Simple Color Picker

Version 0.5.0 : By George Corkery

Thank you for downloading *Simple Color Picker*.

This asset is designed to make the implementation of a color picker quick and easy with only a few steps necessary to master.

If you have any suggestions or queries, please contact me through the GameMaker Marketplace and I will be sure to respond as soon as I can.

Please be aware that changes ***may*** be made to this asset (all parts, including this file) over time with features being added, removed or changed to ensure that the asset is made to the highest standard possible. An example, is the need for varied slider and box component options, these will be updated and provided in a free update when they're ready.

A change log is provided on the final page of this file, as well as on the main Marketplace page, and any major changes listed and updated with as much detail as possible.

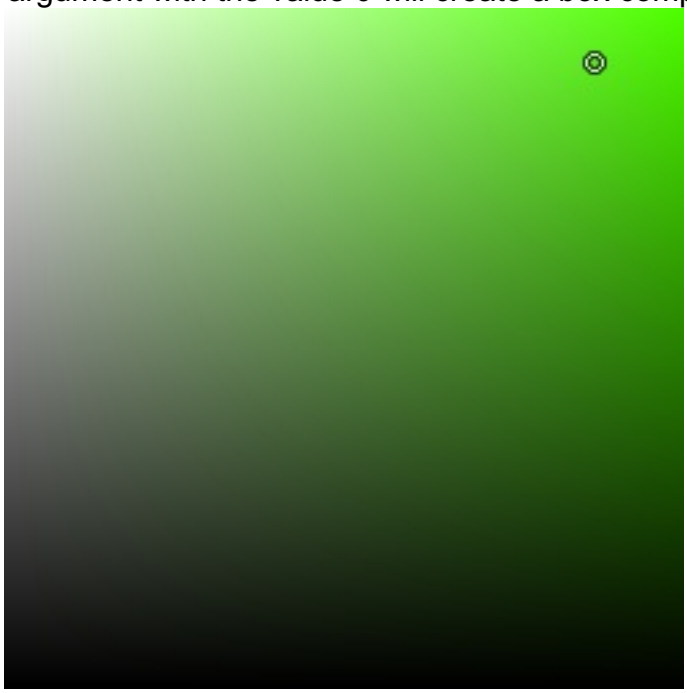
You may use this asset in person, or commercial projects, but you may not redistribute the asset. If you wish to share the asset with others, please provide them the link to the asset on the marketplace.

Functions

scp_add(*type, *x, *y);

This function takes a mandatory argument *type* and two *optional arguments *x* and *y*. The type argument specifies the type of component to add.

A type argument with the value 0 will create a box component.



The box component

A type argument with the value 1 will create a slider component.

The slider component

With the optional *x* and *y arguments, you can specify a specific location to place the component.*

scp_set(*id, hue, sat, val*);

This component takes four arguments. The *id* argument requires a variable containing the id of an scp component, while the *hue*, *sat* and *val* arguments are the hue, saturation, and value (luminosity) you wish that component to have applied to it.

scp_set_hue(*id, hue*); scp_set_sat(*id, sat*); scp_set_val(*id, val*);

These three components, *scp_set_hue*, *scp_set_sat*, and *scp_set_val* all fulfill the same role as the *scp_set* argument, but gives you the ability to set the hue, saturation and value components indivually.

scp_get_color(*id*);

This function will return the component supplied through the *id* argument's current color.

How does it work, and how do I use it?

Provided with the extension is an example object, *scp_object_example*. Looking through this, and reading the comments, should be enough to get a grasp on how the extension works, however, I will also explain here, what is happening, so you can better understand, if you are having trouble working out how it's doing things.

Looking at the example object

Create event

The very first thing we need to do, is to create the components we need. We will do this within the objects create event.

We want a single box, and a single slider. We create both of these components by calling the *scp_add* function and providing the relevant arguments. We want to place our box at the coordinates (25,25), so we call,

cp_box = scp_add(0, 25, 25);

we pass in 0 for the first argument as this defines the *type* of the scp component. 0 being a color box, 1 being a hue slider.

Now, we have the box component and its id stored in the cp_box variable. Repeating this, we create a slider object, which follows an almost identical procedure, but we pass in 1 as the first argument, as this will set it to be a slider object. We also want to place the slider 275 pixels to the right of the color box, so we end up with the following,

cp_slider = scp_add(1, 300, 25);

Now, we'll set both of the components to be a solid red color, by using the *scp_set* function like so,

```
scp_set(cp_box, 255, 255, 255);
scp_set(cp_slider, 255, 255, 255);
```

scp_set takes an initial argument of a reference to the object you wish to change, and then the hue, saturation, and luminosity respectively. Here, we define the hue, sat and val arguments as 255, as it sets each to it's maximum value, meaning it's at it's highest hue (red), highest saturation, and highest luminosity, which gives us a nice bright red color.

Step event

If you were to run the project now, you could play about with the slider, and the box, however, when changing the hue of the slider, it wouldn't change the box to match. You need to link them together. This is a very simple process, requiring you to only set the hue of the box, to match the slider. We do this every step, so that it updates

appropriately.

scp_set_hue(cp_box, cp_slider.hue);

This is a very simple line. We set the component, *cp_box*'s hue, equal to that of the value of *cp_slider.hue*

As you can see, we can get the sliders hue value by simply using the reference operator to access the relevant property.

Just as we can access the hue property, we can also access the following, should we wish.

- x
- y
- canvas_width - The width of the surface used to draw the component.
- canvas_height - The height of the surface used to draw the component.
- hue
- sat
- val

With those steps taken, we have completed all that is necessary to create and link the components together, ready to be used, however…

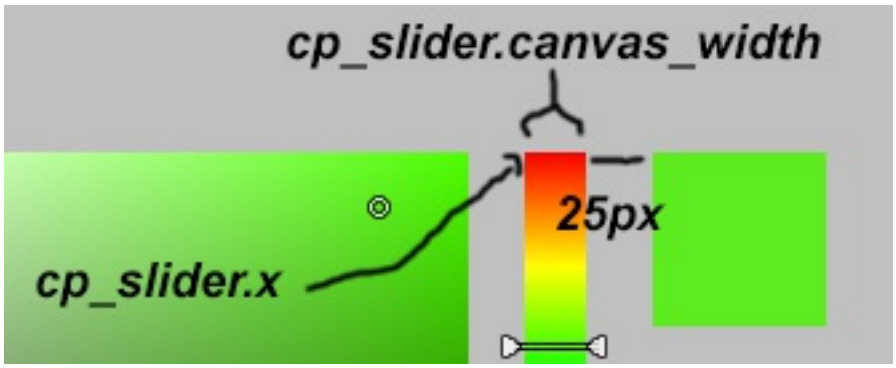
Draw event

We want a way to display, to the user, the currently selected color. This is where the *scp_get_color* function is to be used.

```
var c = scp_get_color(cp_box);
draw_rectangle_colour(cp_slider.x+cp_slider.canvas_width + 25,25,cp_slider.x+cp_slider.canvas_width + 89,89,c,c,c,c,0);
```

Here, we get the color of the box, and store It in the temporary variable *c*. We then draw a rectangle that is 64 x 64 pixels in side, just to the right of the slider.

We're getting the x position of the slider, getting the components visible width, and then adding 25 pixels. From this point, we draw the rectangle so that it is perfectly placed on the right side of the slider.


cp_slider.canvas_width
25px
cp_slider.x

- Version 0.5.0
    - Released asset. \o/