GUIMaker

A WYSIWIG UnityGUIlayout editor

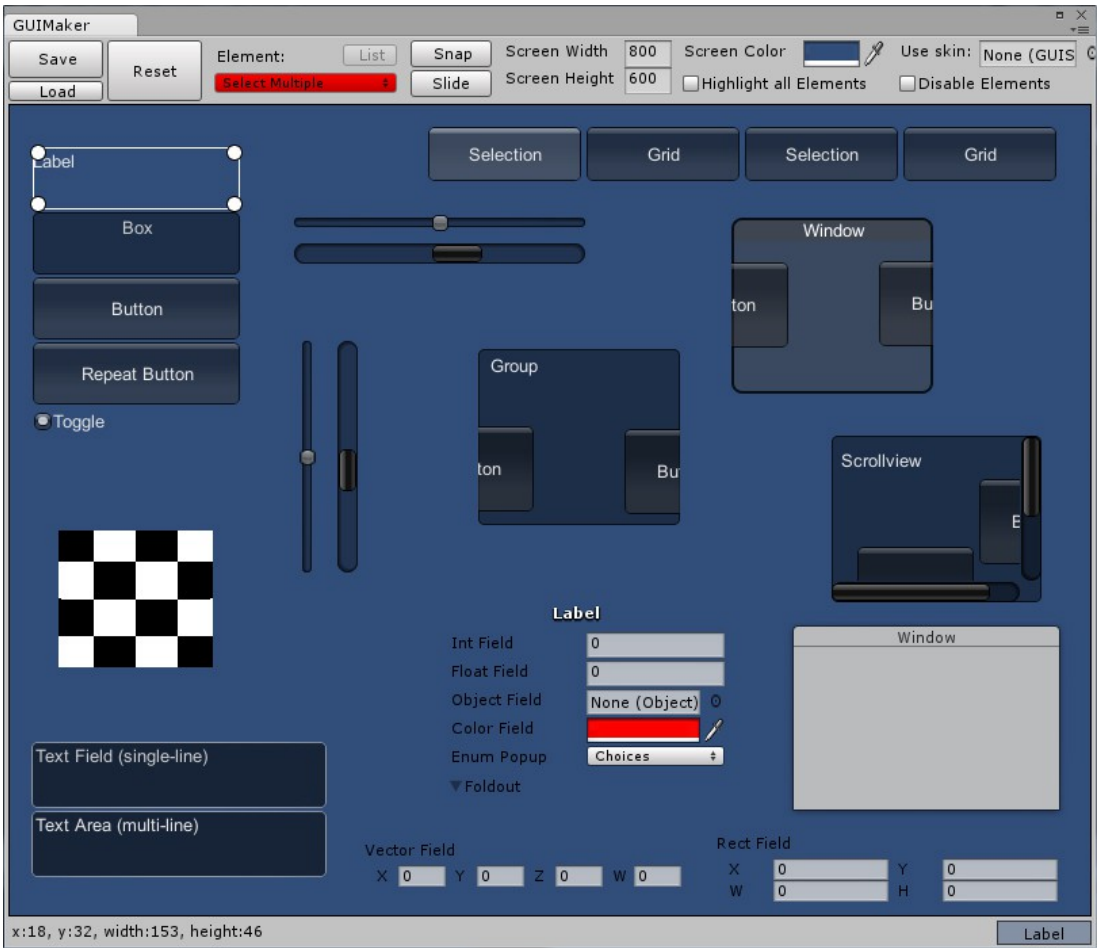
by hpjohn

Table of Contents

Introduction.

"UnityGUI allows you to create a wide variety of highly functional GUIs very quickly and easily... you can do everything at once with just a few lines of code." - Unity3D docs

GUIMaker further extends the simplicity of UnityGUI by providing an intuitive, visual design, WYSIWIG (what you see is what you get) editor that lets you rapidly position GUI elements around the screen with a simple click-and-drag interface, to do away with the manual calculation of positions and sizes, producing nicely formatted and commented script for in-game and editor window GUIs.



Main Screen.

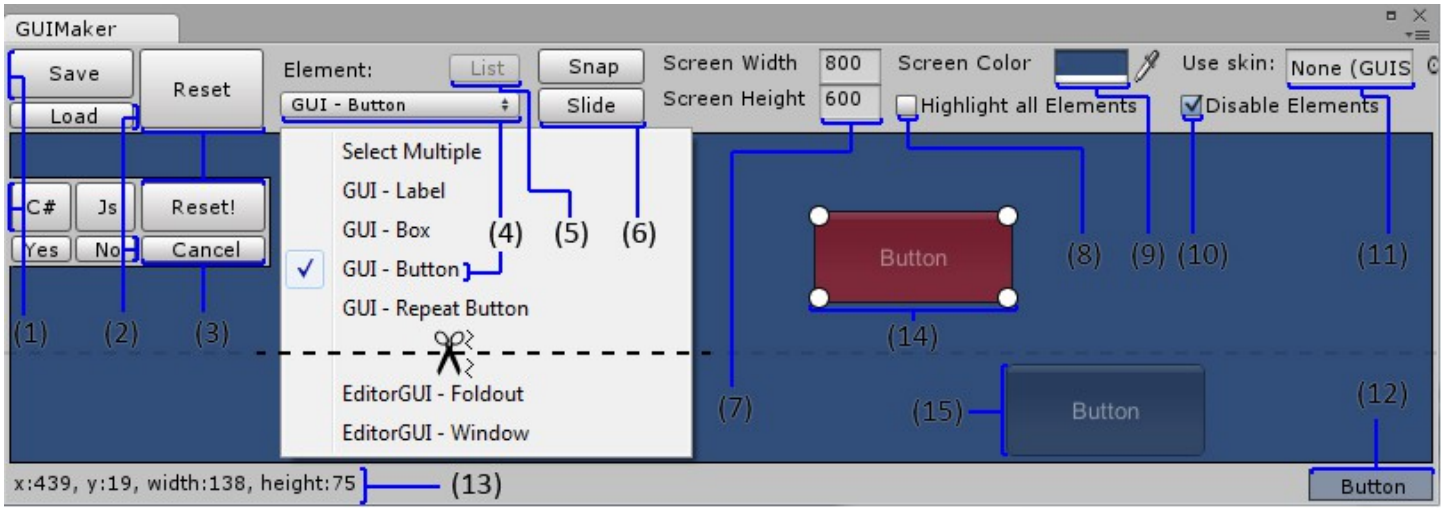


Fig 1. The main screen

(1) Save group. Clicking 'Save' shows 2 options, 'C#' and 'Js'; clicking one of these puts the script needed to generate the designed GUI onto the system clipboard (see Save/Load).

(2) Load group. Clicking 'Load' shows 2 options, 'Yes' and 'No'; clicking 'Yes' will load a layout. Clicking 'No' cancels the load (see Save/Load).

(3) Reset group. Clicking 'Reset' prompts you to confirm or cancel, and will clear all objects from the screen.

(4) Currently selected GUI element type. Clicking the button will drop-down a list of most of the common UnityGUI object types, the currently selected one is displayed (full list not shown). With an element chosen, click and drag in the main screen area to add it.

(5) Add-as-List toggle. This button can be toggled on and off to add new elements as a List-type (see Lists).

(6) Snap and Slide.

- When snap is on, objects moved and resized will snap to other nearby objects. (Hold ctrl while moving or resizing an element(s) to turn snap on temporarily)
- When slide is turned on, this mimics “Axis constrain” as you would find in other graphics applications. (Hold shift while moving or resizing an element(s) to turn slide on temporarily)
- Both can be on simultaneously.

(7) Screen width and height. Adjust this to match the screen size of your Unity project.

(8) Highlight all Elements. Toggle this on to add a highlight overlay to every element on the screen, this can be useful for finding objects that have been obscured by others, or hidden by groups/scrollviews.

(9) Screen color. Change the background color of the screen to match your project.

(10) Disable Elements. (On by default) This disables all the user elements; enabled elements like buttons interfere with the operation of the buttons in the context menu.

(11) Use skin. Drag and drop a GUISkin here from the project view to make all user elements use that skin.

(12) Shows the type of the currently selected element (see Groups).

(13) Shows the currently selected elements' Rect position.

(14) The currently selected GUI element, highlighted in red. Click and drag inside the element to move it. Click and drag the corner handles to resize it.

(15) An unselected element.

Context Menu.

Right-click to bring up the context menu. This shows several options for the current selected element(s).

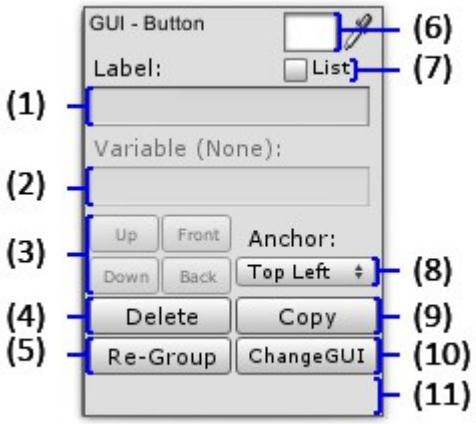


Fig 2. Context menu (for a GUI Button)

(1) Element label. For elements that have them, type the label here.

(2) Element variable. For elements that need them (eg. Toggle (boolean), SelectionGrid (selected integer)), type the variable name here.

(3) Element draw order buttons. Click the up/down/front/back buttons to move the selected element forwards or backwards in the draw order, with respect to its current group (cannot move an element higher than another that is in a higher group).

(4) Click to delete the currently selected element(s). (Press the delete key with an element(s) selected to delete)

(5) Click to change the currently selected element(s) grouping. Click once to close the context menu, then a second time on the desired group/scrollview/window/toggle/foldout element to move the currently selected elements into that group. Click an empty area of the screen to move the currently selected element(s) out of any grouping. (Hold Alt/Option with a group/scrollview/window/toggle/foldout selected when creating new elements to immediately insert the new element into that group)

(6) Choose the currently selected element(s) color.

(7) Toggle this element between List-of and no list (see Lists).

(8) Change the anchor position. Element(s) anchored to the lower-right corner will move as their containing element is resized; to the center, will remain in position relative to the center of their containing group, etc.

(9) Click to make a clone of all selected elements. (Hold Ctrl before click-dragging an element to immediately make a clone of it)

(10) Click to change the current selection into a different type of element.

(11) Some elements have extra properties in this space:

- Sliders: Min and Max values.
- SelectionGrid: Total button count and horizontal button count.
- Group and Scrollview: Fill with box (Often you may want to position a box that exactly fills a group or scrollview. Placing one manually will prevent you from selecting the group, as the borders would overlap; use this option to place a box instead).
- EditorGUI fields: Label width. By default, the Int, Float, etc fields are created with the label taking up exactly half the width of the element. Use this option to specify the label width.
- Vector field: choose between Vector2, Vector3, and Vector4.

Groups.

Groups, scrollviews, windows, toggles and foldouts have special behaviors.

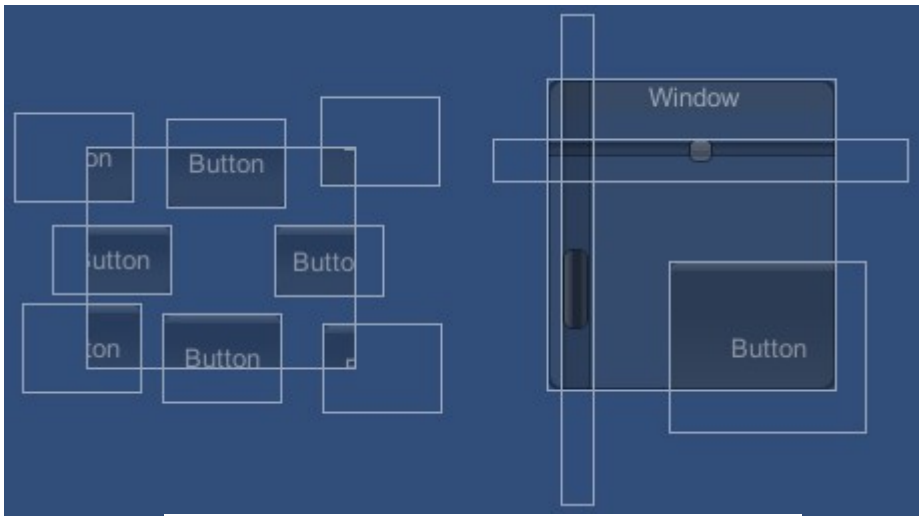


Fig 3. A Group and Window, partially obscuring buttons and sliders

Groups, scollviews and windows (collectively groups from now on) can contain other elements, so that they can be moved together, and will partially (or completely) hide any element that falls outside their area.

To add an element (eg. A button) to one of these groups, right-click the button, click 'Re-Group', then click the group. Alternatively, when first creating a button, select the desired group, hold Alt (Option) and then click-drag to create the new element already inside the group.

Fig 4. The scrollview 'internal area' handle

Scrollviews have an additional control handle. When selected, you will see a green border drawn inside the scrollview element. This is to define the 'internal area' of the scrollview (called viewRect in the Unity documentaion). If the internal area of a scrollview is larger than the visible area, UnityGUI adds scrollbars to the element as needed.

Toggles and foldouts function slightly differently, although they can 'contain' other elements, elements put 'into' a toggle do not move as a group when the toggle element is moved, and they are not partially hidden by them; instead, anything assigned to a toggle is assumed to have it's visiblity controlled by that toggle (enable all elements [Fig 1. (10)] and click on the toggle to hide/unhide all elements associated with it). This functionality is reflected in the final script that GUIMaker produces, with an if statement as follows:

```
if( someToggleBoolean ) { ...<other GUI elements>... }
```

Groups, windows and toggles can all be nested inside each other, to produce neat layouts. Note: although it is possible to create a window inside another window using GUIMaker, UnityGUI is incapable of nesting windows in this fashion. If you do attempt to create a window-in-window, the final script created by GUIMaker will replace the second window with a regular group, and draw a box that looks like a window, without the functionality of one:

```
void WindowFunction ( int windowID ) {
//Note: You tried to start a window inside another window here, this is impossible, instead, you get a group with a skin
    GUI.BeginGroup( new Rect(50, 42, 86, 77) );
        GUI.Box( new Rect( 0, 0, 86, 77), "Window", "Window" );
    GUI.EndGroup();
    GUI.DragWindow();
}
```

When you have elements nested in others, the bottom right of the window [Fig 1. (12)] displays a 'breadcrumb' trail to indicate which element is the container for the currently selected one. Clicking one of the buttons in the trail [Fig 5.] will select that element.

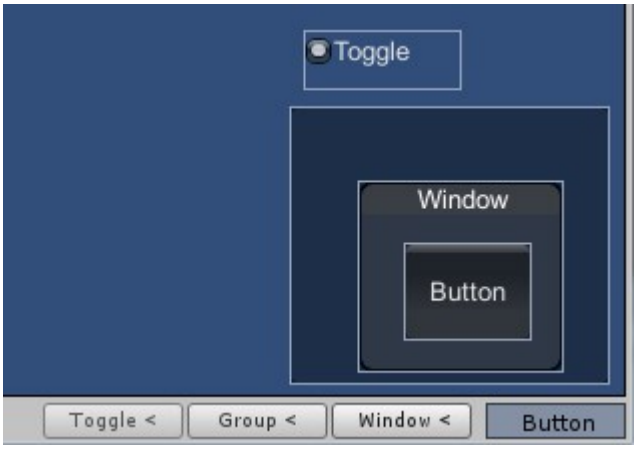


Fig 5. Nested elements and their containers

Lists.

In a project you may find the need to display a GUI element for every item in a collection, eg. You have a list of players, and want to show a label on-screen for each name. This involves iterating over the collection, and calling a GUI method at each step, and offsetting each time by a certain amount. GUIMaker can do this for you.

If an element is indicated as a List (either turn on [Fig 1. (5)] when creating an element, or turn it on later in the context menu [Fig 2. (7)]) the editor shows several copies of that element at incremental offsets. When the 'main' element is selected, an additional control handle is shown, representing the offset for each subsequent element. Click and drag this handle to adjust it. (Hold Ctrl and use the arrow keys to nudge)

When GUIMaker outputs the script, it will make a few assumptions regarding the source list, which you will have to replace to fit your requirements. For example (creating a list of buttons) :

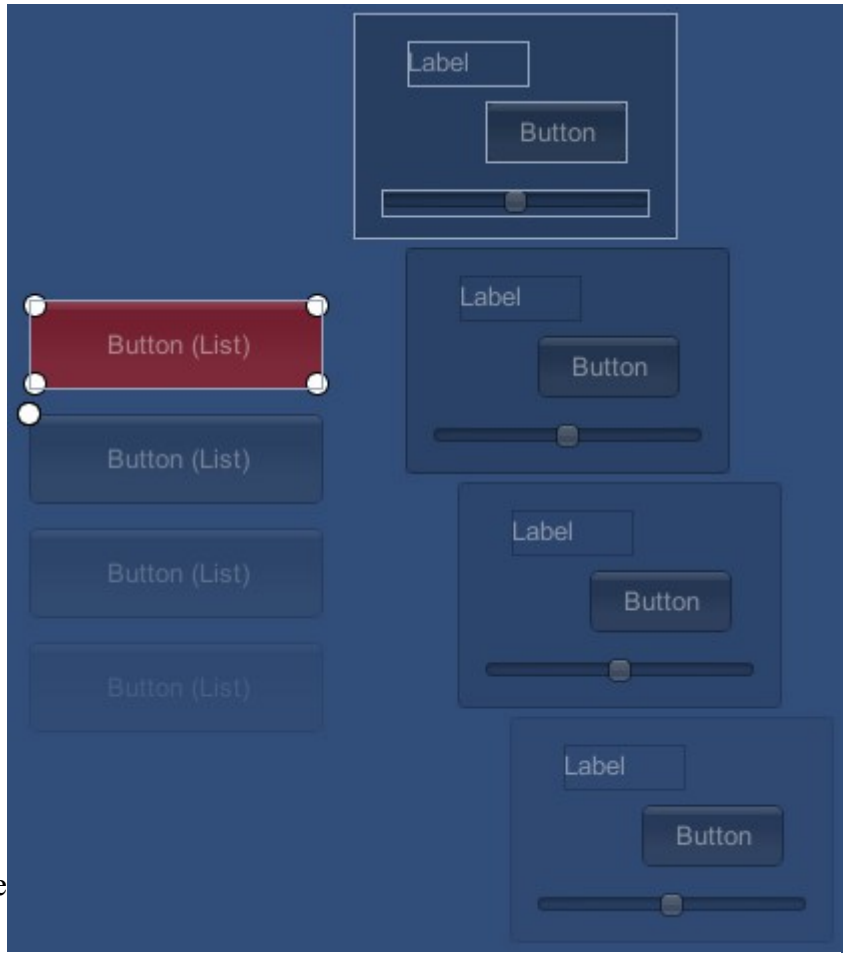


Fig 6. List of buttons, List of groups (with contents)

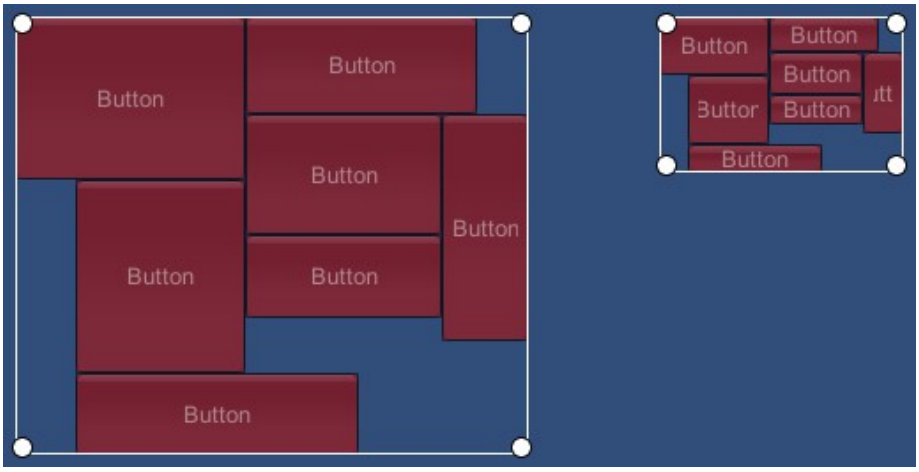
```
//someList0 is used in a loop somewhere; populate it, or replace 'someList0' with your own collection (and correct Type)
List<Object> someList0 = new List<Object>();

void OnGUI () {
    //someList0 needs to be an existing collection eg. List<type> or type[], You may need to change .Count to .Length (for []'s)
    for ( int list0Index = 0; list0Index < someList0.Count; list0Index++ ) {
        if ( GUI.Button( new Rect(222 + 2 * list0Index, 188 + 57 * list0Index, 147, 45), "Button (List)" ) ) {
            //DoStuff();
        }
    }
}
```

You should either populate someListX with your data, or find and replace any instance of 'someListX' with the name of your collection. You will probably also want to change the label of the elements inside the for-loop with something from the collection, like 'someListX[i]'.

Groups can become lists, and any elements placed inside that group (see Groups) will be duplicated for each group in the list. [Fig 6.] Listed groups can contain other listed elements, etc.

Selection Tool.

The first choice in the element drop-down list [Fig 1. (4)] is 'Select Multiple'. While this is selected, click-dragging the mouse lets you select more than 1 element at once.



With a selection of more than one element, you can move them all together (Snap & Slide will work), resize them all relative to each other, and delete and copy selections.

Several of the options in the context menu are disabled with a selection.

Holding shift while left clicking appends the selection.

Note: Upon resizing a selection, when you mouse-up, each elements' size and position will be rounded to the nearest integer, this is the default behaviour to stop long floating points values in the Rect construction, but might result in singel-pixel overlaps or gaps between elements. To disable this behaviour, edit the GUIMaker script, at the start of the script is a boolean toggle “IntRounding = true”, changing to false will disable integer rounding on resize.

Note: There is an issue with the following nesting structure: an element 'inside' a Toggle/foldout, inside a group (Group > Toggle > Elements). Due to a Toggles' contain-but-don't-contain behaviour, performing a move with all of those elements selected will incorrectly move the toggle-nested elements. Solution: Just move the Group.

Save/Load.

Clicking the Save button [Fig 1. (1)] presents you with the option for C# script, or Javascript (Unityscript). Choosing one of them will dump the script required to create the GUI onto the system clipboard, then you can simply paste into a script file. For most simple cases, this will be sufficient to get your GUI working in-game, or in an Editor window. However, some elements need a little script work to finish them off, these are often commented to bring your attention to them.

Any elements that take a variable, but where no variable was given [Fig 2. (2)] will be output with a generic 'someVariable'. Blank labels are left blank. Any window elements will have their own window function, even if their contents are identical (you can easily make any GUIWindow call the same function – see the Unity docs).

A short example might look something like this:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

//Remember to change SCRIPTNAME to the name of the script
public class SCRIPTNAME : MonoBehaviour {

//Required Variables:
int someSelectionGridInteger0 = 0;
//This array is the labels for each button in a selection grid, replace the x's with your own stuff, or replace the variable
with another collection of strings
string[] someStringArray0 = new string[4] { "x", "x", "x", "x" };
float someHSliderFloat0 = 0;


void OnGUI () {
    someSelectionGridInteger0 = GUI.SelectionGrid( new Rect(31, 19, 385, 27), someSelectionGridInteger0, someStringArray0, 4 );

    if ( GUI.Button( new Rect(48, 69, 59, 29), "Button" ) ) {
        //DoStuff();
    }

    if ( GUI.Button( new Rect(177, 73, 60, 25), "Button" ) ) {
        //DoStuff();
    }

    GUI.BeginGroup( new Rect(35, 116, 181, 135) );
    GUI.Box( new Rect( 0, 0, 181, 135), "" );
        someHSliderFloat0 = GUI.HorizontalSlider( new Rect(10, 11, 154, 22), someHSliderFloat0, 0f, 1f);

        GUI.Label( new Rect(21, 49, 112, 35), "Label" );

    GUI.EndGroup();

}
}

/*GUIMaker by hpjohn. Savecode:
...[snip]...
GUIMaker*/
```

At the end of the generated script is a GUIMaker code, which can be copied and used to load [Fig 1. (2)] a generated GUI back into the editor. Make sure to copy from the first 'GUIMaker' to the last, so the editor gets what it needs to load correctly. Note: Of course, any manual adjustments to the main body of the script will not be reflected in the savecode.

Note: GUIMaker will include everything to make a complete script file, from the includes, public class SCRIPTNAME : Monobehaviour, and (if you are making an editor window GUI) the editorWindow and MenuItem code. To disable the inclusion of these 'headers', edit the GUIMaker script, at the start of the script is a boolean toggle “IncludeHeaders = true”, change this to false to omit this extra script.