User Guide

ProTips - UGUI Tooltip System

Unity asset by ModelShark Studio

What Is ProTips?

ProTips is a tooltip solution for Unity that uses the built-in Canvas UI, and works with both TextMeshPro and Unity Text UI out-of-the-box. ProTips supports desktop and PC mouse hover events, as well as press-and-hold events for touch devices. Tooltips can also be triggered from remote events such as button clicks or code, which is useful for making in-game interactive tutorials.

With ProTips, you can create your own tooltip styles visually using the Unity Canvas and the templates provided in our package. This documentation provides step-by-step instructions to get you started.

Features

Works with both TextMeshPro and Unity Text UI. ProTips is designed to with TextMeshPro by default, but if you want to use Unity's Text UI, you can open the ProTips UGUI Demos unitypackage to add those demos

Canvas overflow protection. If a tooltip's size would cause it to run off the UI canvas, it will automatically be repositioned.

Create new tooltip templates visually. Position and style elements on your tooltip templates visually, using the familiar UGUI canvas components. When finished, simply save your tooltip as a prefab and you can use it on any tooltip trigger.

Rich text formatting. Use familiar rich text markup in your tooltips to control font color, size, style, and weight.

Control the position of every tooltip. You can easily control the display position and offset of every tooltip individually.

Easilymodify tooltip delay and fade speed. These parameters are exposed on the Tooltip Manager. Change them in the editor, or at runtime using code.

Tooltipsfor dynamically-created elements. Are you creating dynamic elements through code that need tooltips? No problem!

Contents

Important Terms

Tooltip Style
This is a prefab that contains the tooltip layout, look and feel, and dynamic fields that get replaced with data on the Tooltip Trigger.

Tooltip Trigger
This is any object that launches or "triggers" a tooltip. It requires a Tooltip Style to function

Flexible tooltip options. You can create tooltips that stay open for a while, stay open until closed, block other tooltips from opening, and trigger tooltips from one object that appear over another.

High performance. Only one tooltip is instantiated per tooltip style and turned on/off as needed. This keeps memory usage and draw calls to a minimum.

Attach tooltips to 3D objects. You can attach tooltips to 3D world space object by adding the TooltipTrigger script to them, and the tooltips will be rendered on the canvas.

Tooltips work when Time is Paused. Typically, when time is paused in a game, UI elements are also frozen. But ProTips tooltips will work just fine when time is paused.

Full source code. If you ever need to extend the tooltip system for your specific requirements, full C# source code is included.

Getting Started

We've worked hard to make ProTips as easy to use as possible. To get started right away, follow these simple steps:

Note: There's already a completed QuickStart scene in the ProTips project folder if you want to skip this step-by-step setup

1) Import the ProTips asset package into your project.
2) Create a new scene and add a UI Panel. A Canvas and EventSystem will be automatically added for you.
3) On the top menu in Unity, click Window => ProTips => Quick Setup. This will add the TooltipManager to the scene.
4) Now add a Button to your Panel. Click the Button, and in the inspector choose Add Component => TooltipTrigger script.
5) Now the button is a tooltip trigger. But first we need to specify what tooltip style to use. Find the CleanSimple prefab in the ProTips\Tooltip Prefabs folder and drag it to the Tooltip Style field.
6) Notice how there is a new field added: [BodyText]. Whatever text you type in here will replace the [BodyText] parameter in the tooltip. For now, type "This is a simple tooltip".
7) Run the scene. When you mouse over the button, you should see your tooltip displayed.

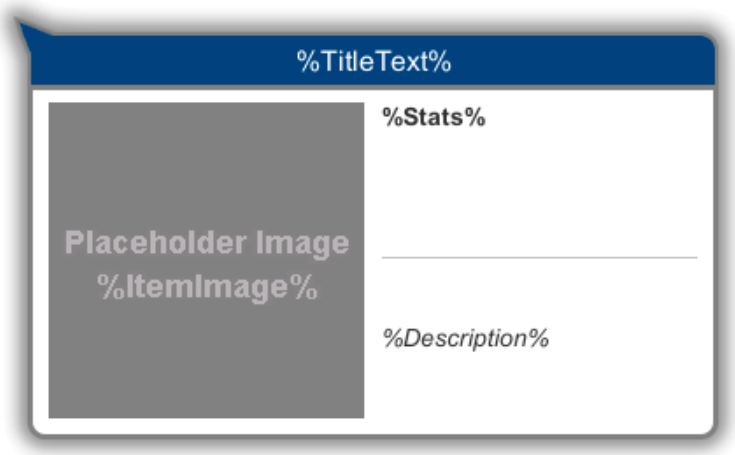
Dynamic Fields

We provide three types of dynamic fields you can use. When you create a dynamic field on your tooltip, it will show up in the TooltipTrigger inspector, where you can change it at design time, or through code at runtime. Our demo scene provides many examples of both methods.

Dynamic Text Fields

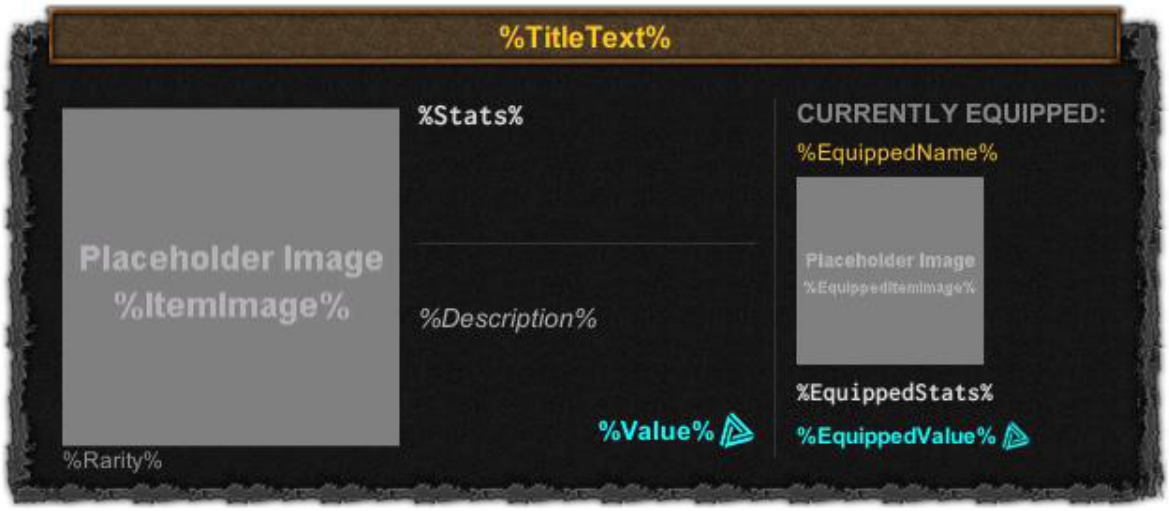
Open the scene named "Tooltip Workshop" to see some of the example templates available. Every field marked with %FieldName% is a dynamic text field that will show up on the TooltipTrigger inspector, ready to be filled in with text that replaces the placeholder at runtime.

Dynamic Image Fields



The placeholder images are also parameter fields. If you look at one of the placholder Image components in the inspector, you'll see that they have a DynamicImage script attached. This allows you to give each dynamic image a placeholder name so you can replace them with a different image when they're triggered at runtime.

Dynamic Sections



You can even have dynamic sections in your tooltips. If you look at this example, the **Equipped Item** section is actually a dynamic section. It has the DynamicSection script attached to it, which gives it a variable name that shows up in the TooltipTrigger inspector. The only option you have with a dynamic section field is on or off. In the equipment example, this section is only turned on for items that are not currently equipped by the player.

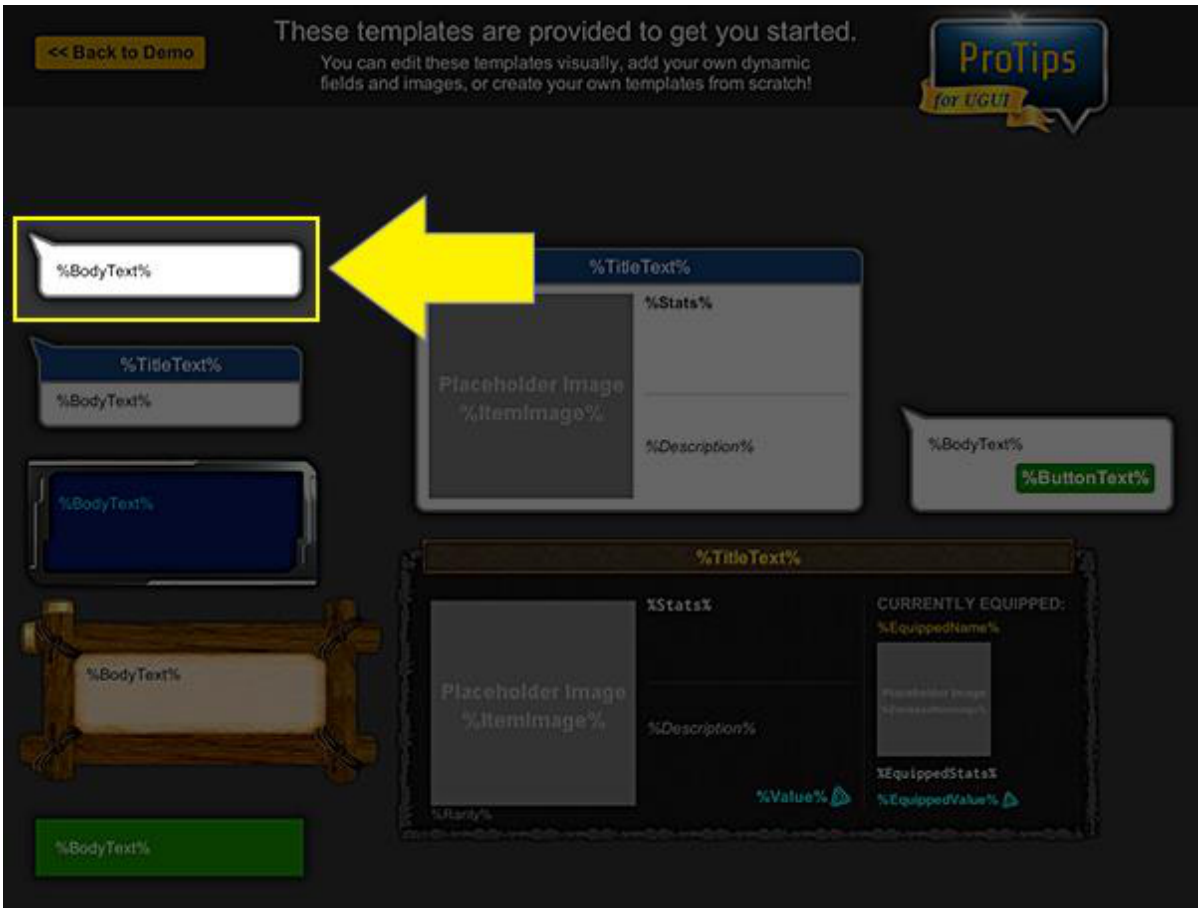
Creating Your Own Tooltips

The easiest way to create your own tooltip is to start with one of ours and modify it to fit your game. Let's go through the process so you can see how it's done!

Picking a Starting Template

Let's say we want to create a radically different tooltip. No matter how different it may be, we always start by deciding what dynamic fields we will need. For our tooltip, let's keep it simple. We just need a **main text area**.

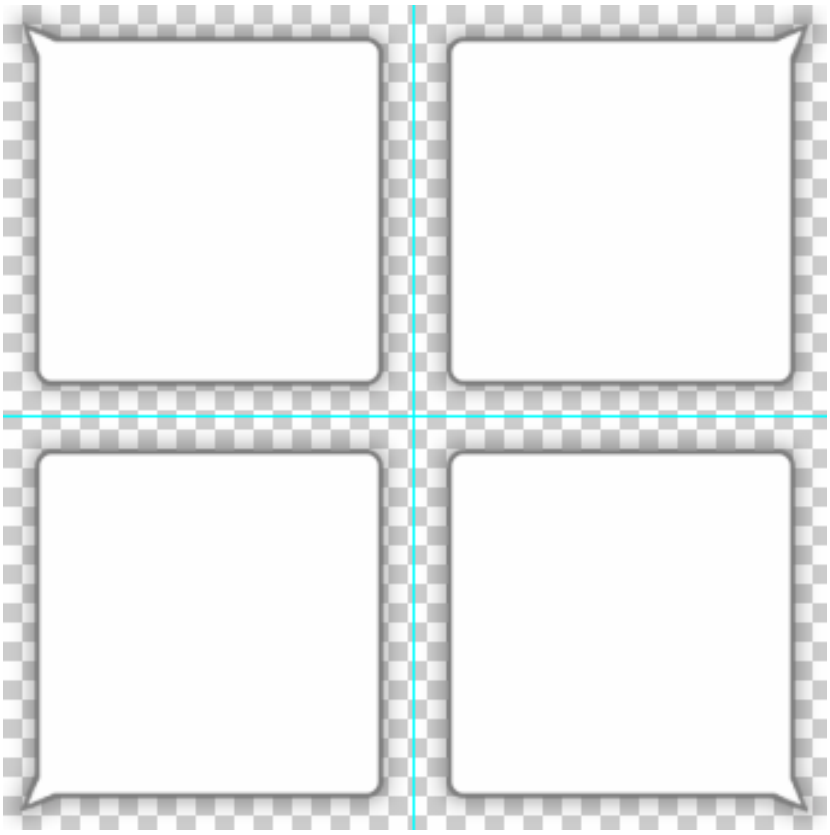
Now that we know what type of tooltip we're creating, let's find something similar in the Tooltip Workshop scene (included in the ProTips package). It looks like the white one with rounded corners (below) has the correct fields, although visually it's not what we have in mind. That's okay, we'll fix it soon.



Creating a Background Image

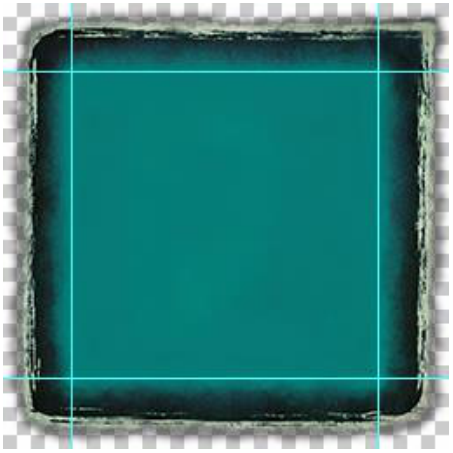
The dynamic field name "%BodyText%" is fine for our tooltip, but the look is too plain. So let's create a new background image for the tooltip. If you click on the tooltip in the Unity editor, you'll see it has a TooltipStyle script component on it. On this script are four fields: Top Left Corner, Top

Right Corner, etc. These fields are the background images for each of the four tip positions. Let's open the **TooltipStyle-CleanSimple.psd** sprite sheet in our image editor and take a look:



This image is 512x512 resolution. So all we need to do is make sure our new artwork fits within each 256x256 quadrant and has its pointer (optional) pointing in the right direction.

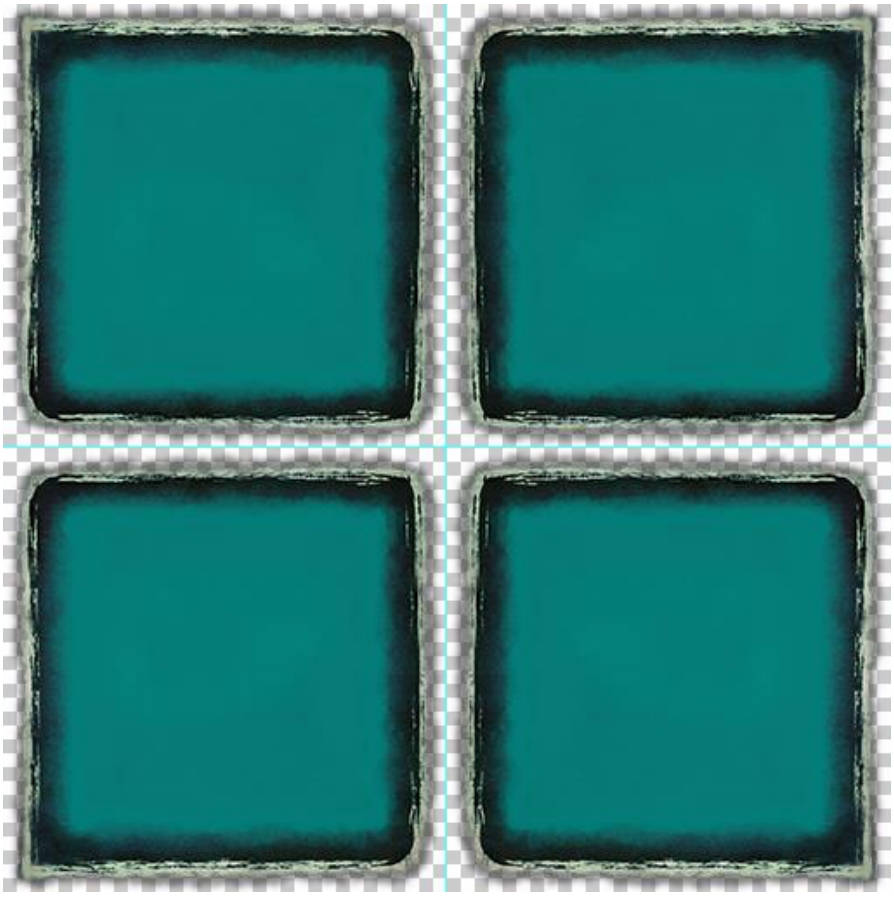
I've created some new artwork to use for our tooltip. It has a blue center and a grunge border style (see below).



The plan is to use the one sharp corner to indicate where the tooltip pointer is. Everything between the blue guide lines will be the "fill" area of the tooltip. The corner is the only part that won't tile. The guide lines are set at **40px** from the edge. We'll need that info later when we import the image into Unity as a sliced sprite.

Note: If you would like to follow along, this artwork is in the ProTips asset package, under \ProTips\Demos (safe to delete)\Textures\BlueGrungeOriginal.psd

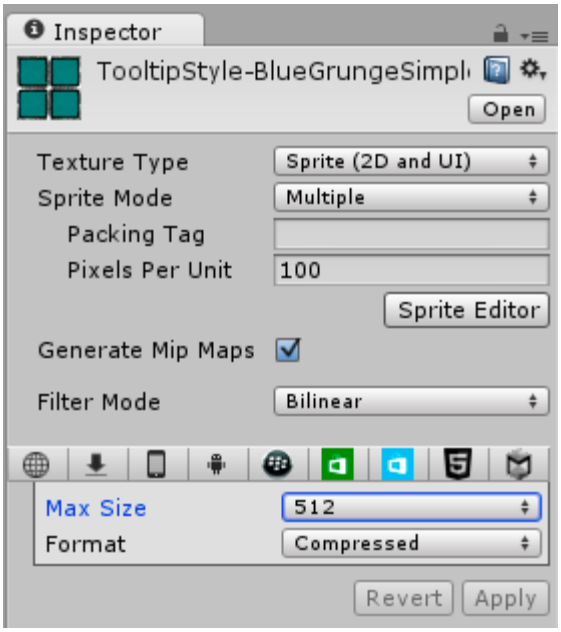
Now we simply arrange the images so the corners point in the same direction the CleanSimple tooltip images did:



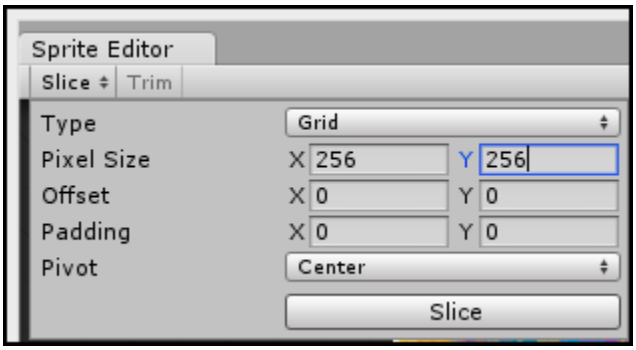
Now save the image to the Unity asset folder, under Textures.

Note: If you're following along, this image is in the ProTips asset package, under \ProTips\Textures\ TooltipStyle-BlueGrungeSimple.psd.

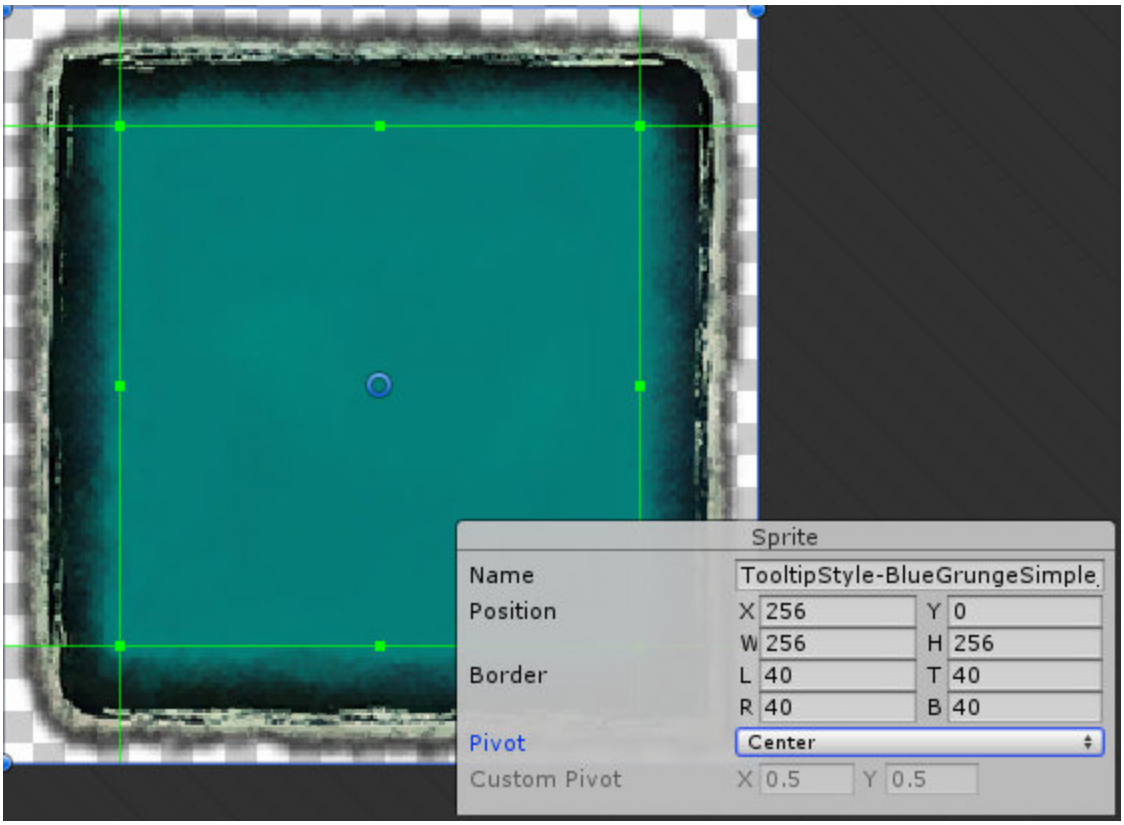
Now click on the image in the Unity inspector and change its import properties to match the properties shown below.



Click on the button for the Sprite Editor (above). An editor window will open so you can adjust the sprite sheet. We want to select Slice -> Type: Grid, Pixel Size: 256x256. Then click Slice. This will slice our 512x512 image into four 256x256 sprites.



Next, we need to tell Unity where our borders are. Remember earlier when our guide lines were 40px from the edge? We just need to assign 40px as the border width for each of our sprites. Click on each of the sliced images and set the border values as shown below.



When finished setting the borders on all four images, click Apply.

Creating a Tooltip Style Prefab

Now we need to create our own TooltipStyle prefab. In the Tooltip Workshop scene, duplicate the CleanSimple game object under the Canvas. Rename it "BlueGrungeSimple" and move it to an empty area of the Canvas so you can work on it.

On the TooltipStyle component of our game object, replace the four background images (Top Left Corner, Top Right Corner, etc) with the sprites from our new artwork. While we're at it, on our tooltip's Image component, go ahead and replace the sprite with our new top left corner image. This will help us troubleshoot any layout or style issues. Your tooltip style should look like this:

It's starting to look good, but there are a couple of issues. On the right middle and left middle borders, there's some tiling artifacts because the tooltip height is allowed to shrink down too much. Also our font isn't going to be very readable against the darker background.

Select the BodyText game object and find the Layout Element component on it. Check "Min Height" and enter "60". On the Text component of the BodyText game object, change the font

and color to something that looks nice. I chose Inconsolata-Bold, normal style, 16pt, and for color pure white (255,255,255) with alpha set to 220. Here's what we have now:

Finally, we need to save our tooltip style as a prefab. Just drag the game object into the \ProTips\Tooltip Prefabs in the project tab. Now we're ready to use it!

Testing Your Tooltip Style

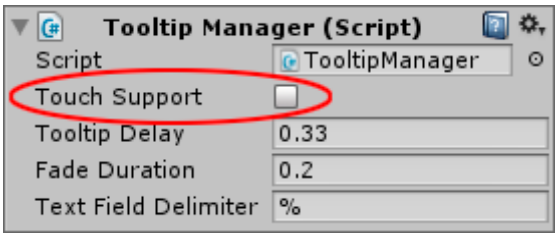
Switch to the ProTips Demo scene and in the hierarchy tab, select the Canvas -> Middle -> Character Panel. Create a new UI -> Button under the Character Panel. Now add the TooltipTrigger component to the button, and drag the TooltipStyle-BlueGrungeSimple prefab to the Tooltip Style field. Add some body text for the tooltip, and run the scene. Here are the final results:

Advanced Options

Here are a few of the more advanced options you may need to know about.

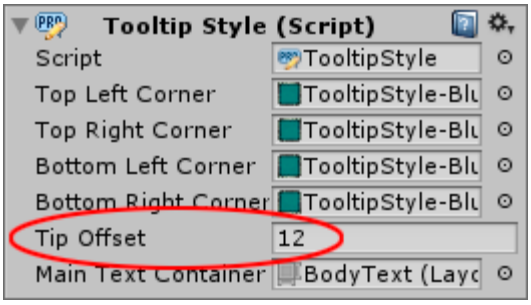
Touch Support

For touch devices, ProTips supports press-and-hold events for triggering tooltips. The demo scene illustrates this with the green "Touch Device" button at the top. The option to turn touch support on/off is located on the Tooltip Manager:



Tooltip Offsets

Sometimes you may need to change how much a tooltip is offset from the trigger point, whether it's the mouse pointer or the corner of the element that triggered the tooltip. This is particularly needed for tooltips with large borders. On the TooltipStyle script, there is a tooltip offset for this purpose:



Adjust this number up, and it will offset the tooltip back towards the pointer. If your tooltip is overlapping the pointer (or trigger element corner), adjust the number back down. You may need to do some back-and-forth testing to find the perfect offset, but this fine control is there if you ever need it.