CS105 Problem Solving

Image Manipulation

Wholeness

- Image Manipulation is a great way to practice using code and make visual changes.
- Understanding Loops and if statements are both critical to make it work!
- In this lecture we will see:
  - Loading an image
  - Getting an array of pixels
  - 2 dimensional access

Loading Images

Images

- You can load an image from the server into the image window. Some of the images already there are:
  - sea.jpg
  - saucer.jpg
  - moon_and_earth.jpg
  - minions.jpg
  - forest.jpg
  - fish.jpg
  - rihno.jpg

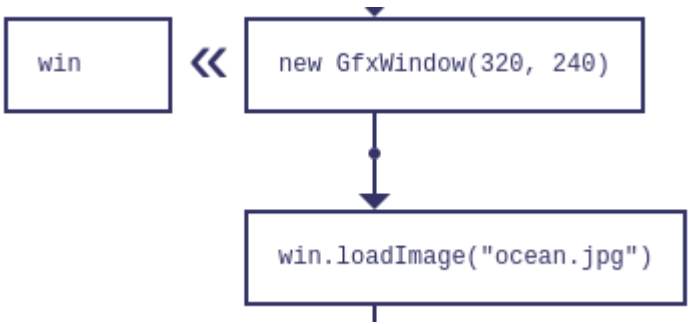
Uploading

- You can upload your own images
  - For everyone’s sanity please only upload small images!

Fish.jpg is 400x181

Inside GfxWindow

- To load an you first have to have a GfxWindow
  - Then you can use window.loadImage(“imageName”)



Example

- Demonstrate loading fish.jpg

Exercise

- Make a program that loads “forest.jpg” into a GfxWindow

Main Point 1

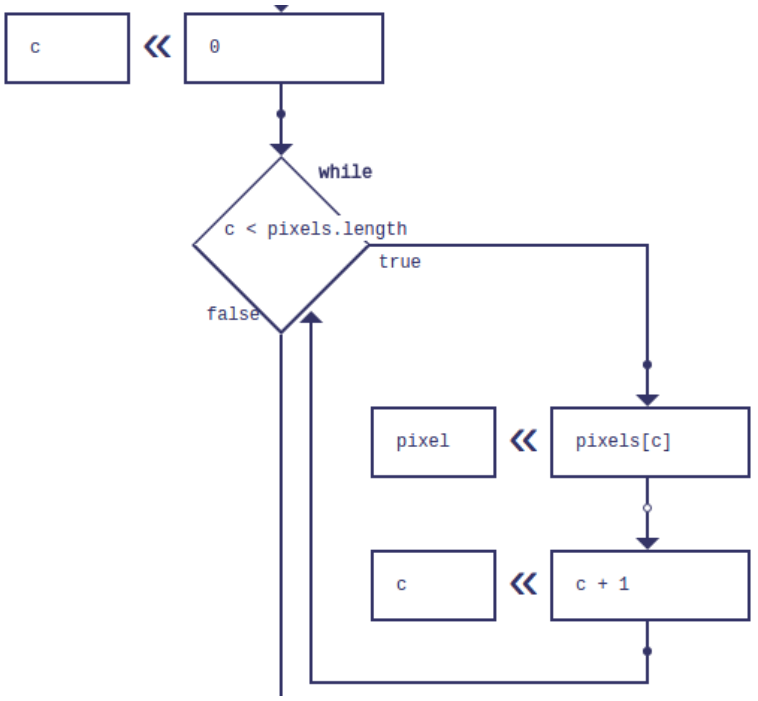
- To load an image open a graphics window and use .loadImage(filename)
- An important restriction is that you can only load images that are on the server (to which you can also upload).

A List of All the Pixels

Regardless of where they are

Window.getPixels()

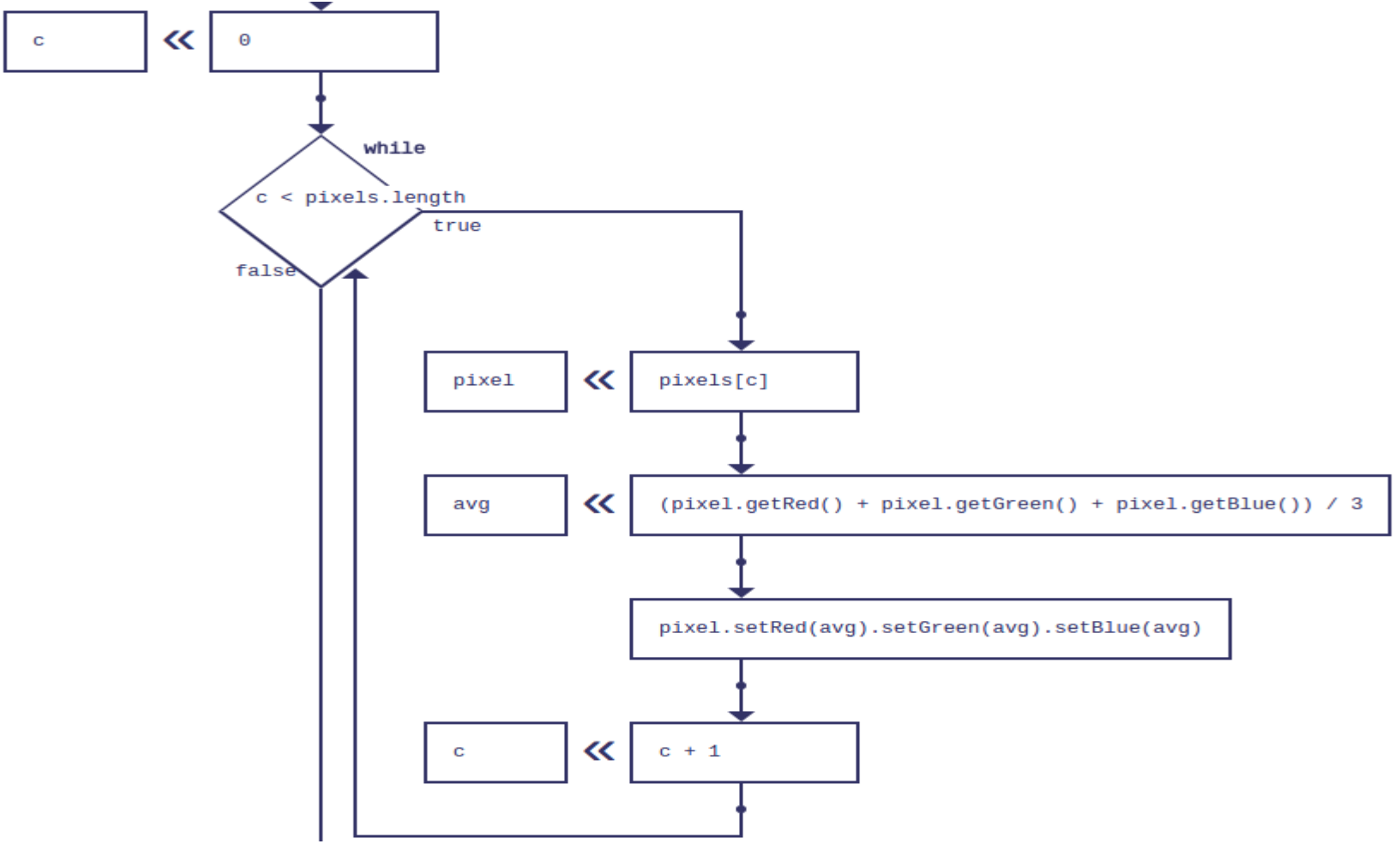
- Returns an array (a list) of pixel objects
  - Data type should be array
  - Does not remember or care where pixel is in image
  - Use a for loop to select one pixel at a time from the array
  - Array syntax uses [ ]
  - Getting the pixel at position **c**
  - Until **c** is at the end of the list (pixels.length)



Changing a Pixel

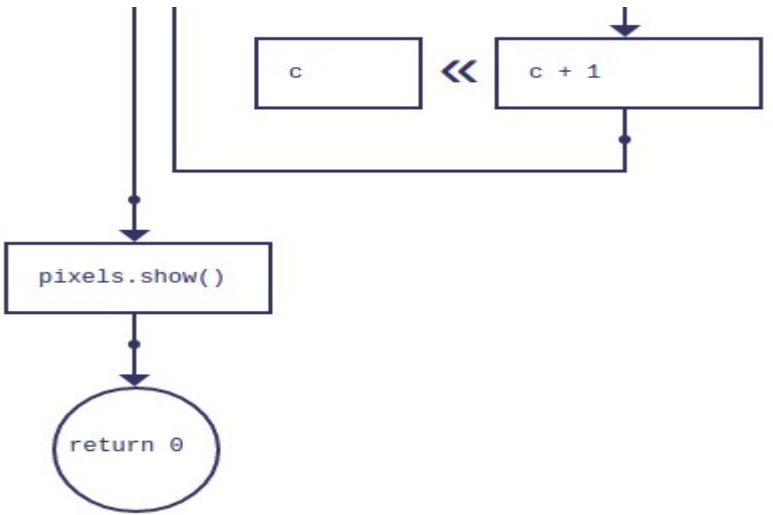
- Once you have a pixel you can get or set its red, blue, green or alpha values.
- The pixel API is:
  - GetRed()
  - GetGreen()
  - GetBlue()
  - GetAlpha()
  - setRed(val)
  - setGreen(val)
  - setBlue(val)
  - setAlpha(val)
- Where value that is get or set is between 0 and 255

Grayscale Example



Showing the Result

- Once you've made your changes (loop is done)
- You can show the changes by asking the pixel array to show itself



Before and After

Exercise

Chose (or upload) and image and then make it grayscale

Main Point 2

- window.getPixels() gives an array of pixels.
- You can get the red / green / blue values of a pixel object using .get… and set them with .set… .
- Do less and accomplish more

Two Dimensional Access

Manipulating Pixels Based on Where They Are

Window.getImageData()

- ImageData is An object that represents the pixels, and knows about their locations
  - The data type should be object

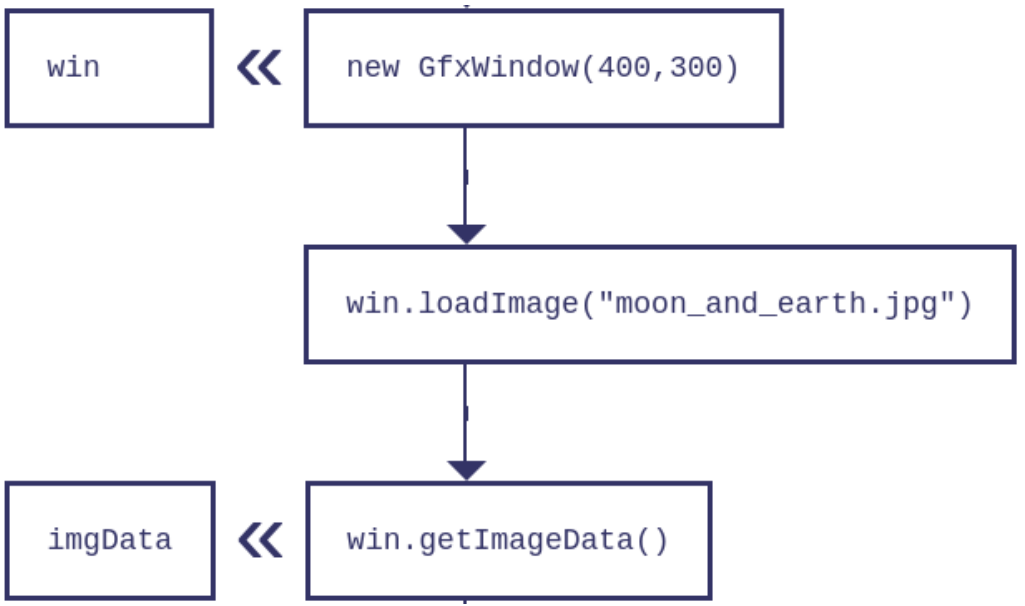
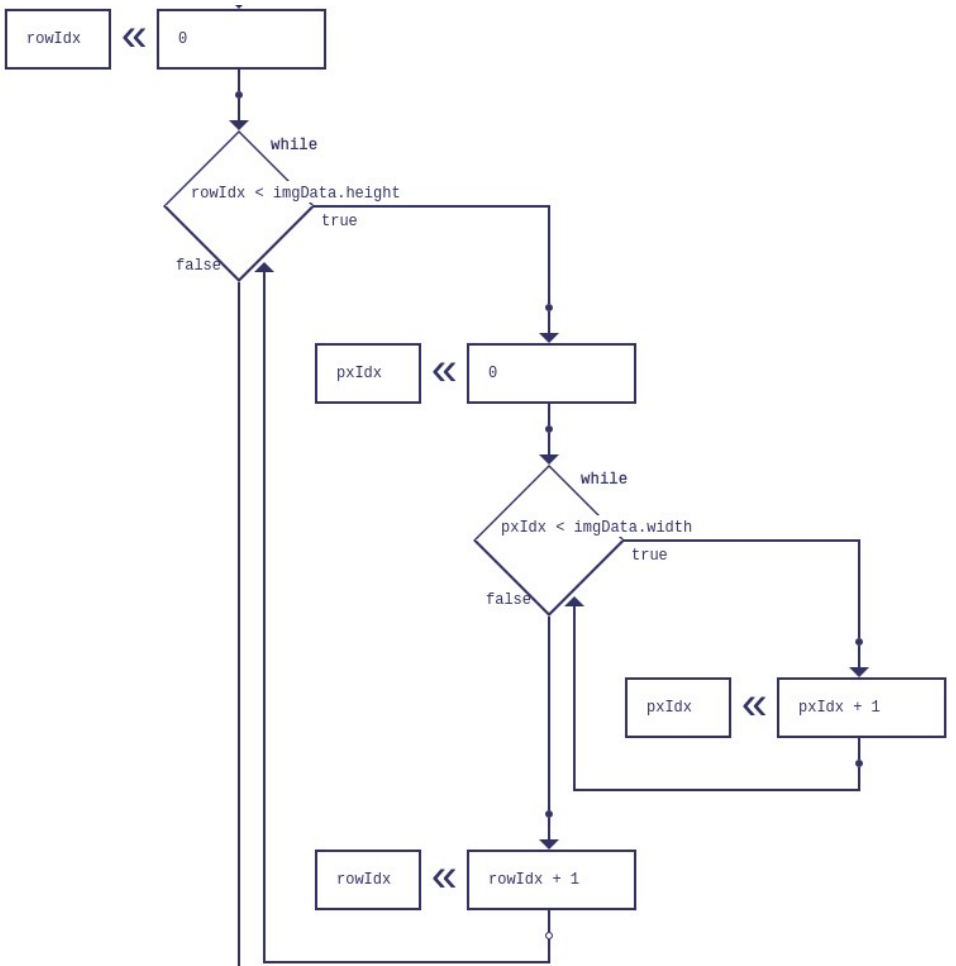


Image Data API

- ImageData objects have the following methods / data:
  - ImageData.getRedAt(x, y)
  - ImageData.getGreenAt(x, y)
  - ImageData.getBlueAt(x, y)
  - ImageData.getAlphaAt(x, y)
  - ImageData.setRedAt(x, y, val)
  - ImageData.setGreenAt(x, y, val)
  - ImageData.setBlueAt(x, y, val)
  - ImageData.setAlphaAt(x, y, val)
  - ImageData.width
  - ImageData.height
- Where the value that is get or set for red /green/ blue is between 0 and 255

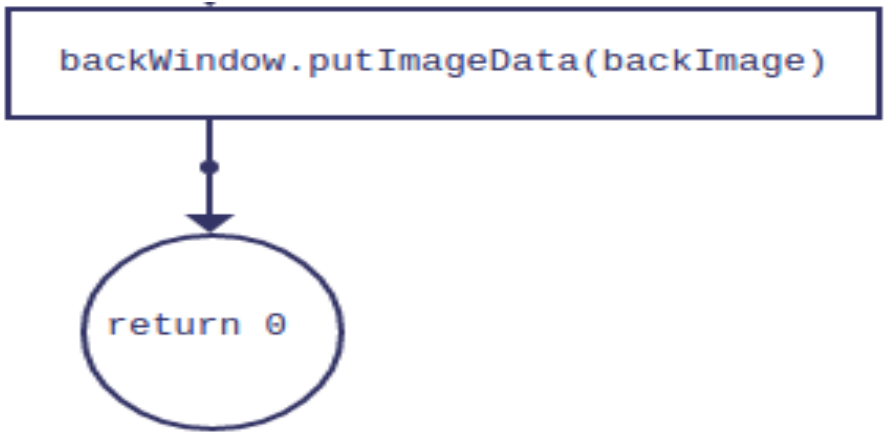
Nested Loop

- To get both x and y we need a loop in a loop.
- Where the outer loop selects a row of pixels (the y value)
- And the inner loop selects the pixel in that row (the x value)
- You don’t have to select all the pixels if don’t need!!!



Window.putImageData()

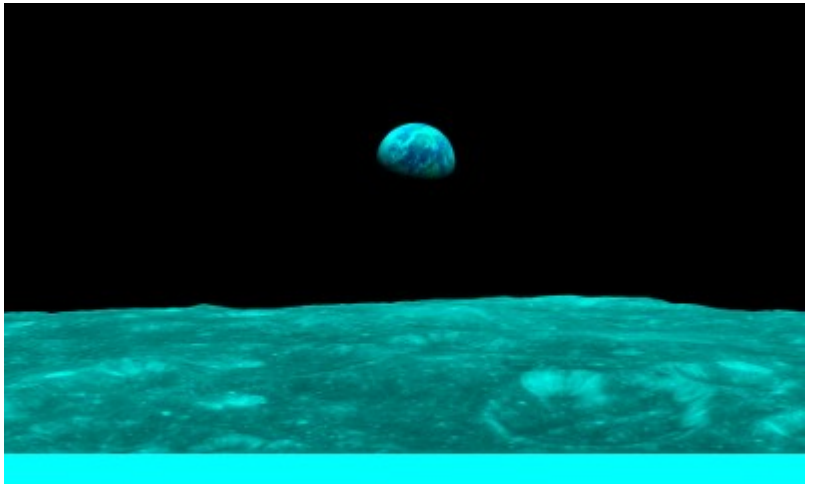
- Once you've changed imageData you can put it back into the window so as to display it
  - Window.putImageData(imageData)



Removing Red

```
function main() {
    var win; // object
    var imgData; // object
    var rowIdx; // number
    var pxIdx; // number
    win = new GfxWindow(400,300);
    win.loadImage("moon_and_earth.jpg");
    imgData = win.getImageData();
    rowIdx = 0;
    while (rowIdx < imgData.height) {
        pxIdx = 0;
        while (pxIdx < imgData.width) {
            imgData.setRedAt(pxIdx, rowIdx, 0);
            pxIdx = pxIdx + 1;
        }
        rowIdx = rowIdx + 1;
    }
    win.putImageData(imgData);
    return 0;
}

main(); // start executing main
```



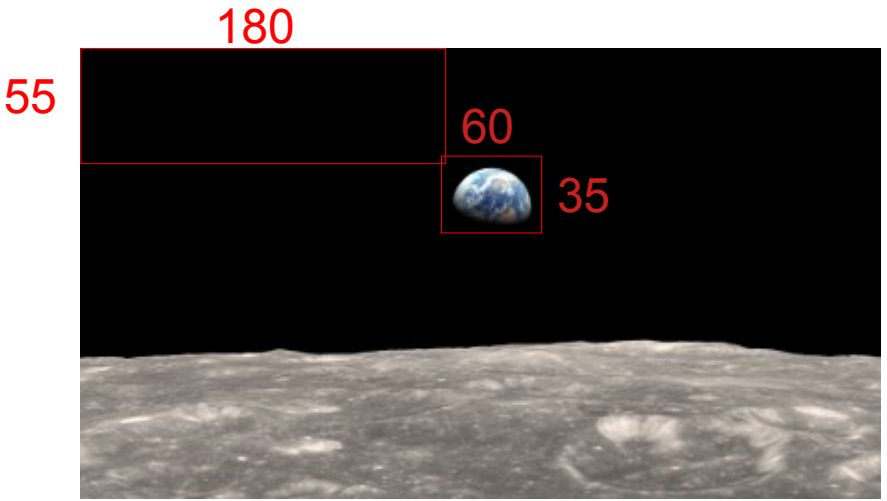
Reason to use X / Y

- The previous example could have been done with getPixels()
  - Why would you use getImageData() ?
  - Nested loop is more work (difficult)

- It allows us to manipulate a part of an image
  - Specify an exact square where we want to make changes

“Cooling” the Earth

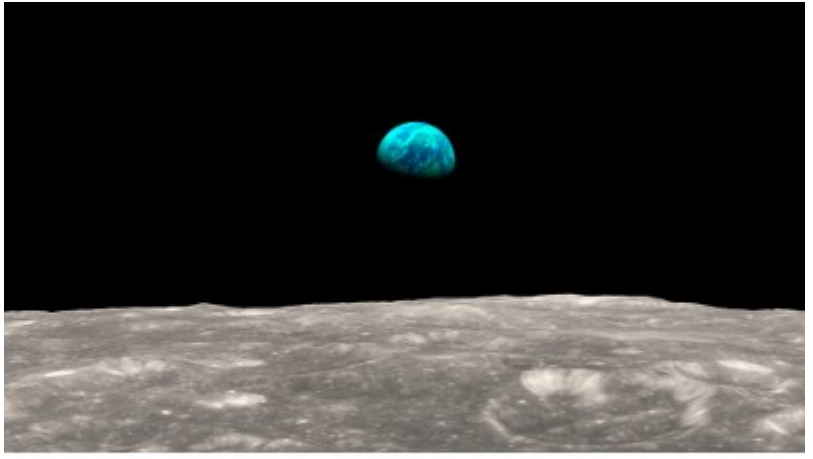
- To only remove red from the Earth
  - Find where it is in the image
  - X starts at 180 goes to 230
  - Y starts at 55 goes to 90



Code & Result

```
function main() {
    var win; // object
    var imgData; // object
    var rowIdx; // number
    var pxIdx; // number
    win = new GfxWindow(400,300);
    win.loadImage("moon_and_earth.jpg");
    imgData = win.getImageData();
    rowIdx = 55;
    while (rowIdx < 90) {
        pxIdx = 180;
        while (pxIdx < 230) {
            imgData.setRedAt(pxIdx, rowIdx, 0);
            pxIdx = pxIdx + 1;
        }
        rowIdx = rowIdx + 1;
    }
    win.putImageData(imgData);
    return 0;
}

main(); // start executing main
```



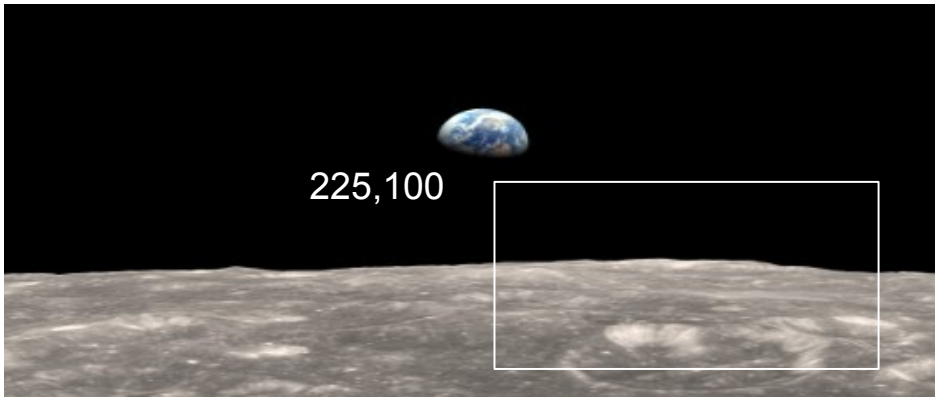
Main Point 3

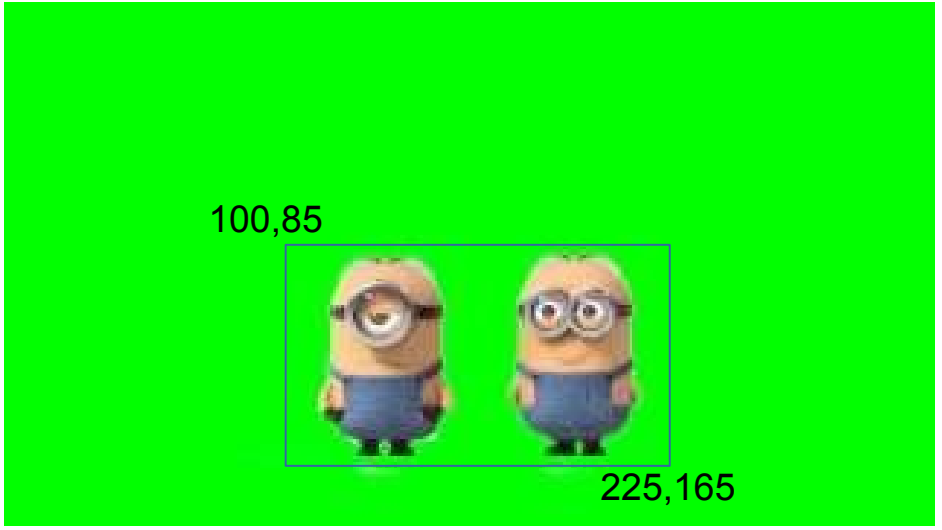
- Manipulating based on x, y locations give us greater power and flexibility
- Deeper levels of reality have greater power and flexibility

Summary

- You can load images using window.loadImage()
- You can get an array of pixels using window.getPixels()
  - Makes it easy to loop through all the pixels
- If you need to work with pixel positions you can use window.getImageData()
  - Gives access to pixels by X, Y coordinates

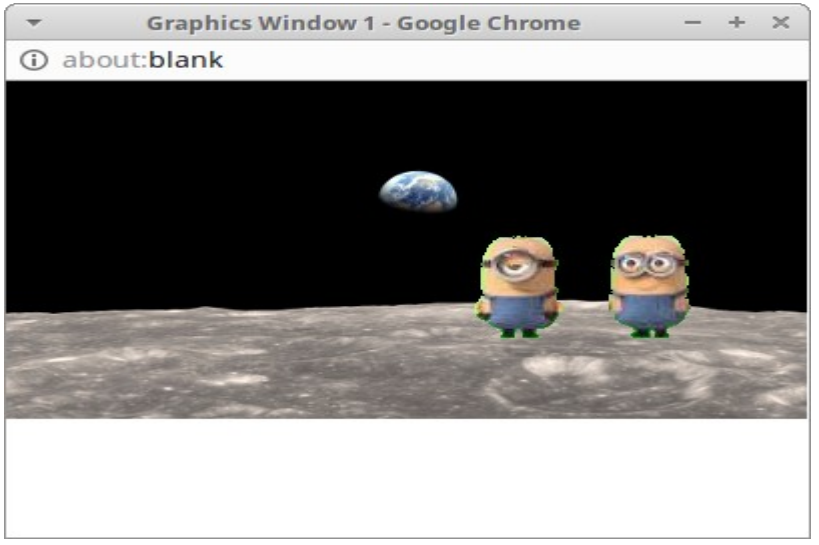
Green Screen Example





To move to a different position
X difference: 125, Y difference: 15

Result



Code

```
async function main () {
    var backWindow; // object
    var foreWindow; // object
    var backImage; // object
    var foreImage; // object
    var x; // number
    var y; // number
    backWindow = new GfxWindow(400, 300);
    await backWindow.loadImage("moon_and_earth.jpg");
    backImage = backWindow.getImageData();
    foreWindow = new GfxWindow(400, 300);
    await foreWindow.loadImage("minions.jpg");
    foreImage = foreWindow.getImageData();
    y = 85;
    while (y < 165) {
        x = 100;
        while (x < 225) {
            if (foreImage.getRedAt(x, y) < 150 && foreImage.getGreenAt(x, y) > 150 && foreImage.getBlueAt(x, y) < 150) {
            } else {
                backImage.setRedAt(x + 125, y + 15, foreImage.getRedAt(x,y));
                backImage.setGreenAt(x + 125, y + 15, foreImage.getGreenAt(x,y));
                backImage.setBlueAt(x + 125, y + 15, foreImage.getBlueAt(x,y));
            }
            x = x + 1;
        }
        y = y + 1;
    }
    backWindow.putImageData(backImage);
    return 0;
}

main(); // start executing main
```