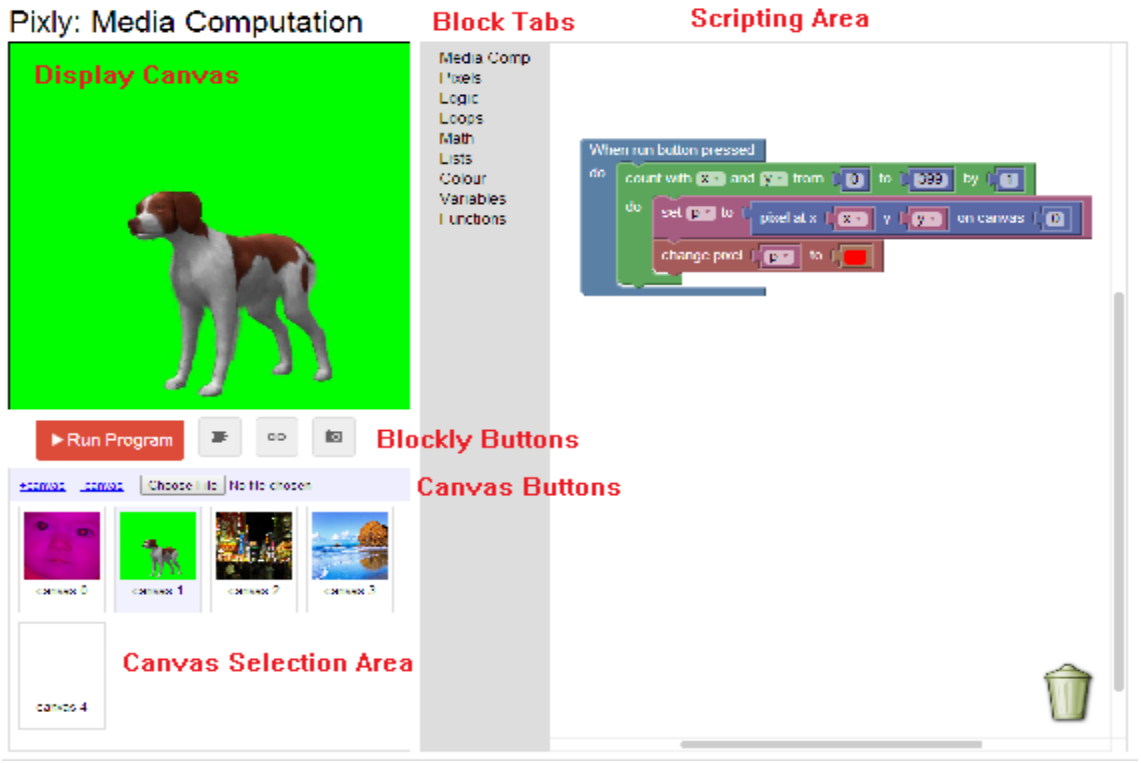
Pixly User Manual

Pixly is a custom extension to the **Blockly** web-based, graphical programming environment. Pixly allows programs to be written in a block language that can manipulate the pixels in an image. This capability supports several different kids of media computation applications like red-eye removal, green-screening (chroma key), negatives, and more.

I. Layout of the Pixly Application

When Pixly begins, you will see a large image in the **Display Canvas** on the left of the screen, with some buttons and smaller images below, and the **Scripting Area** on the right of the screen.



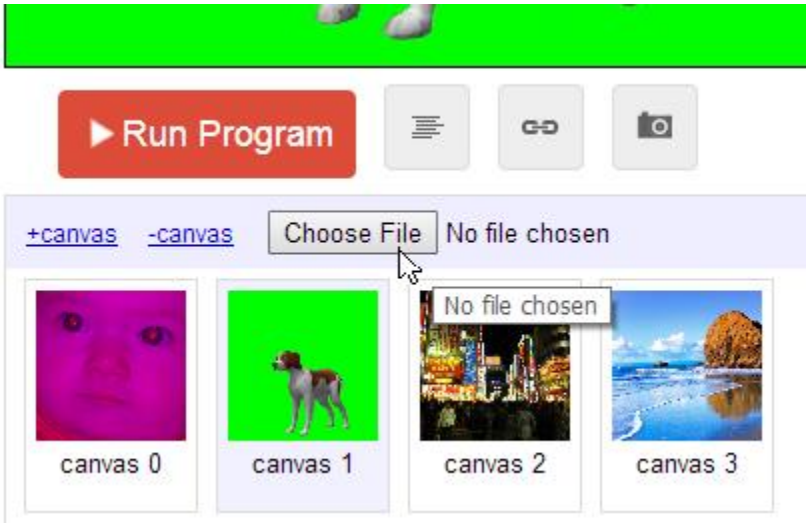
The *scripting area* on the right is where you will drag and drop your blocks to create your Pixly program. The *display canvas* on the left is where you will see the result of your image manipulation program.

II. Selecting, Adding and Removing Canvases

Pixly has multiple canvases displayed in the **Canvas Selection Area**. You can click on any one of these canvases to make that canvas displayed as the *display canvas*. Each canvas in the *canvas selection area* is numbered, and this number is the one that should be referenced in your blocks to specify which canvas(es) you are programmatically changing.

Blank canvases can be added with the **+canvas** button in the **Canvas Buttons** area, and the currently selected canvas can be hidden with the **-canvas** button.

II. Uploading User Images to the Canvas



It is possible to upload images from your own hard drive to edit.

In the *canvas buttons* area, there is a button labeled "**Choose File.**" Clicking on this button will open up a file explorer dialogue that will allow you to select the image you wish to upload onto your Pixly Canvas.

Currently, because Pixly only supports 400 px by 400 px canvases, your image will be scaled to these dimensions after loading.

WARNING: Uploading an image will overwrite whichever current canvas you have selected. *(e.g., if the green dog canvas is selected and is being displayed as the main canvas, when you upload an image, the green dog image will be drawn over with the new image, and will no longer exist!).* For this reason, it is suggested to add a new blank canvas (with the +canvas option) to upload your own personal images.

III. Exporting/Importing Blocks and Saving the Canvas

In the **Blockly Buttons** area, there are four buttons: one large, red **Run Program** button, which you press to execute the Blockly program in the *scripting area*, one blue **Export/Import Blocks** button and the **Code** and **Capture** buttons.

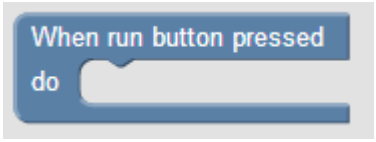
The Export/Import Blocks button will open a dialogue allowing you to copy the XML structure of the blocks in your scripting area for later use, or allowing you to paste in the XML structure of blocks you copied earlier. This allows you to quickly import block programs.

The leftmost gray button is the *code* button, and clicking it will display the Javascript code generated from the blocks that are in the *scripting area*. The rightmost button is the *capture* button, and will save the current display canvas as an image on your hard drive.

IV. Pixly Specific Blocks

This section provides a brief summary of all the blocks available in the Pixly that support the transformation of pixels within images.

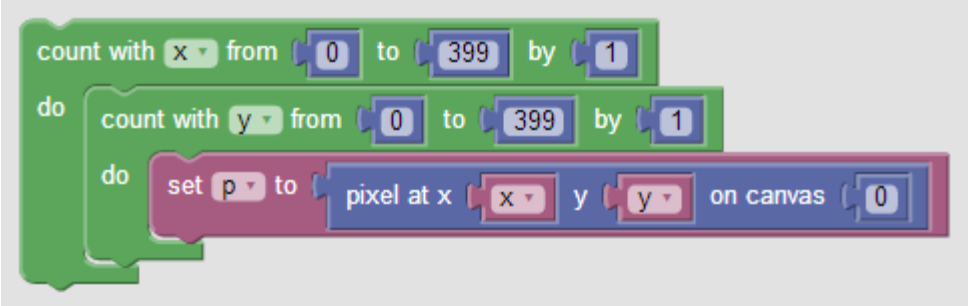
A. Blocks under the Media Comp tab



The “When run button pressed” is the first block under the Media Comp tab, and the only block that is placed on the scripting area whenever Pixly is started. Any blocks that you place within “When run..” will be executed when the run button is pressed.

Any blocks that are outside of this run button block are considered scratch and will not have any direct effect on the canvases. (**NOTE:** an exception is any **function** created outside of the *run* block. These will not be executed directly, but can be referenced from inside of the *run* block).

This block will allow you to perform a computation on each pixel in the specified canvas individually. It does so by “flattening” the canvas image such that the pixels are represented in a single linear list, going from left to right and top to bottom. Inside of this loop, the variable “p” will refer to the pixel object, and all manipulations of the pixels will likely use this variable.



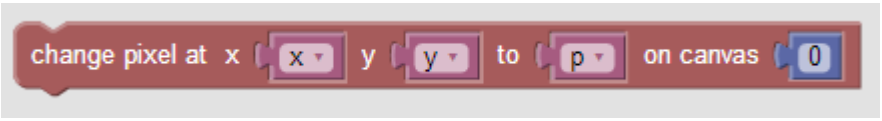
This block combination functionally acts the same as the first block sequence in the above

The difference between the two is that this block specifies a horizontal and vertical “block” of the image to manipulate, instead of just grabbing the entire pixel list. (Additionally, the block labeled “set p to ‘pixel at x, y on canvas 0” is necessary here to assign the pixel object correctly inside of the loop, whereas the “flattened image” loop automatically does this). This allows you to specify in more detail which portions of the image you want to edit (*e.g., if you only want to perform a computation on the top half of the image, you could use this block combination and change the* ***second number*** *in the* ***count with y from 0 to 399*** *block wrapper to* ***199*** *instead of 399).* This could be used to improve the Red Eye Removal example at the end of the manual to focus only on a subrange of the image on the canvas, instead of the whole image (which currently has the side effect of also changing the baby’s shirt color).

B. Blocks Under the Pixels Tab

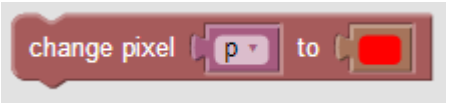
The **get pixel** block. This block is, by default, included in the *main media computation loop*.

This block will return a Pixel object from the x and y coordinates from the specified canvas. A pixel object is a representation of a pixel on a canvas image, and is composed of three values: **r**, **g**, and **b**, which correspond to the **red**, **green**, and **blue** values of the specific pixel. *(please see http://en.wikipedia.org/wiki/RGB_color_model for more information)*. These values can be retrieved from other blocks in the pixel tab.



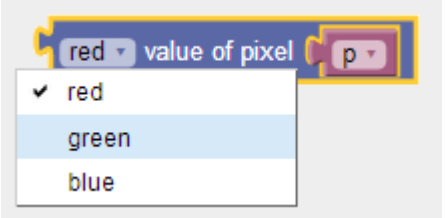
This block will change the pixel at the specified x, y coordinate on the specified canvas to specified value.

The x and y blocks can be replaced with number blocks from the Math tab to specify a specific x and y coordinate not inside of a loop. The pixel value can either be a pixel variable (for instance, **p**), or a colour block from the colour tab. This block can be used in conjunction with the main media computation loop to fully or partially copy one canvas onto another.

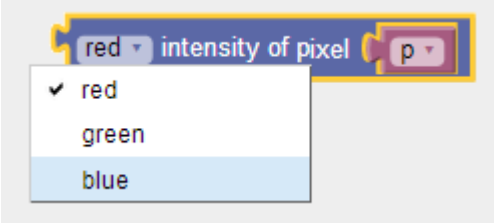


This block will change the specified pixel to the specified value. The pixel on the left needs to be a pixel variable created from the *get pixel* block (either directly, by inserting the **get pixel** block into the left slot), or indirectly, by first creating a variable (like in the *main media computation loop*). The value on the right can either be a colour block from the colour tab, or another pixel.

This block will return the colour value of the specified pixel. The color value acts the same as any colour block from Blockly's colour tab, and is represented in code as a hexadecimal string of the color.



This block will return the **red**, **green**, or **blue** *value* of a pixel. RGB values range from 0 to 255.



This block will return the **red**, **green**, or **blue** *intensity* of a pixel. Color **intensity** is calculated as follows:

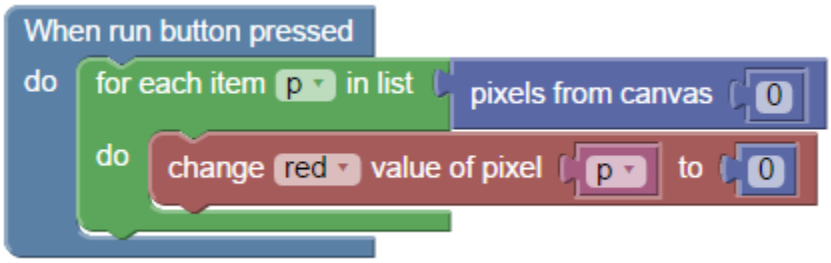
R Intensity: (red value) / ((blue value + green value) / 2)
G Intensity: (green value) / ((blue value + red value) / 2)
B Intensity: (blue value) / ((red value + green value) / 2)

This block is very helpful for red-eye removal or green-screening algorithms.

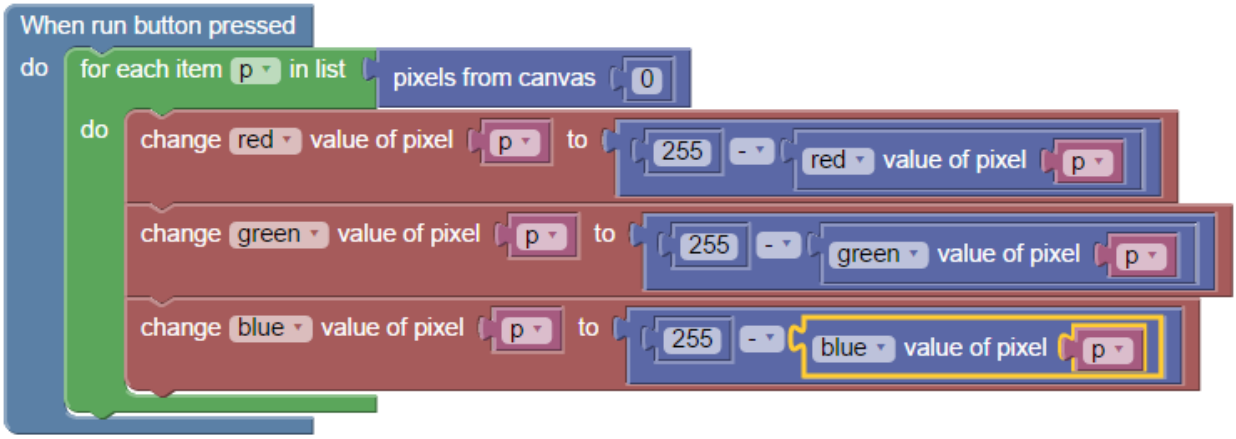
This block can be used to set the **red**, **green**, or **blue** *value* of a pixel. RGB values range from 0 to 255. For instance, if you wanted to make a picture look cerulean (get rid of all the red in the picture), you would use this block to set the red value of every pixel to 0, which would still maintain their blue and green values.

V. Code Examples

A. Remove Red

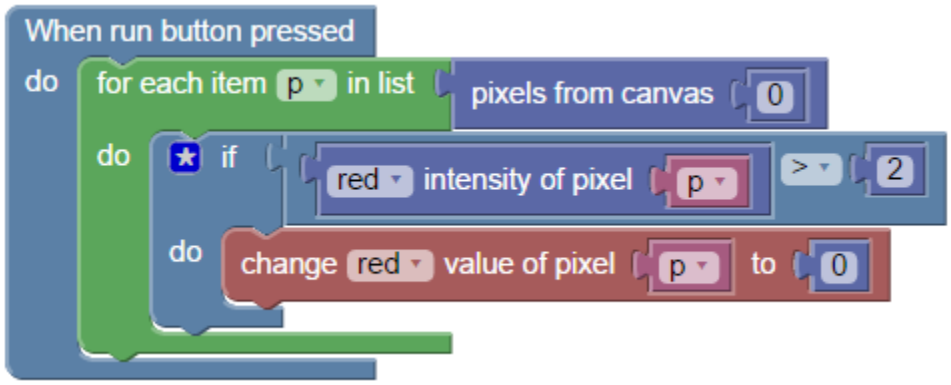


B. Negative Image

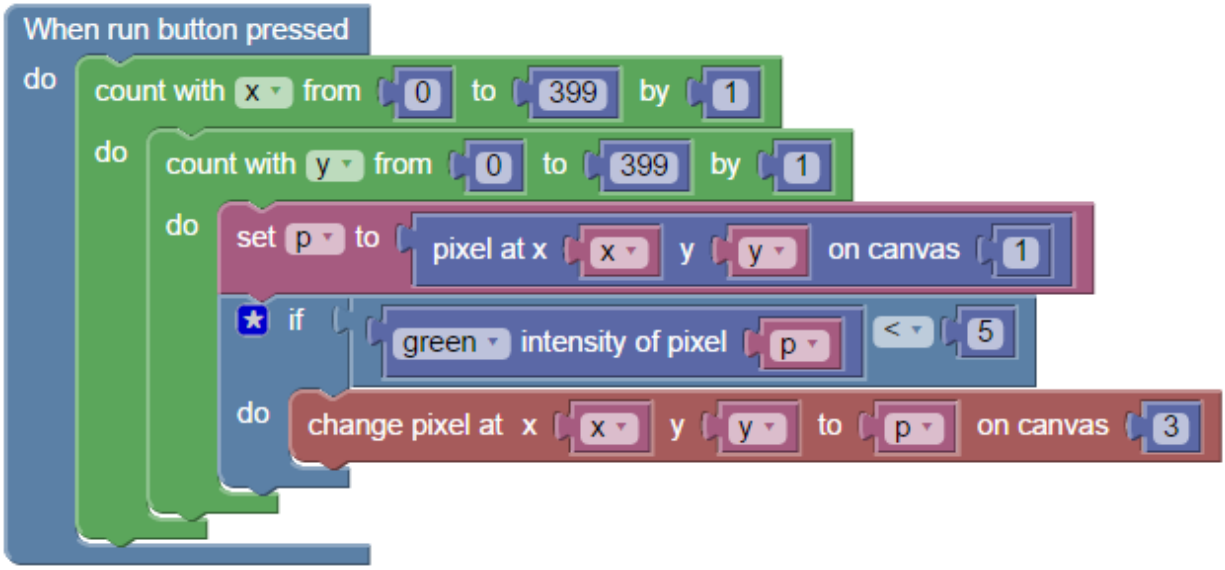


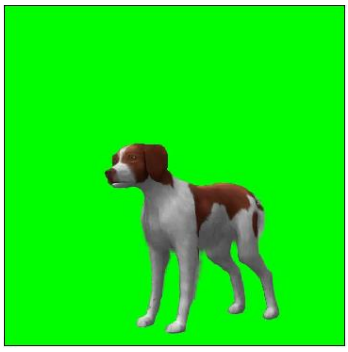
C. Red Eye Removal (Simple)

Applies to whole image (notice change in baby shirt). More detail can be added to limit the specific location of the eyes by changing the loop values.



D. Green Screen (From Canvas 1 to Canvas 3)





VI. Pixly User's Forum

To ask questions or post comments, please visit the Pixly user forum at:

http://tinyurl.com/pixly-forum