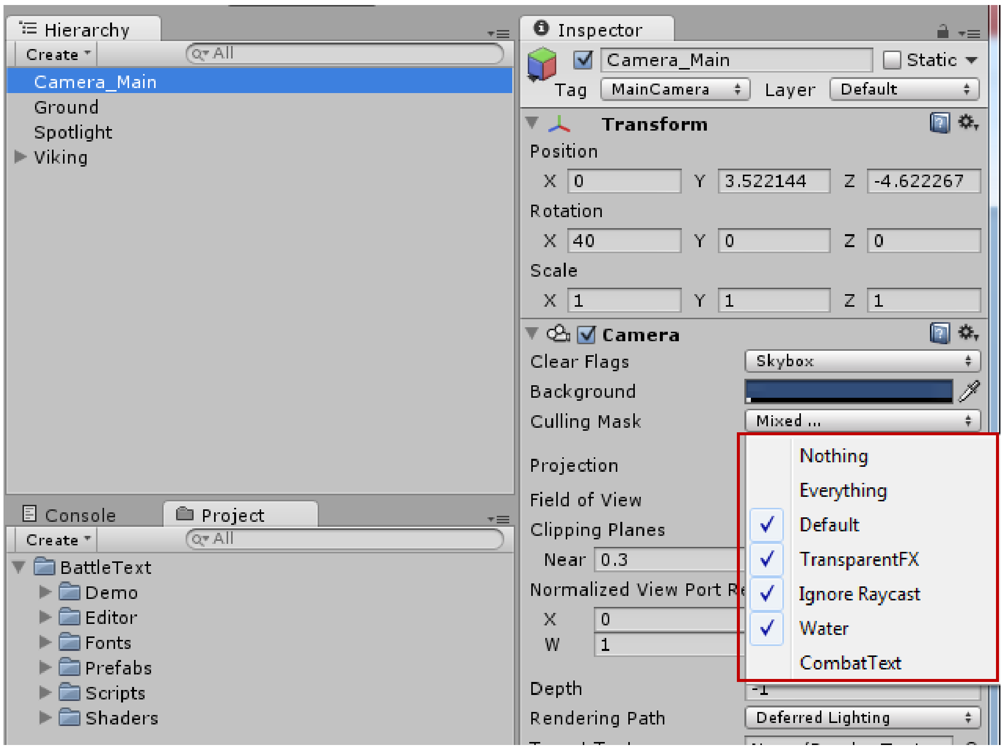
The first thing we're going to do is to create a piece of static text, more specifically a nameplate. Open the scene file BattleText/Demo/Scenes/Manual. This scene contains a basic setup with a floor, spotlight, camera and Viking model.

Next we want to setup a specific Combat Text layer, go into the Tag/Layer manager and add it:



Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
Builtin Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	
Builtin Layer 6	
Builtin Layer 7	
User Layer 8	
User Layer 9	
User Layer 10	
User Layer 11	
User Layer 12	
User Layer 13	
User Layer 14	
User Layer 15	
User Layer 16	
User Layer 17	
User Layer 18	
User Layer 19	
User Layer 20	
User Layer 21	
User Layer 22	
User Layer 23	
User Layer 24	
User Layer 25	
User Layer 26	
User Layer 27	
User Layer 28	
User Layer 29	
User Layer 30	
User Layer 31	CombatText

After creating the layer, find the Camera_Main object in the scene and set its culling mask to everything except the CombatText layer.



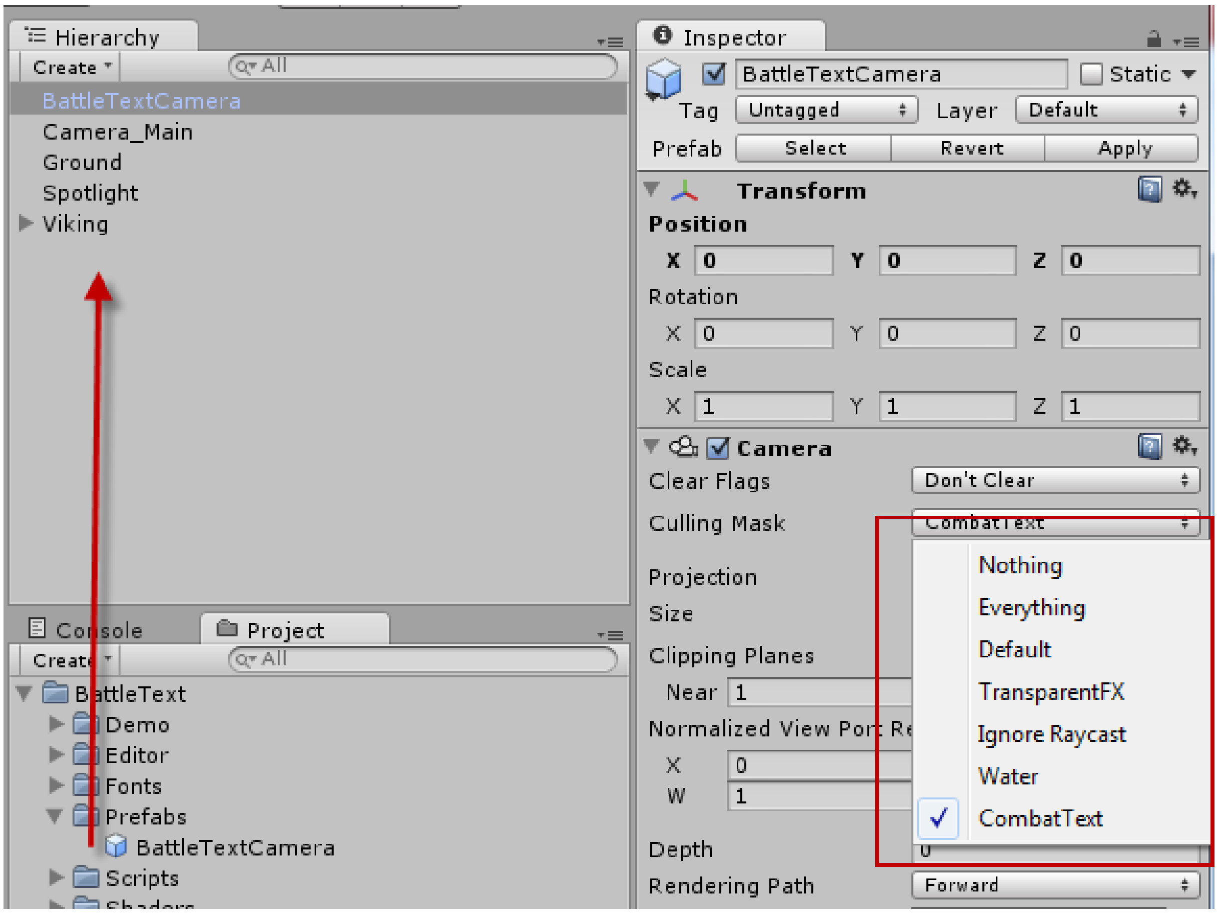
Camera_Main
Tag MainCamera Layer Default

Transform
Position X 0 Y 3.522144 Z -4.622267
Rotation X 40 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags Skybox
Background
Culling Mask Mixed ...
Projection
Field of View
Clipping Planes
Near 0.3
Normalized View Port Re
X 0
W 1
Depth -1
Rendering Path Deferred Lighting

Culling Mask menu:
Nothing
Everything
✓ Default
✓ TransparentFX
✓ Ignore Raycast
✓ Water
CombatText

After setting up the layers we're going to create a battle-text camera, find the prefab in BattleText/Prefabs called BattleTextCamera and drag it into the scene. Make sure that the culling mask of the camera is set to only render the CombatText layer.



BattleTextCamera
Tag Untagged Layer Default
Prefab

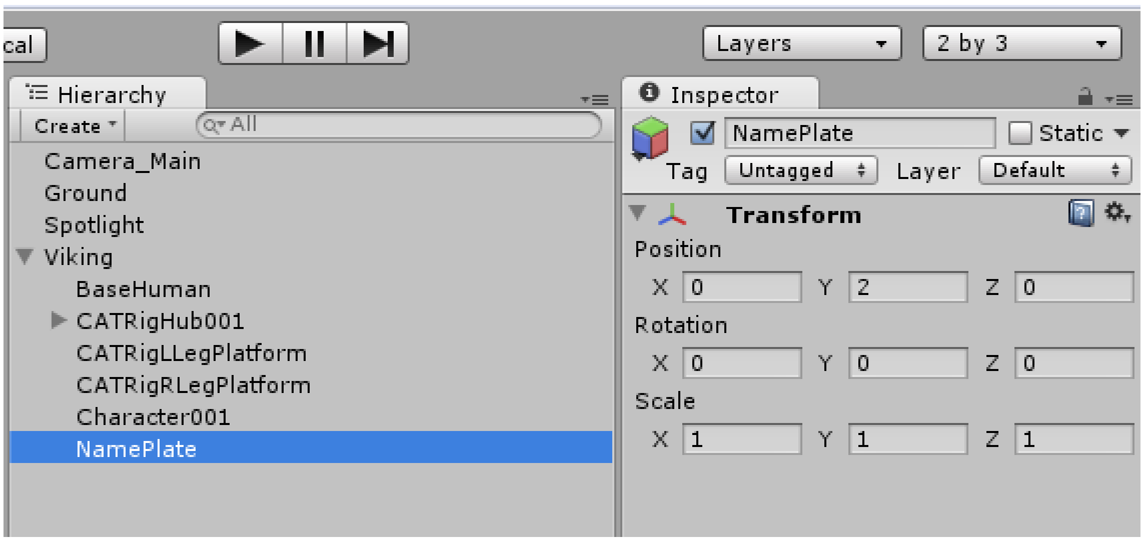
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags Don't Clear
Culling Mask CombatText
Projection
Size
Clipping Planes
Near 1
Normalized View Port Re
X 0
W 1
Depth 0
Rendering Path Forward

Culling Mask menu:
Nothing
Everything
Default
TransparentFX
Ignore Raycast
Water
✓ CombatText

Static Text

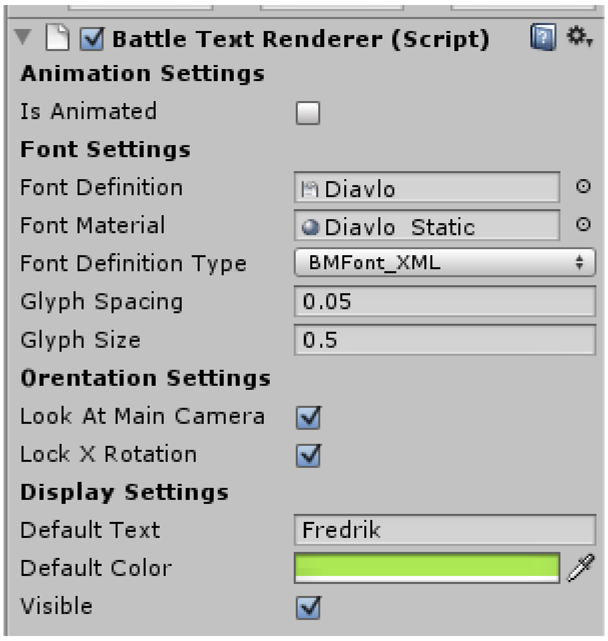
Now it's time to render some text. We're going to start with a simple static text or more specifically a name plate for the Viking. Create a new empty game object, call it "NamePlate" and add it as a child to the Viking object, make sure the position is set to (0, 2, 0):



Camera_Main
Ground
Spotlight
Viking
BaseHuman
CATRigHub001
CATRigLLegPlatform
CATRigRLegPlatform
Character001
NamePlate

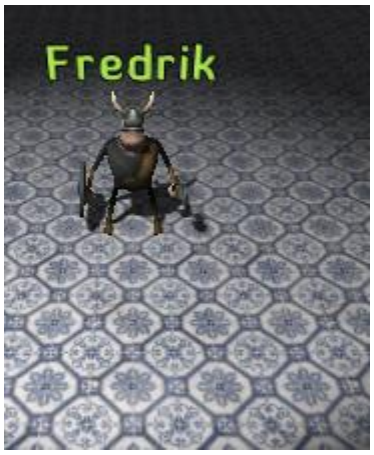
NamePlate
Tag Untagged Layer Default
Transform
Position X 0 Y 2 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Next, drag an instance of the BattleTextRenderer script from BattleText/Scripts to the NamePlate object. Un-check "Is Animated" and a couple of more options will open up. In the Font Definition drag the BattleText/Fonts/Diavlo text asset to it. In the Font Material drag the BattleText/Fonts/Diavlo_Static material. Make sure to check both "Look at Main Camera" and "Lock X Rotation". Under Display Settings you can set a text and default color, I chose my name "Fredrik" and a nice green color. You can also control the size and spacing of the glyphs under Font Settings.



Battle Text Renderer (Script)
Animation Settings
Is Animated ☐
Font Settings
Font Definition: Diavlo
Font Material: Diavlo Static
Font Definition Type: BMFont_XML
Glyph Spacing: 0.05
Glyph Size: 0.5
Orentation Settings
Look At Main Camera ✓
Lock X Rotation ✓
Display Settings
Default Text: Fredrik
Default Color: 🟩
Visible ✓

If you hit play now, the Viking will appear with a nameplate above its head.



If you need to set or change the text during runtime, you can grab the BattleTextRenderer script from the NamePlate object and call the SetText method on it, like this:

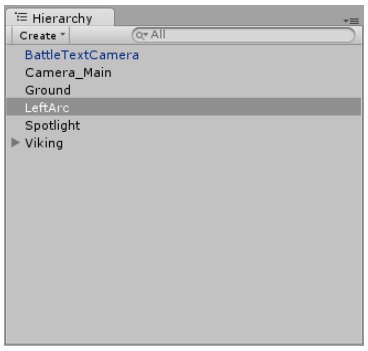
```
GetComponent<BattleTextRenderer>().SetText("New Text");
```

There is also an overload for SetText which lets you update the font definition and material if you need to change it during runtime. That's really all there is to creating static text plates.

Animated Text

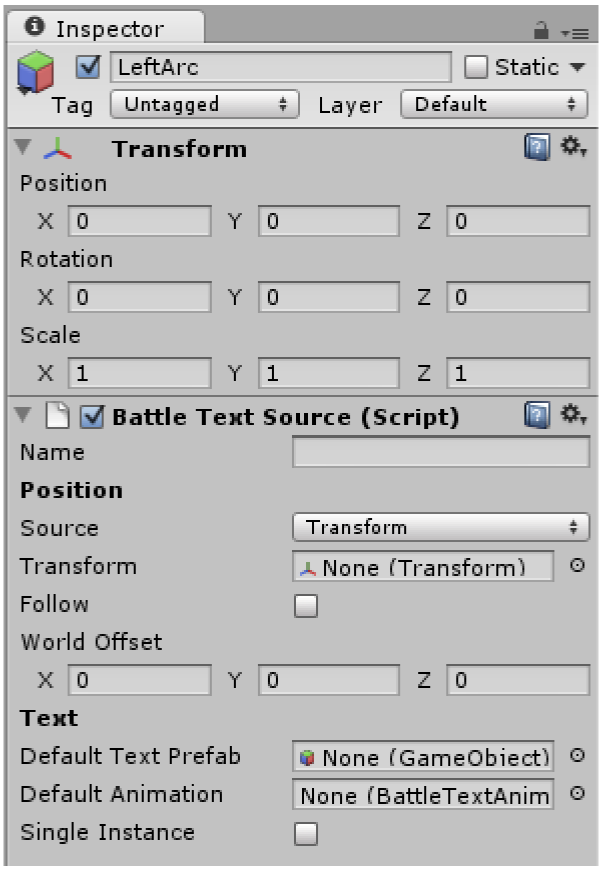
Let's get into the real meat of BattleText: Animated Text, this allows you to create all kinds of animations and effects on your text. There are two main ways of displaying animated text, you can either display it in screen or world mode. Screen mode positions the text on screen and uses the BattleTextCamera to render it at the same size with no scaling, etc. World mode actually puts the text in the world and renders it using the standard camera, giving it perspective relative to its position to the camera.

Let's start by creating one of the "scrolling combat text" on-screen arcs found in the webplayer demo. Start by creating an empty game object, and call it "LeftArc":



BattleTextCamera
Camera_Main
Ground
LeftArc
Spotlight
Viking

Now drag an instance of the BattleTextSource script from BattleText/Scripts to it:


Inspector

LeftArc — Static

Tag: Untagged — Layer: Default

Transform

Position: X 0, Y 0, Z 0

Rotation: X 0, Y 0, Z 0

Scale: X 1, Y 1, Z 1

Battle Text Source (Script)

Name:

Position

Source: Transform

Transform: None (Transform)

Follow: ☐

World Offset: X 0, Y 0, Z 0

Text

Default Text Prefab: None (GameObject)

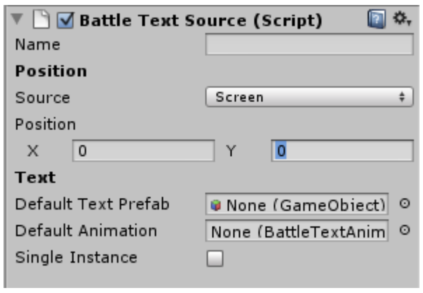
Default Animation: None (BattleTextAnim

Single Instance: ☐

Let’s step through the different settings available, first up is the Name – this is purely for organizational reasons and is not required. Since there will be cases where you need more then on Battle Text Source attached to a game object, you can give them a name and then retrieve them with an extension method called GetBattleTextSource which exists on both MonoBehaviour and GameObject instances, like this:

```
this.GetBattleTextSource("name of source");
```

The Position section of the settings allow you to customize the positioning of the text, there are two possible position sources: Transform and Screen. The most complex being Transform, it asks for a target transform to pull its position from. If you check “Follow” the text will keep following the transform around as it moves, if it’s not checked only the initial position will be taken from the Transform. You can also specify an offset that gets added to the position pulled from the Transform. However, as we’re creating an arc we want to be position on our screen, switch Source to “Screen”, and it will look like this:


Battle Text Source (Script)

Name:

Position

Source: Screen

Position: X 0, Y 0

Text

Default Text Prefab: None (GameObject)

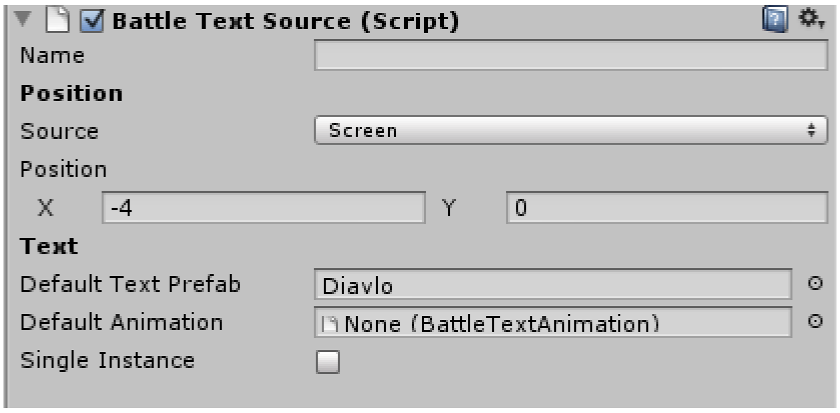
Default Animation: None (BattleTextAnim

Single Instance: ☐

Positioning using the Screen source is a bit easier, you can set two coordinates: X and Y, -10 being left/top and 10 being right/bottom.

Under the Text settings you can specify a default prefab and default animation to use if none is specified when calling the text sources DisplayText method. If you check Single Instance the text source will only allow one text to be displayed at any one time and will destroy the previous existing text if one already exists when you call DisplayText. This is used for the “Trigger Notice” text in the web player demo.

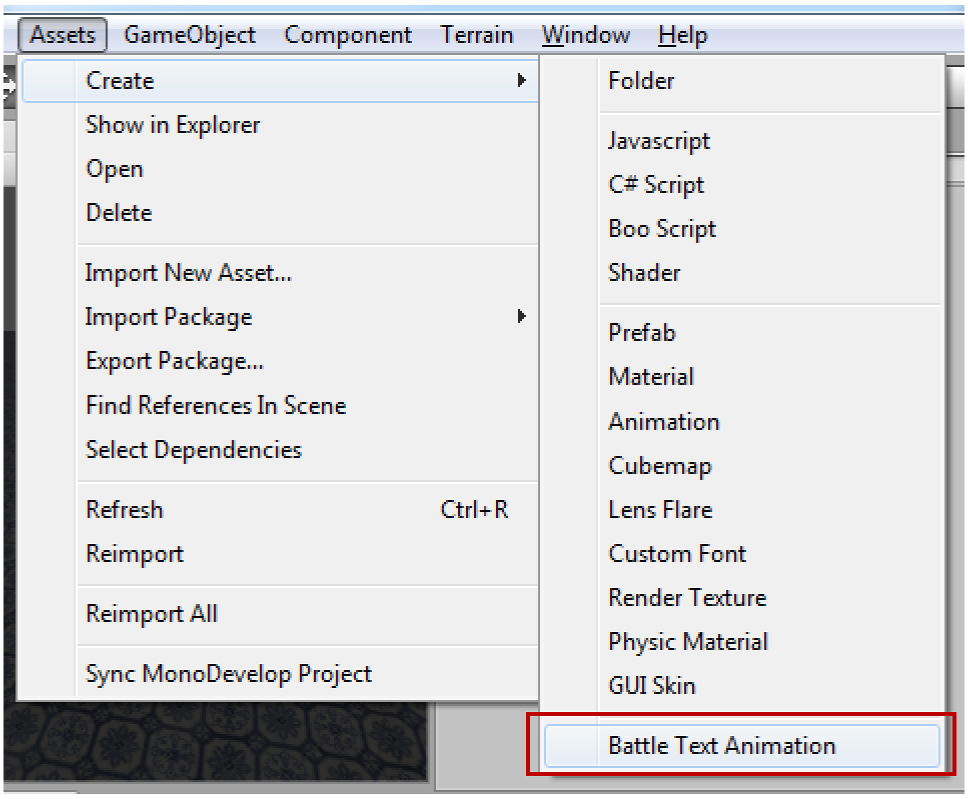
We want an X screen position of -4, and the Text Prefab to be set to the Diavlo prefab available in the BattleText/Fonts folder, like this:



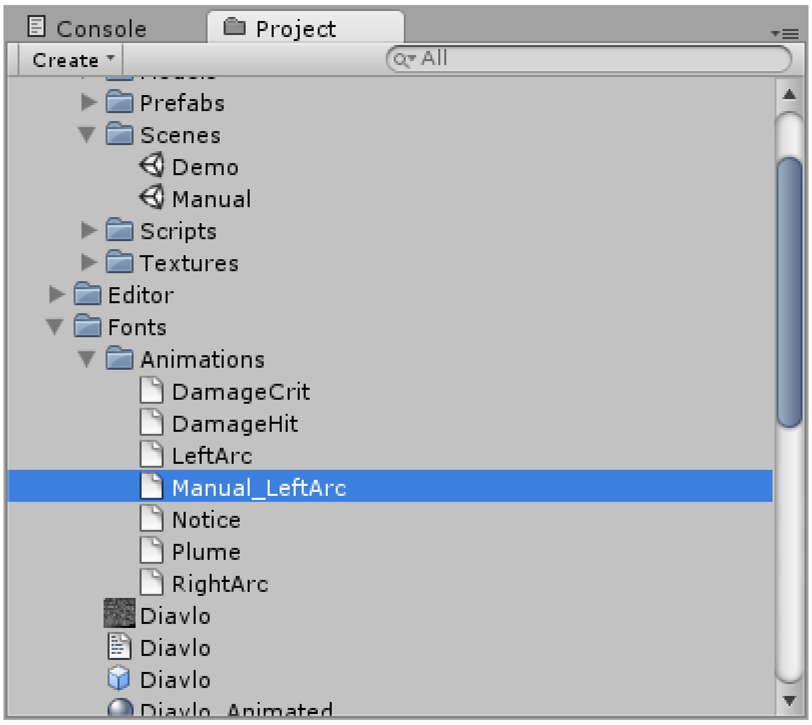
Battle Text Source (Script)

Name	
Position	
Source	Screen
Position	
X	-4
Y	0
Text	
Default Text Prefab	Diavlo
Default Animation	None (BattleTextAnimation)
Single Instance	☐

We can’t fill in the Default Animation yet, as we don’t have one, let’s create it right now. Go the Assets/Create/Battle Text Animation option:



This will create a new animation in the BattleText/Fonts/Animations folder, I'm going to call it Manual_LeftArc.



Now let's create our animation and go through our animations settings.

Animate In World
This will display the text in the worlds coordinate space instead of the screen space, the "Trigger Plume" effect in the webplayer demo is set to animate in world which shows it above the head of the Viking as he moves around. Don't enable this.

Time
The amount of time, in seconds, until the text with this animation will be destroyed. Set this to 4.

Fade Delay
The amount of time, in seconds, until the text will start fading out. The time it takes for the fade itself is specified on the font material. Set this to 0.75.

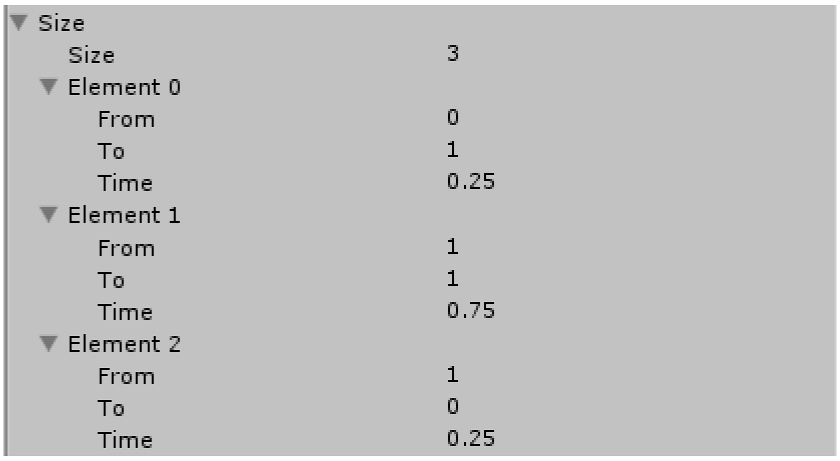
Default Color
The default color if none is specified when calling DisplayText on a BattleTextSource script. Pick any color you want.

Fixed Offset
A fixed offset that will be applied to the position the animation gets from the BattleTextSource. Leave this at (0, 0, 0)

Random Offset
A random offset that will be calculated for each instance of text using this animation. Set this to (1, 1, 1)

Size
Allows you to animate the size of the text over time, each element will be applied after the previous one expired. Create three elements in here, and give them these values:

1. From = 0, To = 1, Time = 0.25
2. From = 1, To = 1, Time = 1
3. From = 1, To = 0, Time = 0.25



```
▼ Size
    Size              3
  ▼ Element 0
      From            0
      To              1
      Time            0.25
  ▼ Element 1
      From            1
      To              1
      Time            0.75
  ▼ Element 2
      From            1
      To              0
      Time            0.25
```

This will pop the text from 0 to 100% size over 0.25 second, then display the text at 100% size for 0.75 seconds and then shrink it from 100% to 0 size over 0.25 seconds.

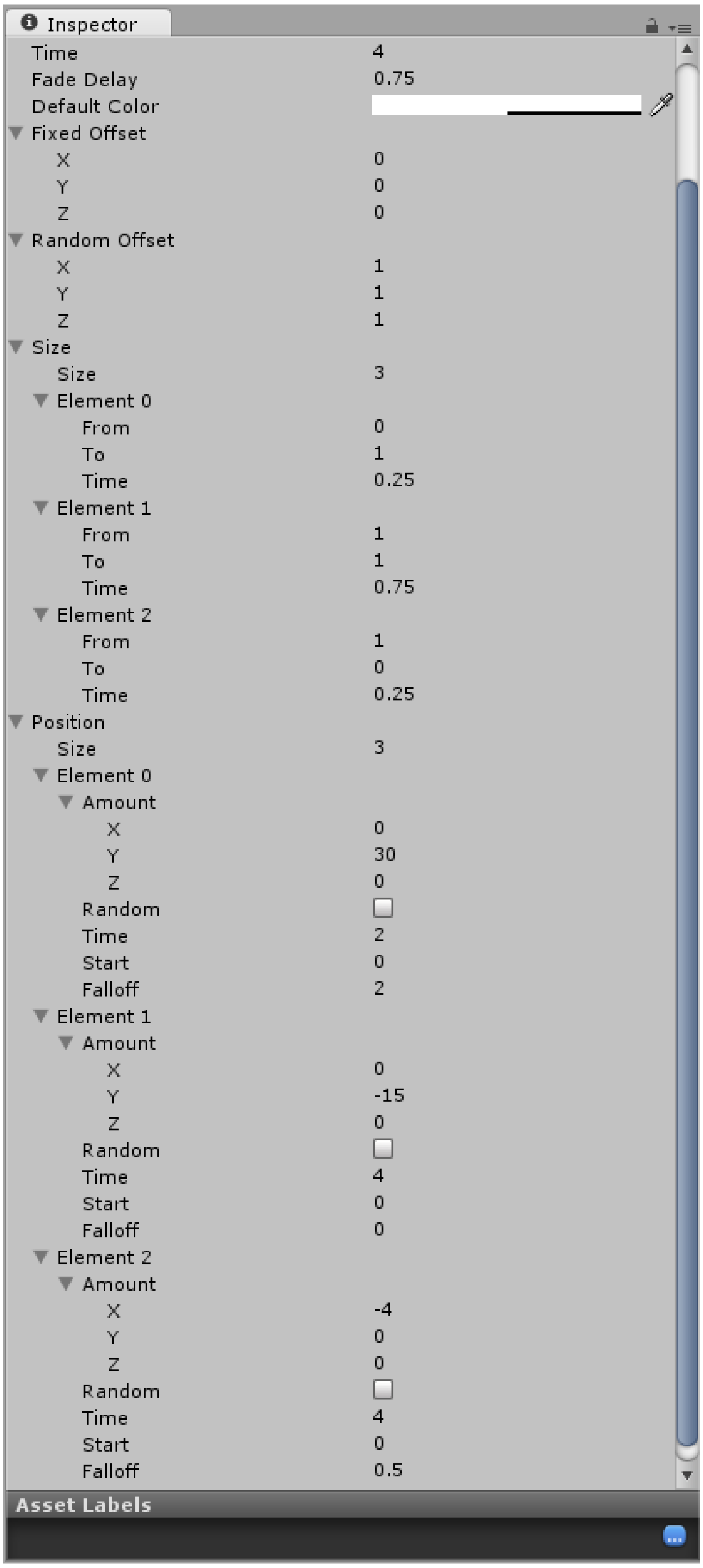
Position
Allows you to animate the position of a text over time, however while the Size animations are applied after one another – the position animations are applied all at the same time. Create three position animations here, and give them the following values:

1. Amount = (0, 30, 0), Time = 2, Start = 0, Falloff = 2
2. Amount = (0, -15, 0), Time = 4, Start = 0, Falloff = 0
3. Amount = (-4, 0, 0), Time = 4, Start = 0, Falloff = 0.5

Leave “Random” unchecked on all of them. Let me go into detail on what each setting does.

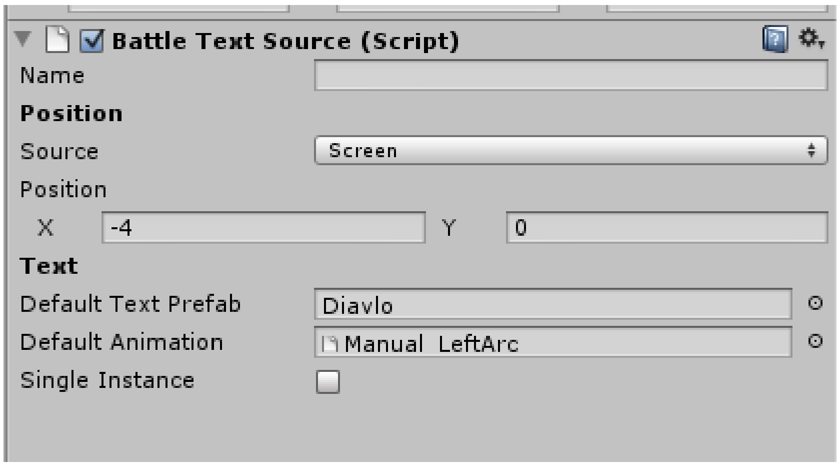
- Amount, The force and direction of the position change
- Random, Instead of applying amount directly a random value is picked that lies between –amount and amount
- Time, The duration of the position change, in seconds
- Start, Delay start of Time by this many seconds
- Falloff, After starting, reduce to 0 over this amount of time

When you’re done, all the settings for the animation should look like this:



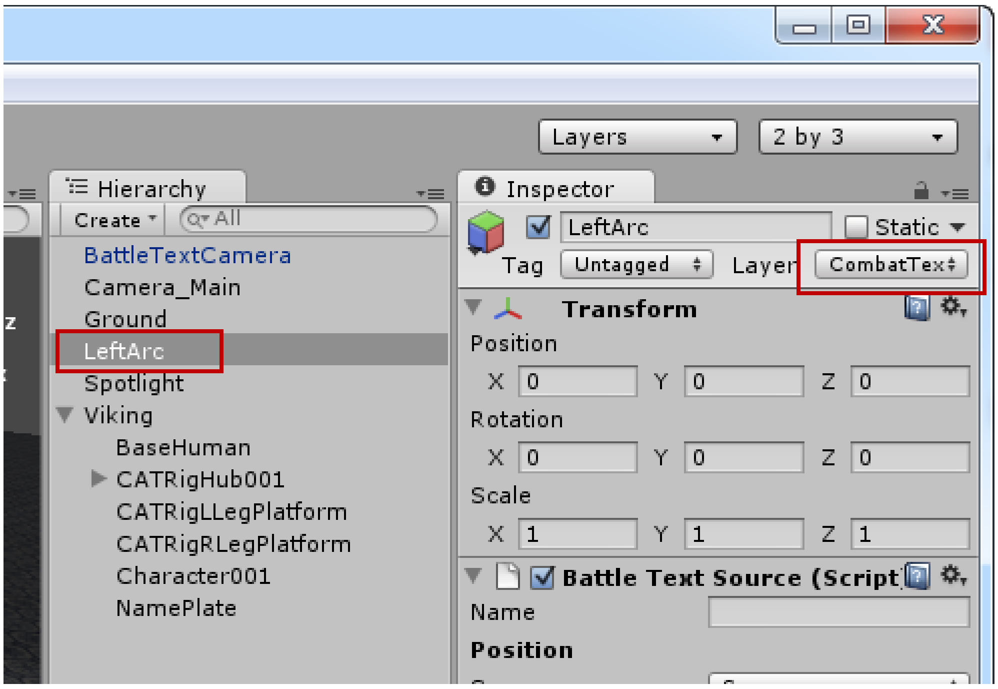
Time	4
Fade Delay	0.75
Default Color	
▼ Fixed Offset	
X	0
Y	0
Z	0
▼ Random Offset	
X	1
Y	1
Z	1
▼ Size	
Size	3
▼ Element 0	
From	0
To	1
Time	0.25
▼ Element 1	
From	1
To	1
Time	0.75
▼ Element 2	
From	1
To	0
Time	0.25
▼ Position	
Size	3
▼ Element 0	
▼ Amount	
X	0
Y	30
Z	0
Random	☐
Time	2
Start	0
Falloff	2
▼ Element 1	
▼ Amount	
X	0
Y	-15
Z	0
Random	☐
Time	4
Start	0
Falloff	0
▼ Element 2	
▼ Amount	
X	-4
Y	0
Z	0
Random	☐
Time	4
Start	0
Falloff	0.5

Now drag the Manual_LeftArc animation into the Default Animation slot on the Battle Text Source script attached to the LeftArc game object:



Battle Text Source (Script)
Name
Position
Source: Screen
Position
X -4 Y 0
Text
Default Text Prefab: Diavlo
Default Animation: Manual LeftArc
Single Instance: ☐

Last but not least, remember to set the layer for the LeftArc object to "CombatText".



Hierarchy
BattleTextCamera
Camera_Main
Ground
LeftArc
Spotlight
Viking
BaseHuman
CATRigHub001
CATRigLLegPlatform
CATRigRLegPlatform
Character001
NamePlate

Inspector
LeftArc — Static
Tag: Untagged — Layer: CombatText
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Battle Text Source (Script)
Name
Position

Now we're ready to use our text source, all we need to do is to create a tiny script that triggers the text for demonstration purposes. The script looks like this, and if you don't want to write it yourself you can find it in the BattleText/Demo/Scripts folder.

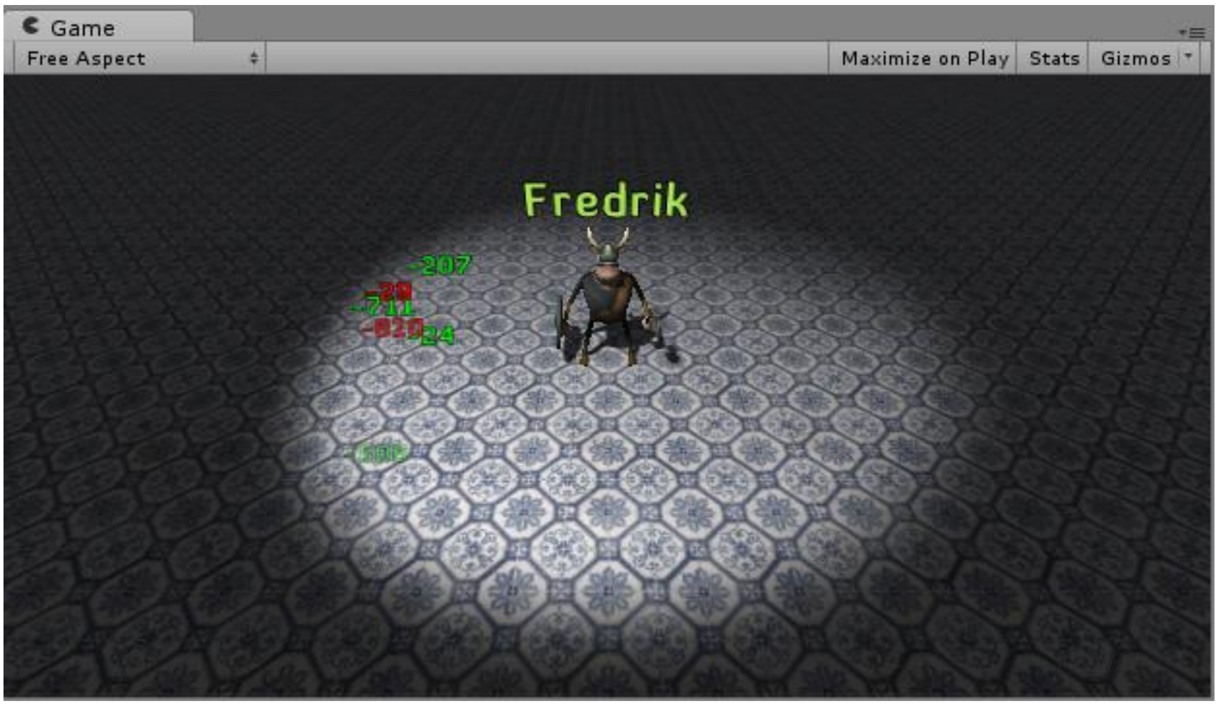
```
using UnityEngine;

public class TriggerArc : MonoBehaviour
{
    Color[] colors = new Color[] { Color.red, Color.green };
    BattleTextSource source;

    void Start()
    {
        source = GetComponent<BattleTextSource>();
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.T))
        {
            source.DisplayText("-" + Random.Range(1, 1000).ToString(),
colors[Random.Range(0, colors.Length)]);
        }
    }
}
```

If we attach the TriggerArc script to the LeftArc game object and hit play, we will be able to trigger the arc by pressing T several times, it will look something like this:



And that's really all there is to creating animated texts, if you want to see more examples on how to create different effects I urge you to check out the scripts and scene that can be found in the BattleText/Demo folder.

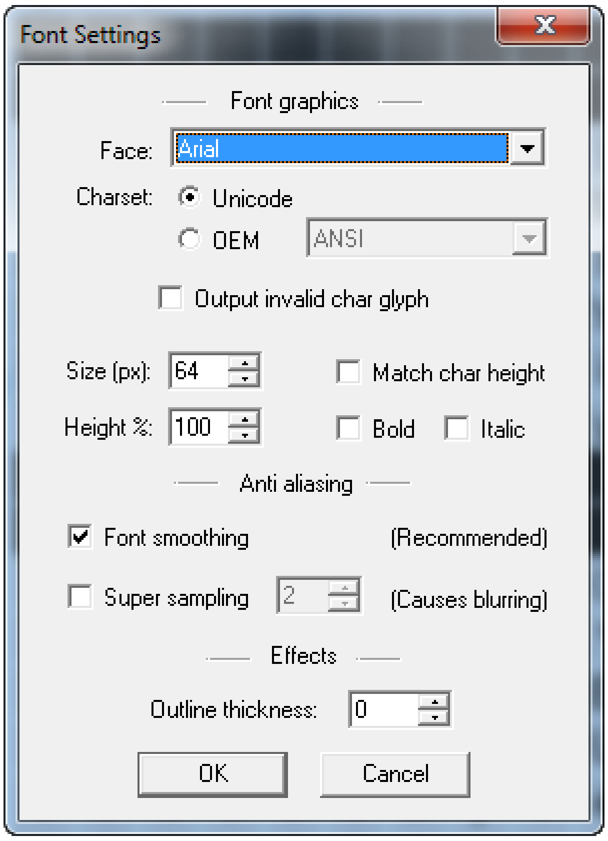
Importing your own font

Notice: Importing your own fonts currently requires you to use a program called BMFont which is only available for Windows currently. Future support for the OS X based Glyph Designer is planned.

To import your font for use with BattleText, follow these steps.

Step 1

Start BM Font and open font settings, select the font you want and the pixel size you want, 32 or 64 are common options depending on how many characters you need. For this tutorial I'm going to export Arial:



Font Settings

Font graphics

Face: Arial

Charset: [✓ Unicode] OEM ANSI

☐ Output invalid char glyph

Size (px): 64 ☐ Match char height

Height %: 100 ☐ Bold ☐ Italic

Anti aliasing

☑ Font smoothing (Recommended)

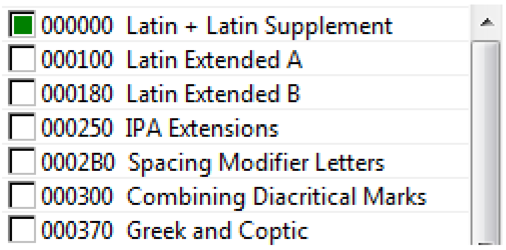
☐ Super sampling 2 (Causes blurring)

Effects

Outline thickness: 0

Step 2

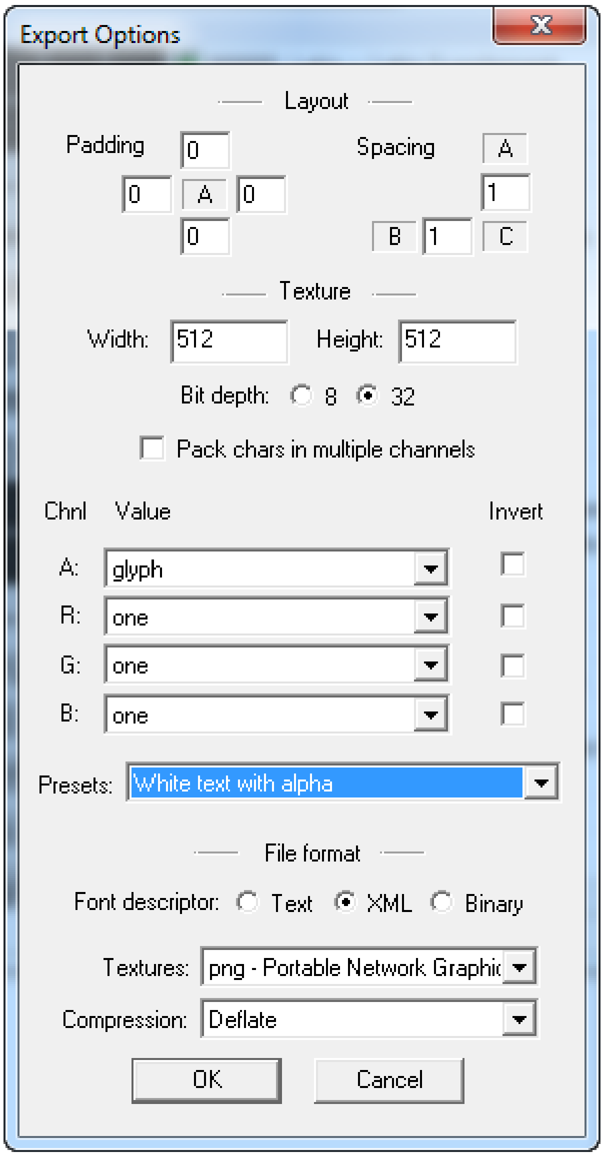
Check the glyphs you want in the main BM Font window:



🟩 000000 Latin + Latin Supplement
☐ 000100 Latin Extended A
☐ 000180 Latin Extended B
☐ 000250 IPA Extensions
☐ 0002B0 Spacing Modifier Letters
☐ 000300 Combining Diacritical Marks
☐ 000370 Greek and Coptic

Step 3

Open export options, from presets select “White text with alpha”. On font descriptor select XML, on textures select png. On compression select deflate. I usually use a texture size of 512x512 as it fits most western characters in 64x64 size.



Export Options

Layout

Padding: 0, 0, 0, 0

Spacing: A 1, B 1

Texture

Width: 512 Height: 512

Bit depth: 8 [✓ 32]

[ ] Pack chars in multiple channels

Chnl	Value	Invert
A:	glyph	[ ]
R:	one	[ ]
G:	one	[ ]
B:	one	[ ]

Presets: White text with alpha

File format

Font descriptor: Text [✓ XML] Binary

Textures: png - Portable Network Graphic

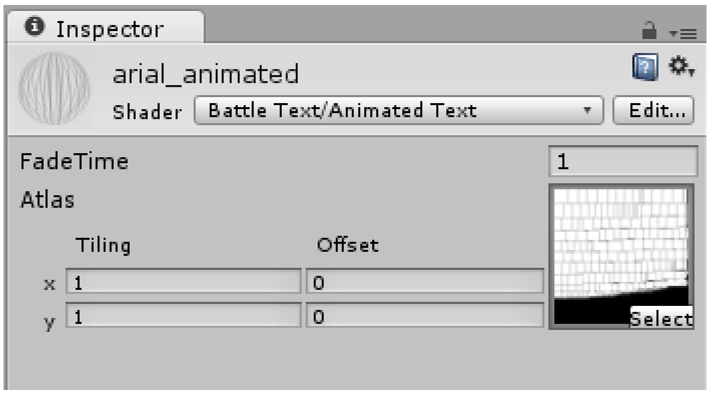
Compression: Deflate

Step 4

From the options menu select “Save bitmap font as…”, save it somewhere on your hard drive. I chose the name “arial” and saved it on my desktop. This creates two files, one arial_0.png and arial.fnt. Rename the arial_0.png to arial.png and rename arial.fnt to arial.txt.

Step 5

Drag both arial.png and arial.txt into the BattleText/Fonts folder. Create two new materials, one called arial_static and one called arial_animated, set the Battle Text/Static shader on the static one and Battle Text/Animated shader on the animated one.


Inspector

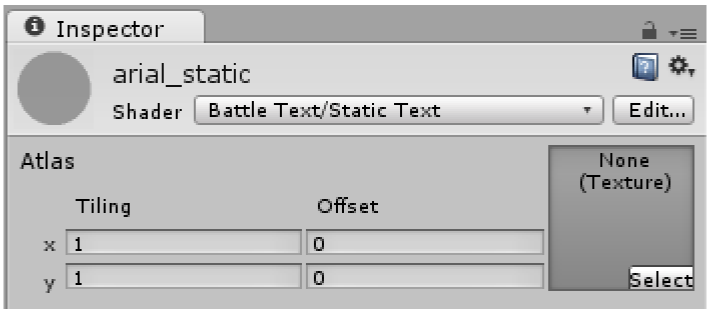
arial_animated

Shader: Battle Text/Animated Text

FadeTime: 1

Atlas

	Tiling	Offset
x	1	0
y	1	0


Inspector

arial_static

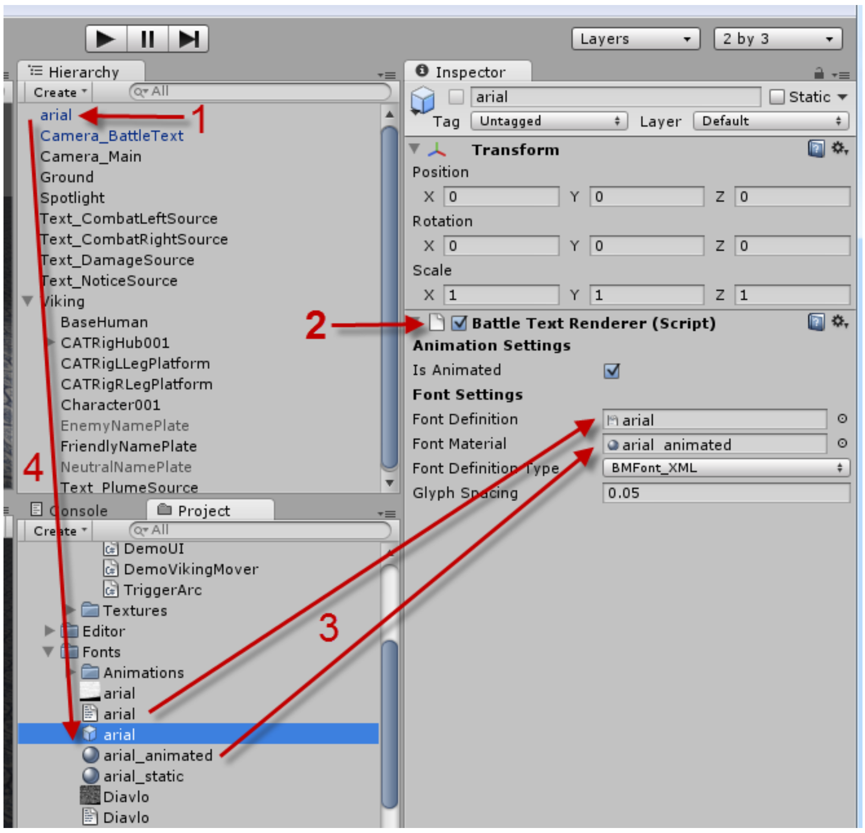
Shader: Battle Text/Static Text

Atlas: None (Texture)

	Tiling	Offset
x	1	0
y	1	0

Now do this

1. Create a new game object, call it arial
2. Drag a BattleTextRenderer script to it
3. Drag the arial font definition and arial_animated material to the script
4. Drag the game object into the BattleText/Fonts directory to create a prefab



arial, Static

Tag: Untagged, Layer: Default

Transform

Position: X 0, Y 0, Z 0

Rotation: X 0, Y 0, Z 0

Scale: X 1, Y 1, Z 1

Battle Text Renderer (Script)

Animation Settings

Is Animated: ✓

Font Settings

Font Definition: arial

Font Material: arial_animated

Font Definition Type: BMFont_XML

Glyph Spacing: 0.05

Now you’re ready to use your custom font!
That’s all folks!