Daikon Forge GUI Library

Table Of Contents

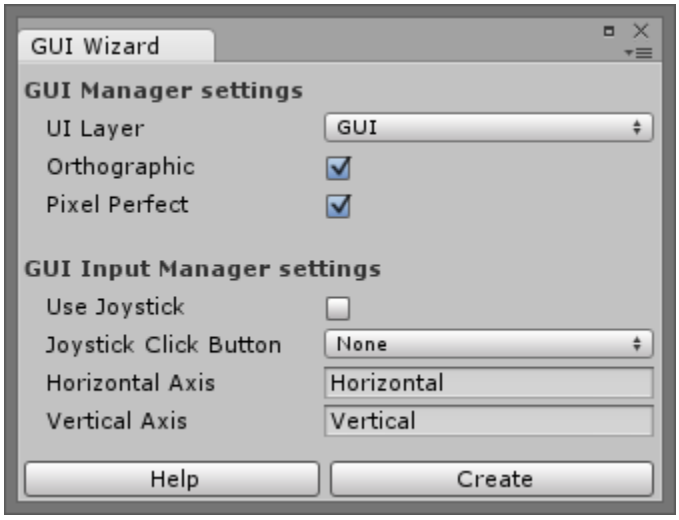
Getting Started

Installation

To install DFGUI from the Asset Store, open Window → Asset Store, open the Download Manager, navigate to the Daikon Forge GUI package, and click Download/Import.

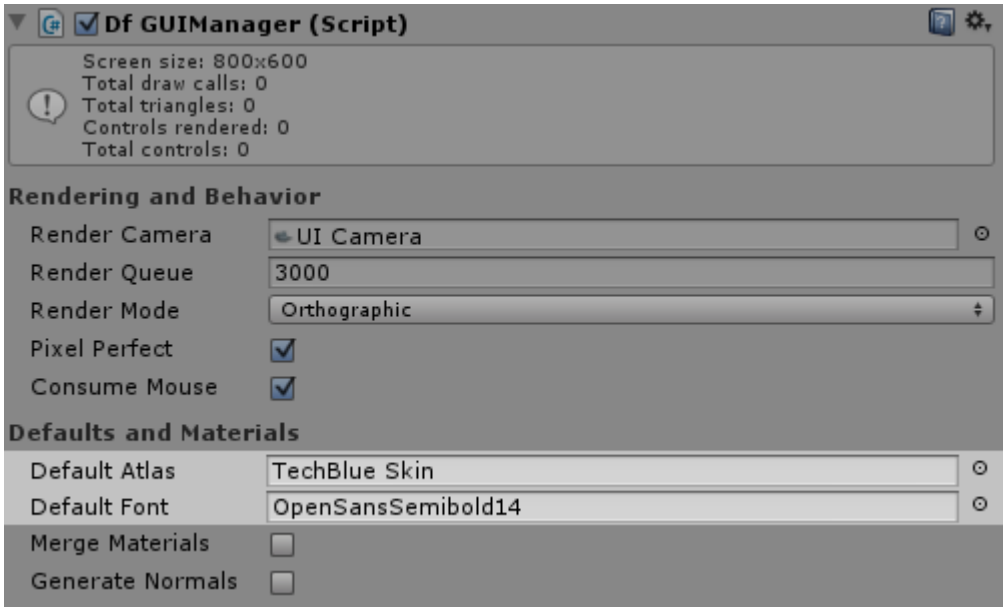
Basic Setup

To create a new UI, first create a new layer for the UI to reside on. Open the Layer manager and create a layer (any layer except for 31).
Next, open the UI Wizard (GameObject → Daikon Forge → UI Wizard). Pick the layer you created for the UI Layer, and click Create.



The UI Wizard

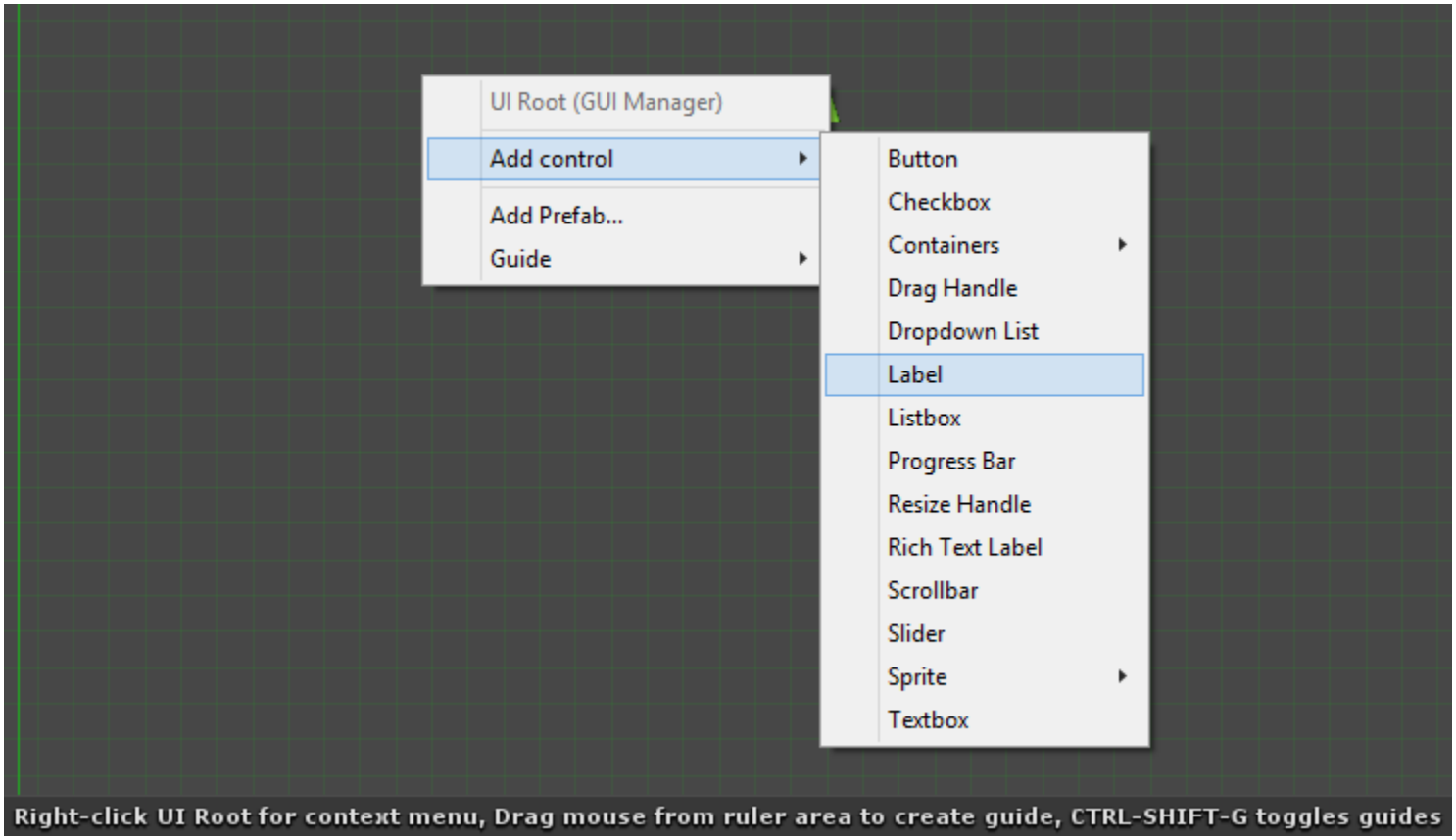
You should now have a new UI Root.
Before we add controls, let's assign a default texture atlas and font to use. First, in the inspector for the dfGUIManager component attached to the UI Root, find the Default Atlas slot and click the circle next to it. Select the included TechBlue Skin atlas. Additionally, click the circle next to the Default Font slot and pick OpenSansSemibold14.



Assigning an atlas and font to use.

Adding Controls

To add a new control, first select the UI Root, and right click it in the scene view. Select Add Control, and pick the control you wish to add. We'll add a label for this example.



Adding a Label to the UI

You should see a label added at the location of the mouse cursor. There are a variety of other controls available, which we will cover later.

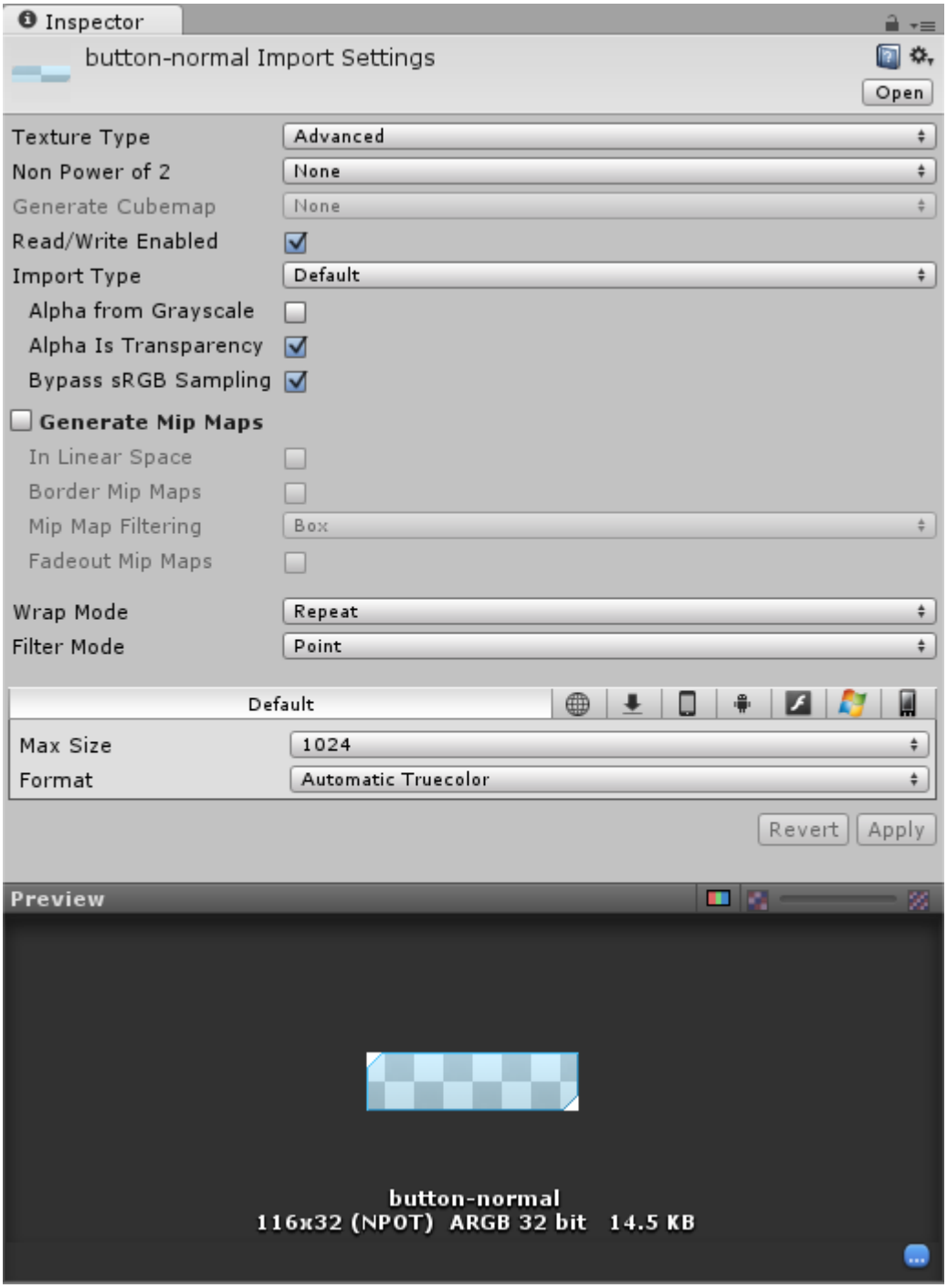
Texture Atlases and Draw calls

DFGUI makes use of Texture Atlases in order to reduce draw calls as much as possible. A Texture Atlas contains a collection of sprites, all arranged in one large image.

DFGUI issues as few draw calls as necessary in order to correctly render the UI. If all controls utilize the same atlas, DFGUI will be able to render all of them in one draw call. If controls use different atlases, DFGUI must split up the controls into different draw calls. Note that this depends on render order. Given three controls, two using atlas A and one using atlas B, if the third control is positioned on bottom or top, all three controls will be rendered in two draw calls. However, if the third control is layered in between the first two, they must be rendered in three draw calls in order to render in the correct order. This behaviour can be changed by checking the “Merge Materials” checkbox on the UI Root, but keep in mind this can result in draw order issues if controls overlap.

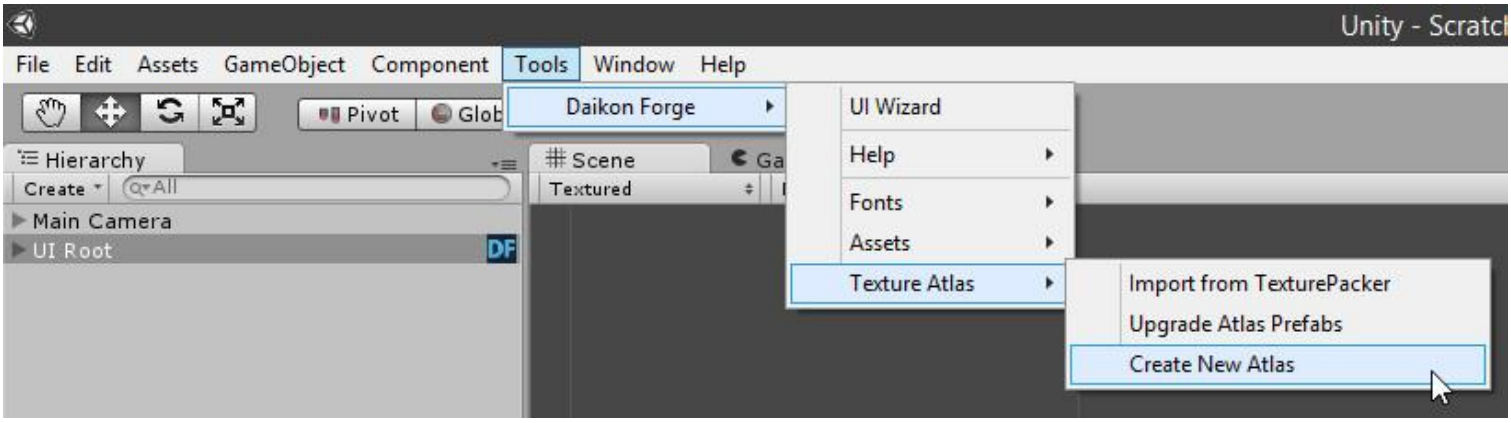
Creating a DFGUI Texture Atlas

To create a texture atlas from source textures, first set all Source textures to Advanced, ensure Read/Write Enabled is checked, and that all textures are set to RGBA 32 bit.



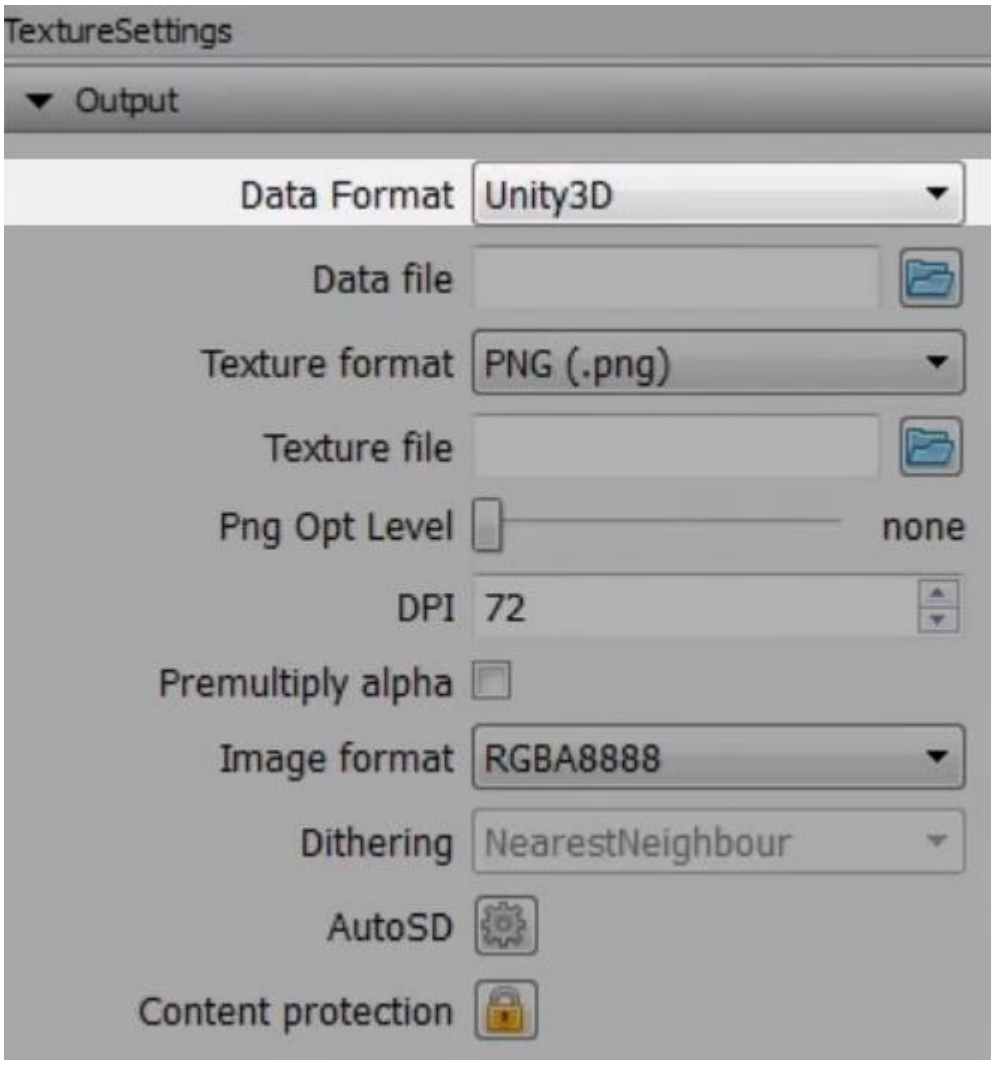
Example texture import settings

Select all of the textures you wish to include, and select Assets → Daikon Forge → Texture Atlas → Create New Atlas. Choose a location to save the atlas to and click Save.

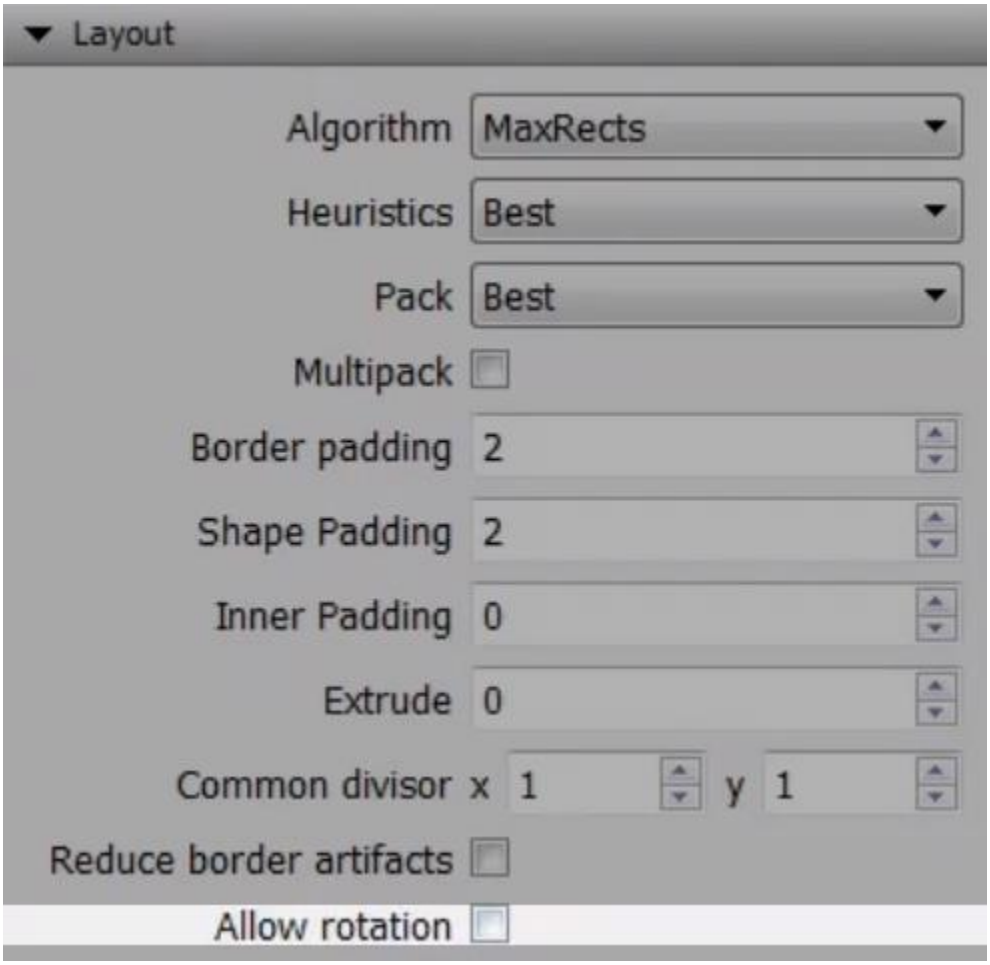


Importing a TexturePacker Atlas

To import a texture atlas from TexturePacker, select Assets → Daikon Forge → Texture Atlas → Import Texture Atlas. Take the texture and data file generated from TexturePacker and drop them onto the appropriate slots in the pop up window. Make sure to use the Unity3D format when exporting from TexturePacker

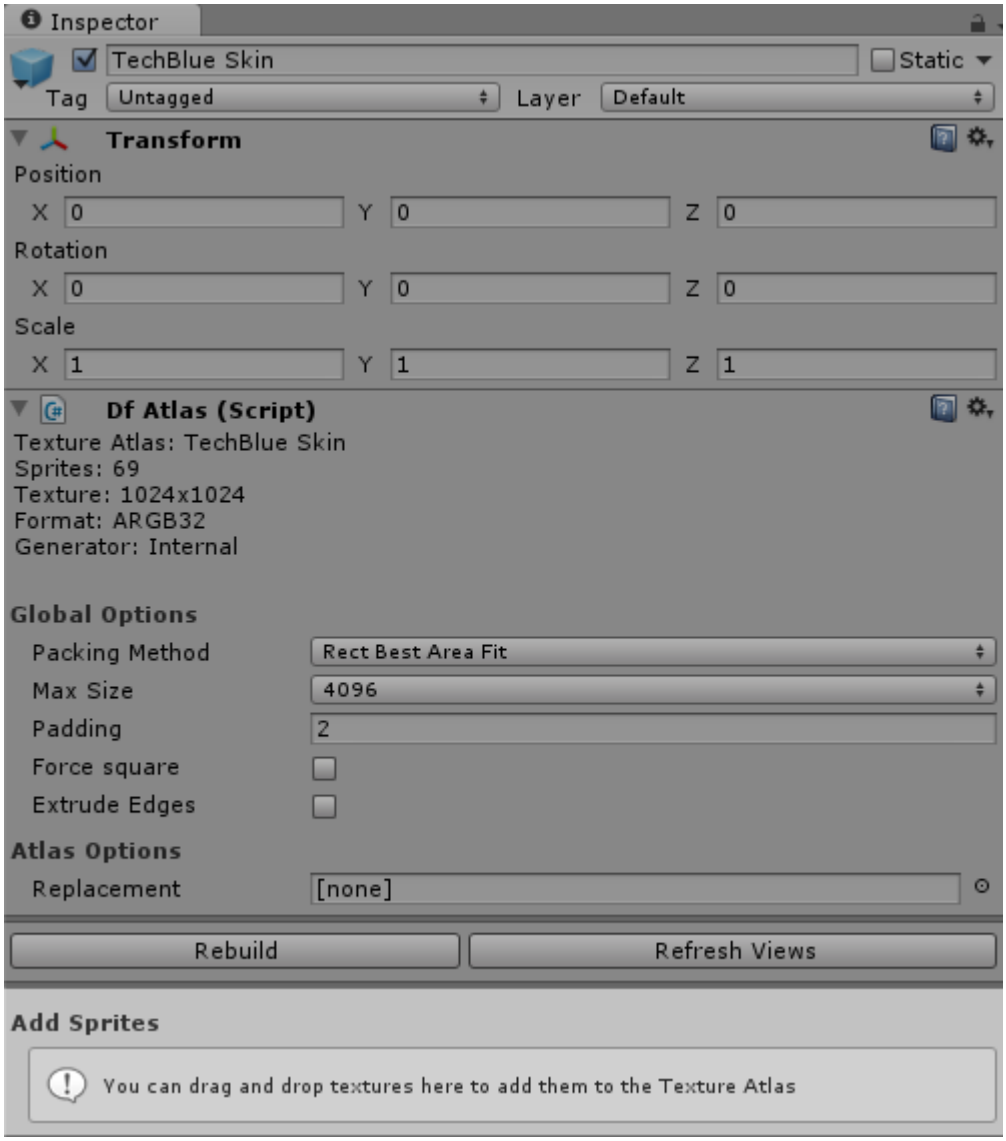


Additionally, DFGUI does not yet support image rotation, so be sure to disable it in Texture Packer.



Adding And Removing Sprites

When using a DFGUI texture atlas, you can add sprites at any time by dragging the desired sprite onto the Add Sprite area of the Texture Atlas prefab.

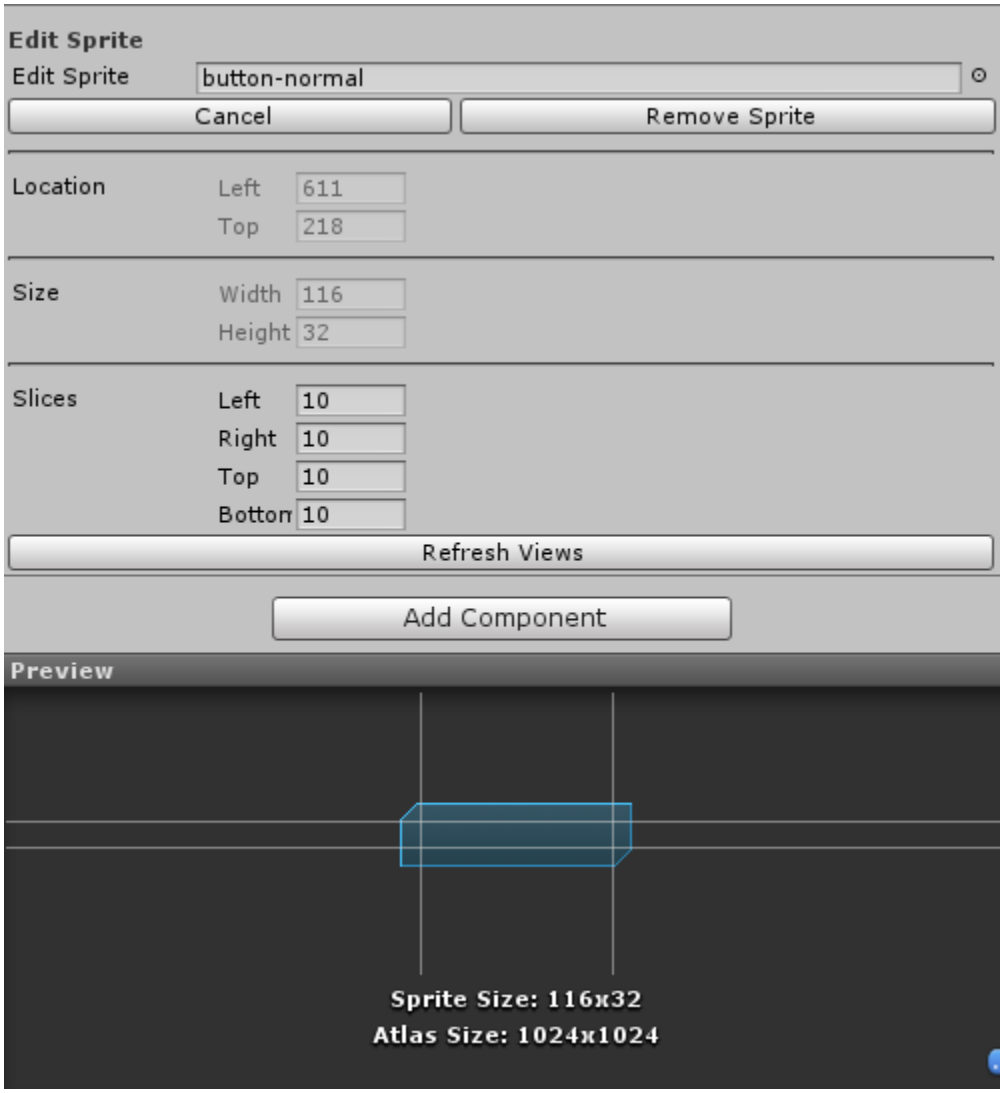


Drag textures onto the Add Sprites area to add them to an atlas

You can also remove sprites by locating them in the list of sprites which follows and clicking “Delete”.

You can edit sprite settings or delete them from the atlas.

You can click Edit to edit the sprite slice settings (used for sliced sprite rendering).



Editing a sprite's slice settings

Importing Fonts

In DFGUI, there are two kinds of fonts: Bitmapped fonts, and dynamic fonts. Bitmap fonts are packed into an atlas and therefore do not incur extra drawcalls - however, all characters to be used must be known ahead of time, making them unsuitable for very large character sets such as Chinese. Dynamic fonts can display any Unicode character without having to pack a font image into an atlas, however they will incur extra draw calls.

Importing Bitmap Fonts

DFGUI uses the AngelCode format for bitmap fonts. Many programs can export this format, such as BMFont or Hiero. To import a bitmap font, select the font definition file and select Assets → Daikon Forge → Fonts → Create Bitmapped Font, and choose a location to save the font to. Additionally, you **must** add the font texture to any atlases you wish to use the font with as a sprite.

Importing Dynamic Fonts

To import a dynamic font, simply select the TTF file you want to use and select Assets → Daikon Forge → Fonts → Create Dynamic Font and choose a location to save the font to.

Rendering Text

There are two ways to draw text in DFGUI - with regular Labels, or with Rich Text Labels. Labels are more efficient as they have a simpler markup syntax (or can even have markup disabled), but are more limiting. Rich Text Labels employ a subset of HTML and are therefore very powerful, but are somewhat slower to parse. Additionally, Rich Text Labels rely on dynamic fonts for rendering, whereas Labels can use bitmapped fonts.

Standard Labels

To create a standard label, right click the UI root and choose Add Control → Label. Click the circle next to the Font slot and select the font you wish to use, or drag the font prefab from the project view onto the slot. Text can be entered into the Text area at the bottom. Labels use the following syntax:

- [color #rrggbb][/color]
    - Allows you to colorize a span of text. Uses standard hex notation for colors.
- [sprite spritename]
    - Allows you to insert a sprite in the middle of text. For instance, if your atlas had a sprite called “circle” you would write “[sprite circle]” to insert the sprite into the text.

Rich Text Labels

To create a rich text label, right click the UI root and choose Add Control → Rich Text Label. Click the circle next to the Font slot and select the dynamic font you wish to use, or drag the font prefab from the project view onto the slot. Text can be entered into the Text area at the bottom.
Rich text labels use a limited subset of HTML. To see examples of the officially supported HTML features in DFGUI, check out the included Rich Text Example demo scene.

Sprites

Basic

To create a basic sprite, right click the UI root and choose Add Control → Sprite → Basic. Basic sprites can render a 2D sprite from a texture atlas. To choose a sprite, click the circle next to the Sprite slot and choose the sprite to use.

Sliced

To create a sliced sprite, right click the UI root and choose Add Control → Sprite → Sliced. Sliced sprites are just like Basic sprites, except they use nine-slice scaling to allow for better stretching of sprites like buttons and windows. With nine-slice scaling, the sprite is divided into nine sections - four corners, for edges, and the middle. The corners are not stretched, each edge is stretched in one direction, and only the center is stretched in both directions. The size of the borders for a sprite are defined in the Atlas.

Tiled

To create a tiled sprite, right click the UI root and choose Add Control → Sprite → Tiled. Tiled sprites allow you to repeat a single sprite several times within an area, for instance to create a generic patterned fill.

Radial

To create a radial sprite, right click the UI root and choose Add Control → Sprite → Radial. Radial sprites allow you to fill the sprite radially. By default the origin starts in the center, giving a 360^o fill. You can set the origin to one of nine points on the sprite - the center, the corners, and the edges. Corner origins give 90^o fills, edge origins give 180^o fills, and the center as mentioned yields 360^o fills.
This might be used to implement spell cooldowns, for example.

Texture

To create a texture sprite, right click the UI root and choose Add Control → Sprite → Texture. Texture sprites work like Basic sprites, except you can define any Texture or Material to use for the sprite, rather than just a sprite from the atlas. Because of this, Texture sprites will increase draw calls but allow you to use sprites not defined in an atlas.

Web

To create a web sprite, right click the UI root and choose Add Control → Sprite → Web. Web sprites are an extension of Texture sprites, and allow you to specify the URL of an image to download and display.

Basic Controls

Button

To create a button, right click the UI root and choose Add Control → Button. You can specify images for Normal, Hover, Click, Focused, and Disabled, as well as colors. You can also set a Font to use for displaying button text.

Checkbox

To create a checkbox, right click the UI root and choose Add Control → Checkbox. Create a label and a sprite, and parent them to the checkbox object, then drag these onto the Label and Check Icon slots of the Checkbox. If you want the checkbox to act as a radio button in a group, you can specify a container which contains all of the checkboxes in the group.

Textbox

To create a textbox, right click the UI root and choose Add Control → Textbox. Choose a font for your textbox, as well as a Blank texture (a small, plain white image). The Blank texture is used to render the selection cursor and text selection highlight.

Progress Bar

To create a progress bar, right click the UI root and choose Add Control → Progress Bar. Choose a background sprite, and a sprite to use for fill.

Slider

To create a slider, right click the UI root and choose Add Control → Slider. Choose a sprite to use for the track. Next, create a sprite to use for the thumb, parent it to the slider, and drag it onto the Thumb slot of the Slider. Optionally, you can add a sprite to use for progress and drag it onto the Progress slot.

Scrollbar

To create a scrollbar, right click the UI root and choose Add Control → Scrollbar. Create sprites for the Track and the Thumb, and optionally for increment/decrement buttons, and parent them to the scrollbar. Drag each onto the appropriate slot of the Scrollbar.

Containers

Panel

To create a panel, right click the UI root and choose Add Control → Containers → Panel. Panels can serve as containers for other controls. Panels can be used to group controls, provide clipping, assist in control layouts, and can also perform automatic layout with the FlowLayout component.

Scrollable Panel

To create a scrollable panel, right click the UI root and choose Add Control → Containers → Scrollable Panel. Scrollable Panels are a lot like regular Panels, except they have support for scrolling children. You can also associate horizontal and vertical scrollbars with a scrollable panel.

Tab Control

To create a tab control, first create the tab page container via Add Control → Containers → Tab Control → Tab Page Container. Then, create the tab strip via Add Control → Containers → Tab Control → Tab Strip. Drag the Tab Page Container control onto the Tab Pages slot of the Tab Strip control. Click “Add Tab” to add a tab control to the strip, and a corresponding tab page to the Tab Page Container.
Tab controls are used to implement tabbed UIs, such as an options screen with, for example, Graphics, Audio, and Control tabs.

Data Binding

In DFGUI, data binding is used to help separate application logic from GUI logic. You can bind properties of components together, hook up a method on one component to an event of another, and more.

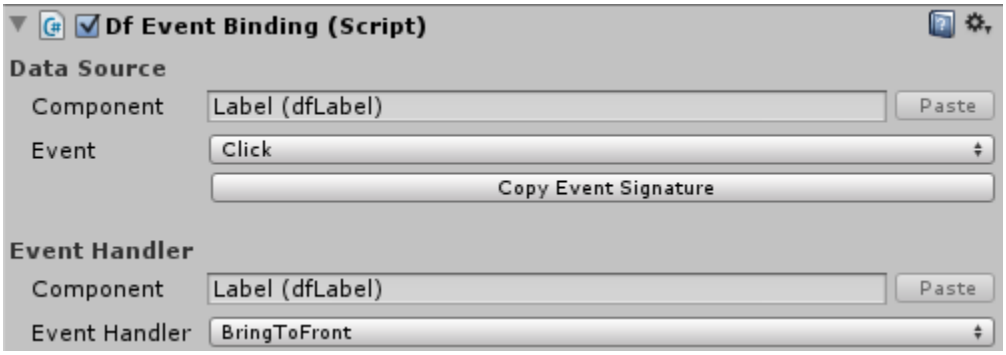
Property Binding

To add a property binding to a control, right click the control and choose Add Binding → Property Binding.

Right click the source component and choose Copy Component Reference, click the Paste button next to the Data Source component slot, then choose the property you want to copy from, from the dropdown. Right click the target component and choose Copy Component Reference, click the Paste button next to the Data Target component slot, then choose the property you want to copy to from the dropdown.
The value of the property on the source component will be automatically copied to the property on the target component. You can choose the Two-Way option to have the values copied both ways as well.
In addition, you can convert a Property Binding to an Event-Driven Property Binding (which may afford performance benefits with many concurrent property bindings) by clicking the Binding Model and choosing Event Driven (and vice versa by choosing Polled)

Event Binding

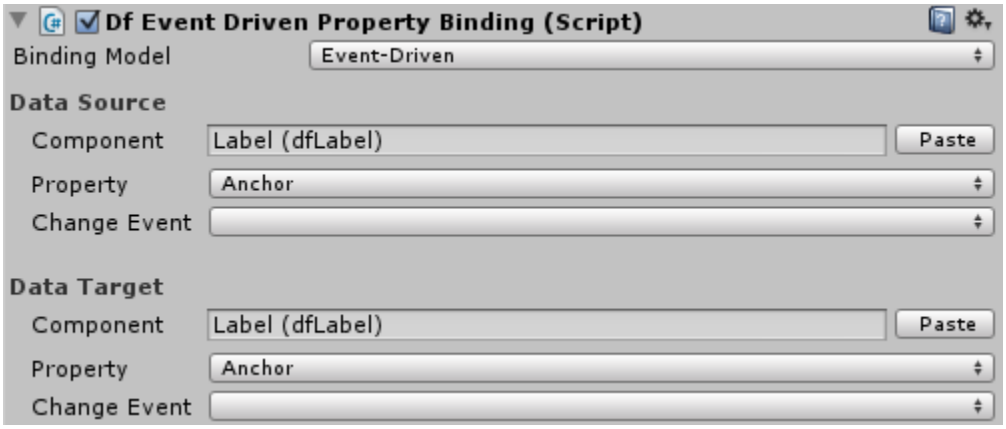
To add an event binding to a control, right click the control and choose Add Binding → Event Binding.



You can assign the Source and Target components the same way you would a Property Binding component. Choose the event of the Data Source, and the function to call on the Data Target. Keep in mind that the target function must either be parameterless, or match the signature of the bound event. IEnumerators are supported as target functions, in which case the function is started as a coroutine.

Event-Driven Property Binding

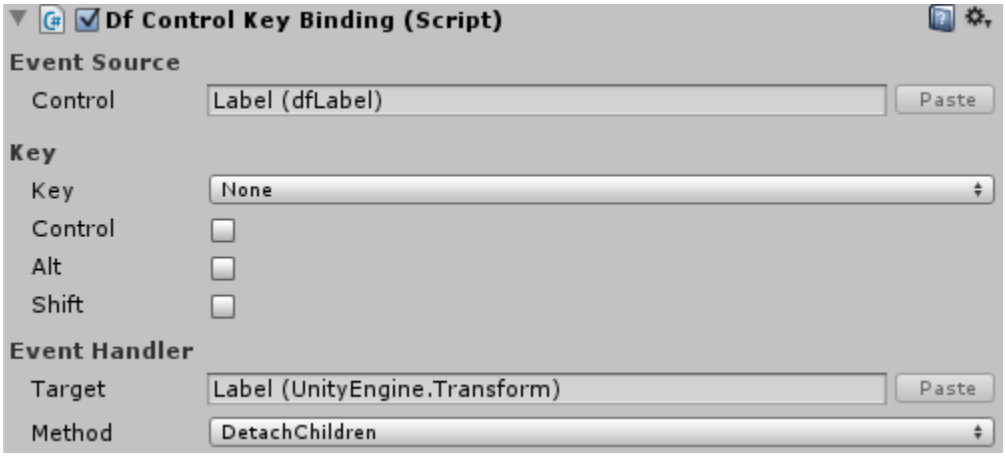
To add an event-driven property binding to a control, right click the control and choose Add Binding → Event-Driven Property Binding.



Event-Driven Property Bindings are a hybrid between Property Bindings and Event Bindings. Assign the Data Source and Target components as normal, choose the source and target property, and additionally choose events on the Source and optionally the Target (when the Source event triggers, the property is copied from Source to Target, and when the Target event triggers the property is copied from the Target to the Source).
You can change an Event-Driven Property Binding to a regular Property Binding by clicking the Binding Model and choosing Polled (and vice versa by choosing Event Driven).

Key Binding

To add a key binding to a control, right click the control and choose Add Binding → Key Binding.



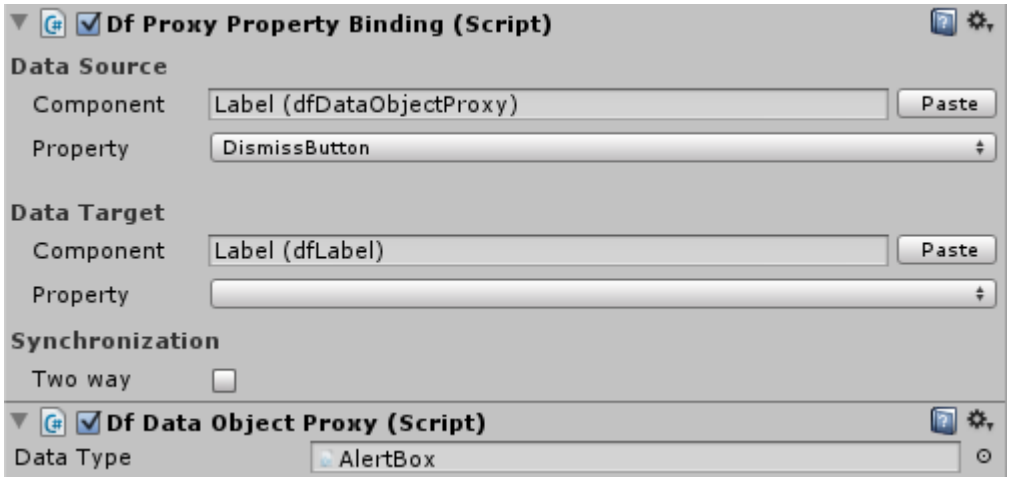
Key Bindings act like Event Bindings, except instead of choosing an event, you choose a keyboard key which will trigger the target method, and optionally modifier keys (Control, Alt, and/or Shift)

Proxy Data Object

To add a data proxy object, right click the control and choose Add Binding → Proxy Data Object. While the data binding components bind components together, sometimes it is desirable to bind to a non-Monobehaviour data source. This is what Proxy Data Object is for. You choose the data type it represents, and then you can bind to the proxy component. In script, you would assign the proxy component's Data property to the source data object.

Proxy Property Binding

In order to use Proxy Data Object, you need the Proxy Property Binding, which you can add via Add Binding → Proxy Property Binding.



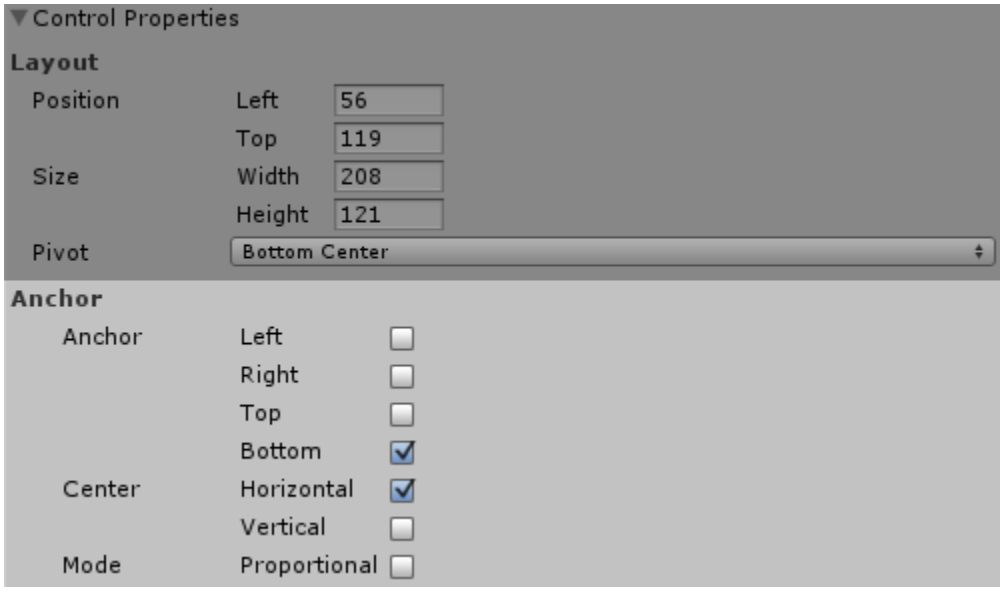
It works exactly as a regular Data Binding, except the source component is a Proxy Data Object which will allow you to bind to properties of the assigned data type (and of the assigned Data object).

Anchors & Layouts

DFGUI features powerful anchoring & layout features for designing multi-resolution, cross-platform GUIs.

Anchors

In DFGUI, each control features independant Top, Left, Right, and Bottom anchors. These will keep the given side of the control relative to the same side of the parent control, or the screen if there is no parent. Each control can also be set to Center Horizontal or Vertical, as well as use Proportional anchors.
If proportional anchors are used, control anchors act as percents rather than absolute pixel values. If you want a control to take up a relative area of a parent, enable all Top, Left, Right, and Bottom anchors and also check Proportional. This can be used to make much more resolution-independant GUIs without sacrificing pixel-perfect rendering.



Layouts

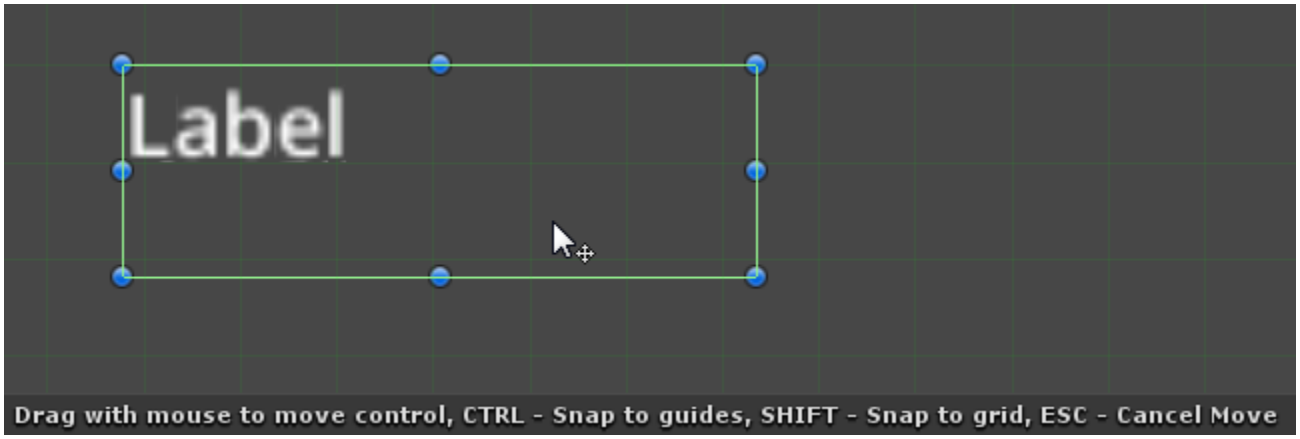
To an extent, you can also use automatic layouts. This is available as a checkbox on Scrollable Panels, while on regular Panels you can attach the Flow Layout component. This allows you to automatically arrange controls horizontally or vertically, as well as configure padding between controls.

Editor Features

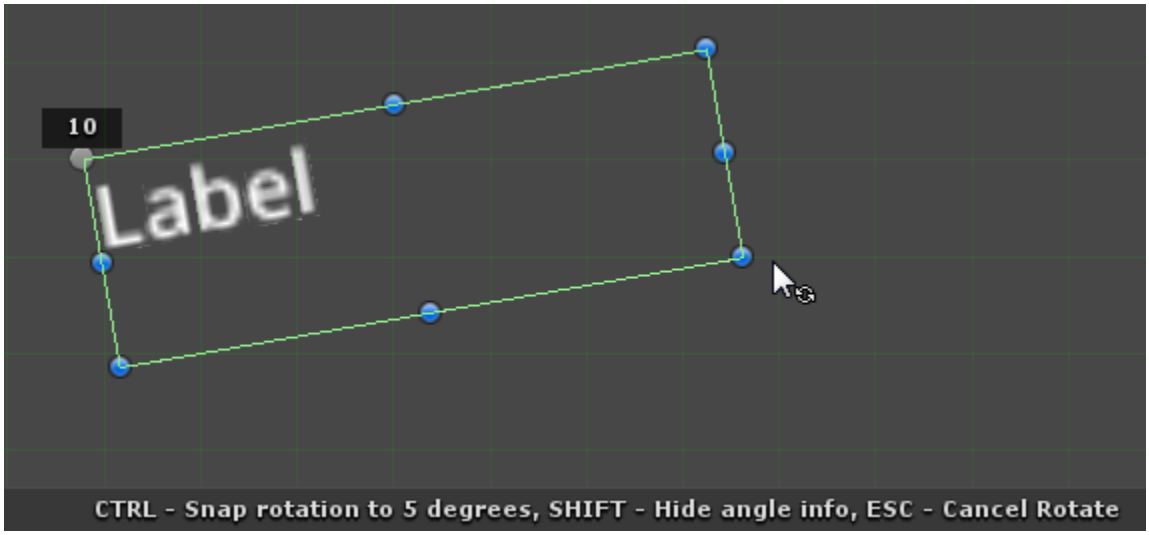
DFGUI includes many powerful design features geared toward designers and artists.

Moving Controls

With a control selected, left-click and drag it around to move it. You can also drag the blue handles to resize controls.



Finally, you can hover your mouse just outside the corners and drag to rotate the control.

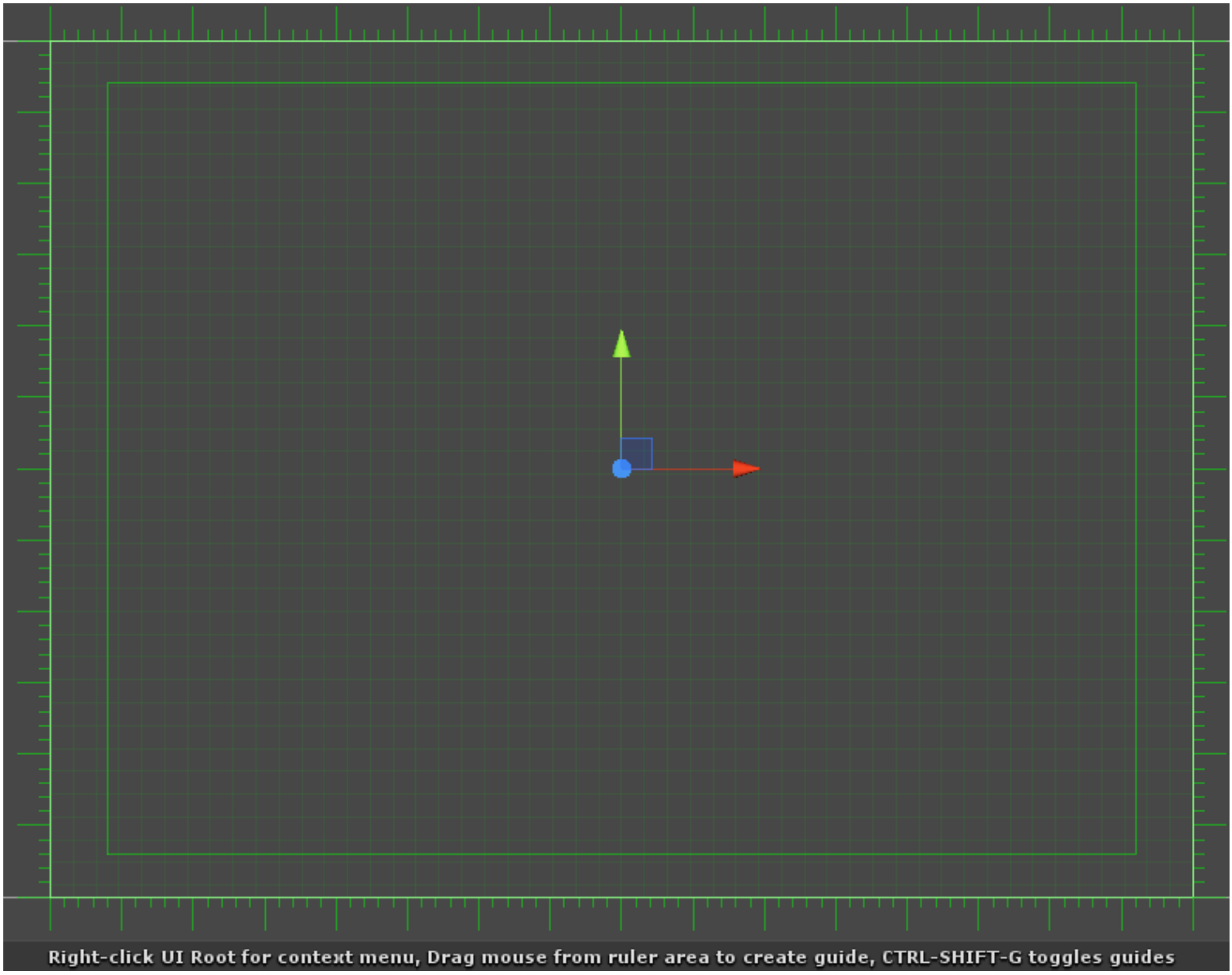


Grids

You can enable grids in DFGUI by selecting the UI root and choosing Show Grid. You can also configure the grid cell size. Grids can be snapped to while moving controls.
Grids are only shown in Orthographic view.

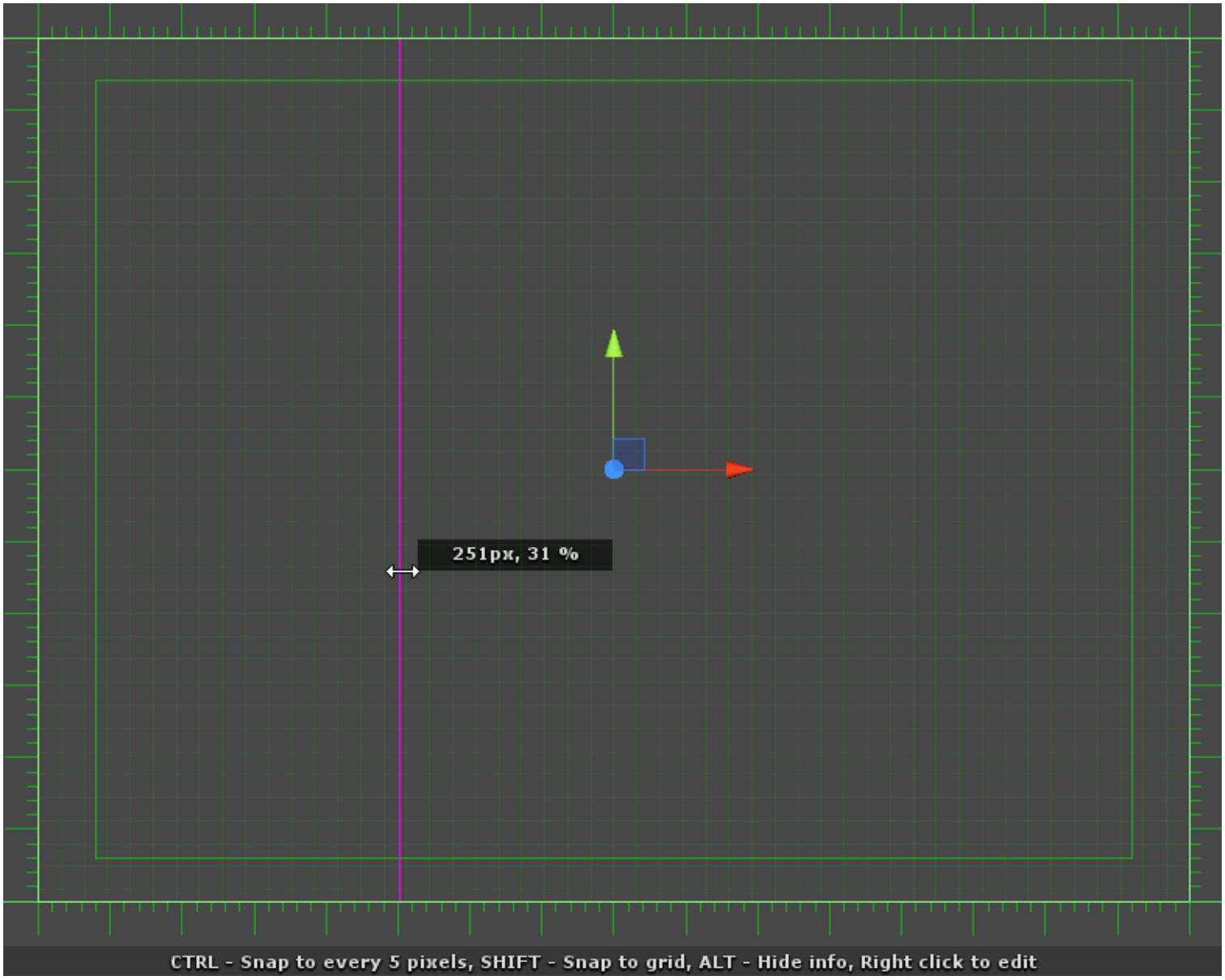
Safe Zone

You can enable the safe zone in DFGUI by selecting the UI root and choosing Show Safe Area. You can also configure the safe zone percent. This can be used to aid in developing UIs for TV display which may cut off some amount of the image border.



Guides

You can create guides in DFGUI by dragging from the ruler area onto the main design area. You can delete them by dragging them back into the ruler area, or by right clicking one and choosing Guide → Delete Guide. You can also create new guides by right clicking the UI root and choosing Guide → New Vertical Guide or New Horizontal Guide. Finally, guides can be edited by right clicking and choosing Guide → Edit Guide.
Guides can be used to aid in aligning elements, and can also be snapped to while moving or resizing controls.


251px, 31 %
CTRL - Snap to every 5 pixels, SHIFT - Snap to grid, ALT - Hide info, Right click to edit