Dynamic Control

Plugin for Indigo to create Dynamically Changing control images

Index

Dynamic Control Overview

The Dynamic Control plugin provides a means to create more dynamic buttons and images on control pages. You define a device that has image properties and you can adjust those properties by calling certain actions. These images can then be used as refreshing images on control pages (The images are currently only squares - future version may allow other shapes).

Example Uses:

Initial Setup

Installation

Dynamic Control requires the Python Image Library module to be installed (Pillow). This module is not included in the base install of Python and must be manually installed.

To install run the following from the terminal:

```
sudo pip install Pillow
```

If you have issues with the above command you may need to run:

```
sudo -H pip install Pillow
```

Once Pillow is installed, download the Dynamic Control plugin found here: https://github.com/bkmar1192/DynamicControl/. Then follow the instructions for installing a plugin in Indigo.

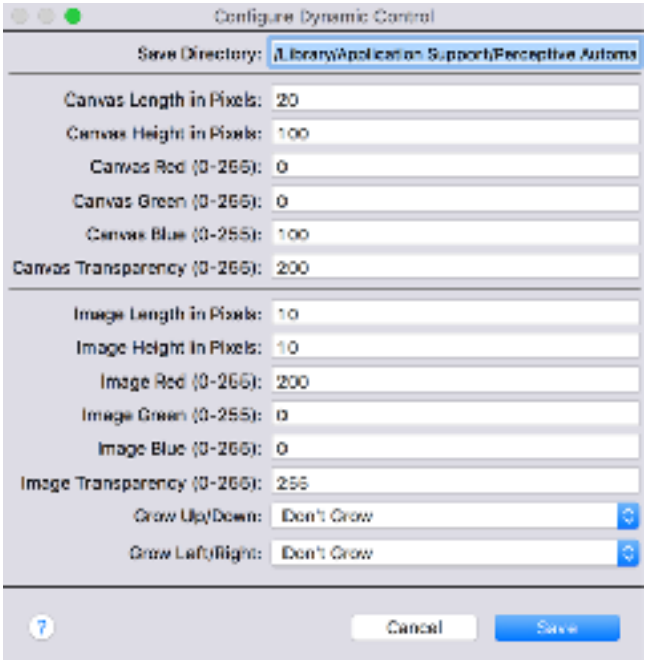
Updating

The Dynamic Control plugin is stored on GitHub and can be updated through the plugin menu items listed below.

Check for Updates: Checks for new versions of the Dynamic Control plugin on GitHub.

Update: Updates the plugin to the most recent version.

Device Configuration



Save Directory: Directory where the image files will be saved.

There is one section for the canvas and one for the image. The canvas is the background and image is the foreground image. Each section has the following settings (all of these settings can be updated through actions):

Length/Height in Pixels: The initial dimensions of the canvas or image

Red/Green/Blue: The initial color of the image or canvas

Transparency: The initial level of transparency of the image or canvas (0 is fully transparent)

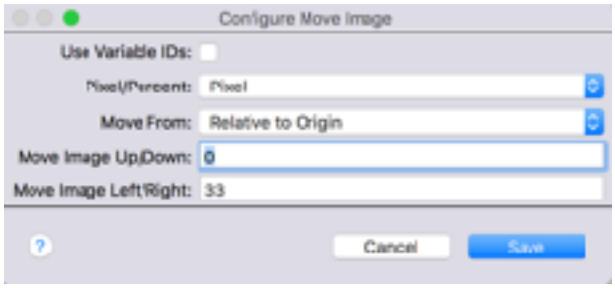
There are also two other settings for the image:

Grow Up/Down or Grow Left/Right. These control if the image keeps its original dimensions when it moves or if it is "locked" to its starting position.

> E*xample*: A scroll bar would use *Don't Grow* to keep a single square moving up and down the canvas. A temperature gauge would use the *Grow Up/Down* to display a growing bar.

Actions

Move Image



Use Variable IDs: Checking this flag to allow the Move Image Up/Down and Move Image Left/Right fields to use variables versus hard coded numbers.

Pixel/Percent: Specify if the changes in the fields are a percent change or a pixel change.

> *Example*: Assuming a canvas that is 200 pixels wide - setting the action to Percent and entering 25 in the Move Image Left/Right field will move the image 50 pixels to the left. Using -25 will move the image to the right 50 pixels.
>
> Alternatively setting the action to Pixel and entering 25 in the Move Image Left/Right field will move the image 25 pixels to the left. Using -25 will move the image 25 pixels to the right.

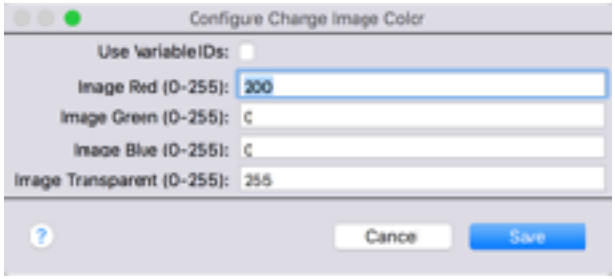
Move From: Move from the Origin (0,0) or move from the current location.

> *Example*: Assuming a canvas that is 100 pixels wide - setting to Pixels and Move From Relative to Origin and entering 25 pixels in the Move Image Left/Right will move the image 25 pixels. Running it again would keep it at 25 pixels.
>
> Alternatively setting to Pixels and Move From Relative to Current Location and entering 25 pixels in the Move Image Left/Right will move the image 25 pixels. Running it again would move it to the 50 pixel location.

Move Image Up/Down Left/Right: The number of pixels or the percent of movement within the canvas to move the image. Positive numbers will move the image Up/Right, negative numbers will move the image Down/Left.

If the Use Variable IDs is selected enter a variable ID into this field. The value in that variable will be used in as the pixel or percent value. If a 0 is entered then this field will not get updated.

Change Image/Canvas Color



Use Variable IDs*:* Checking this flag will allow the color fields to use variables versus hard coded values.

Red/Green/Blue/Transparent: Set the color and transparency of the image or canvas (0 is fully transparent).

If the Use Variable IDs is selected enter a variable ID into these fields. The value in that variable will be used in as the pixel or percent value. If a 0 is entered then this field will not get updated.

Appendix
Examples

Volume Control

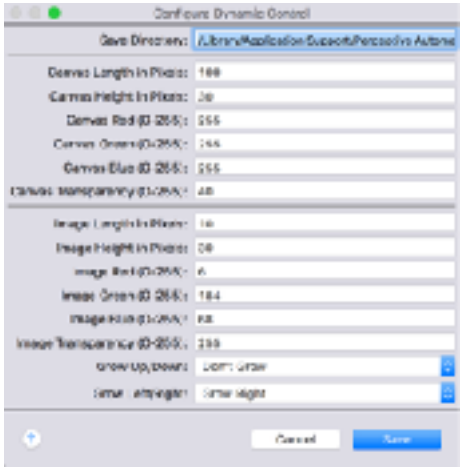
Uses a transparent image on top of a Dynamic Control. Allows a 0 to 100 volume visualization without having to create 100 different images.

Setup:

Create a transparent image that will lay on top of the dynamic control image. The non transparent portion color matches to the control page background.

The + and - are part of the overlay image. There is another transparent button on top of both these that run specific actions.

Device Setup:

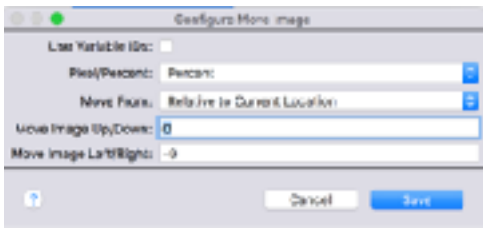


The canvas is the same dimensions as the transparent overlay image. The color of the canvas is white with a 40% transparency. This creates a lighter background where volume bar is not being displayed.

The height of the image is equal to the height of the canvas and the length starts at 10%. The color is equal the green background and there is no transparency.

The height of the control will stay the same so that is left at Don't Grow. The bar grows from left to right so the Grow Left/Right is set to Grow Right.

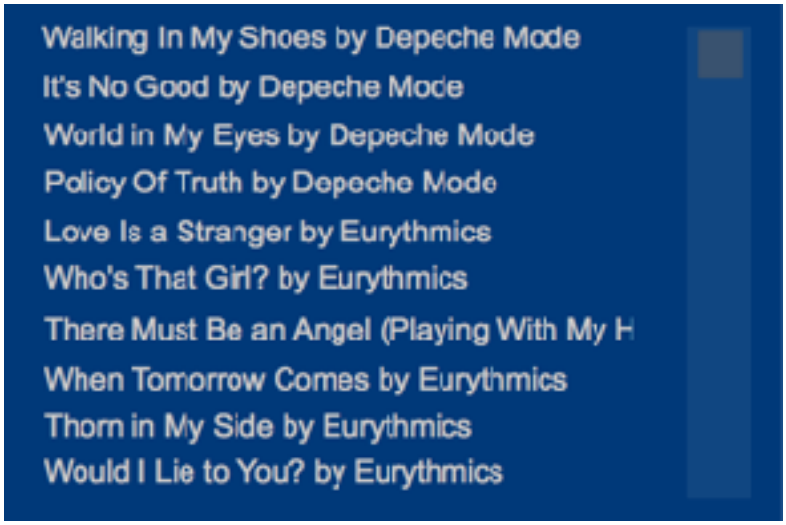
Action run when +/- is clicked:



The action adjusts the Airfoil speaker volume up by 9% then runs the Dynamic Control Move action.

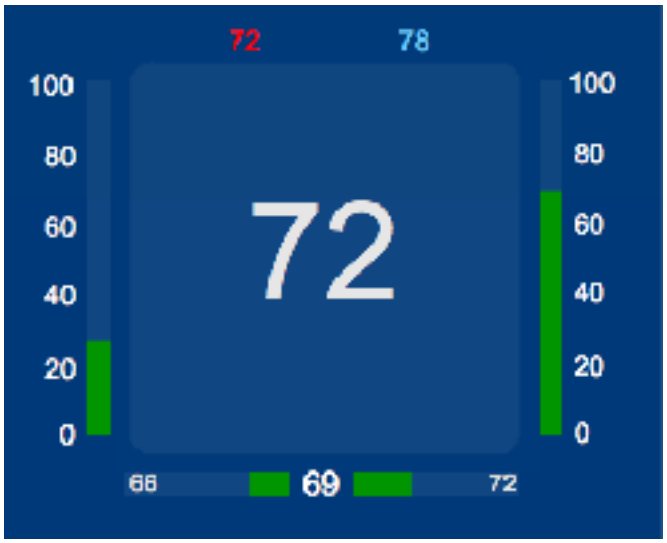
The bar will move in relation to its current position and will shrink to the left by -9%. The action for + is exactly the same except it grows by 9%.

Scrolling List



Scrolling Lists: Combined with some other python coding, this mimics a scrolling list (songs in the current playlist in this example).

Temperature Monitor



Temperature Monitor: This was designed to imitate the Nest thermostat. The square in the center will change to Red or Blue if the heater or AC turn on. The left bar is outside temperature and the right bar is inside temperature. The bars on the bottom shows the difference of the upstairs and the downstairs temperature to the desired average temperature.

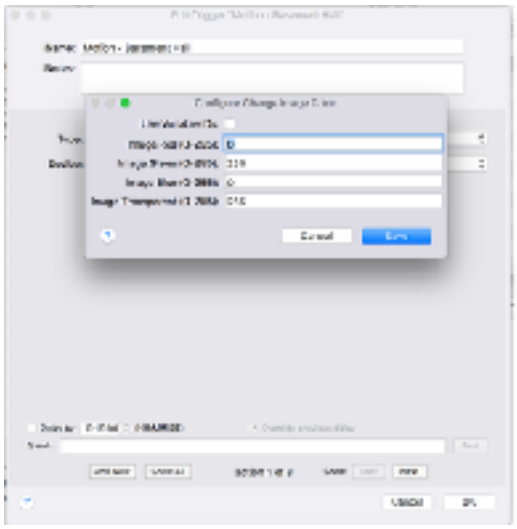
Motion Sensors: As motion is detected areas of the house are highlighted. Over a 3 minute period the motion area becomes more and more transparent until it disappears.

Device Setup: Create a simple 10x10 square box. This will create a small file size but it can be resized on the control page to what ever size you need. The canvas should be a clear background so all values can be set to 0.

The image will fill the canvas so set it to 10x10. Set the red/green/blue to the color values you want for the image color. The image will start as transparent so set Image Transparency to 0.

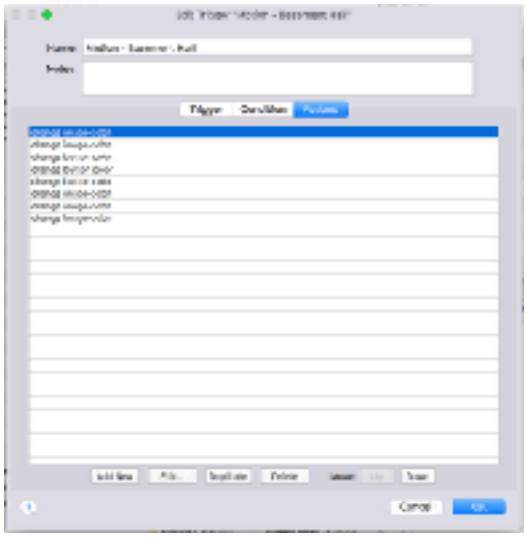
The size of the image will not change so the Grow parameters can be left to 0.

Motion Trigger: Create a trigger that runs when motion is detected



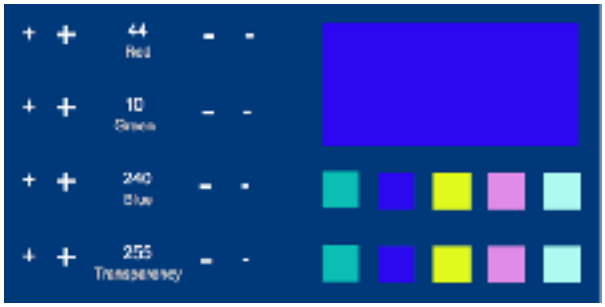
Trigger Action: Create an Action that runs the Dynamic Control - Change Image Color action. Leave the Variable check box unchecked. Set the color Red/Green/Blue to the color of the image.

Set the image to fully opaque by setting the transparency to 255. The plugin will now create a solid 10x10 solid image in the color specified.



Additional Actions: Duplicate the action 8 times. For each of the additional action, set a delay that increases the transparency of the image. Use the following settings:

2. Transparency: 220 Delay: 30 seconds
3. Transparency: 180 Delay: 1 minute
4. Transparency: 140 Delay: 1m 30s
5. Transparency: 100 Delay: 2m
6. Transparency: 70 Delay: 2m 30s
7. Transparency: 30 Delay: 3m
8. Transparency: 0 Delay: 3m 30s



Hue Color Selector: Allows the selection and saving of 5 different image colors that can be used as colors settings for a hue bulb.

Change Log