Sound and Image Processing
Project Process Log

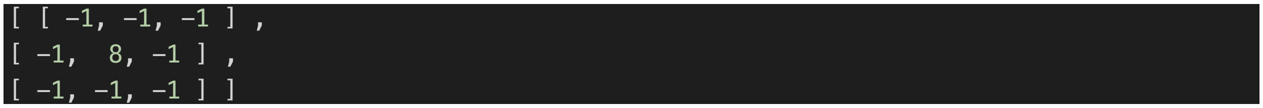
Project: Game of Life Simulation

Idea

An interactive sketch that utilise the use of mouse to enhance the edges of the photograph.

Starter code

Below is the matrix I used for this project.

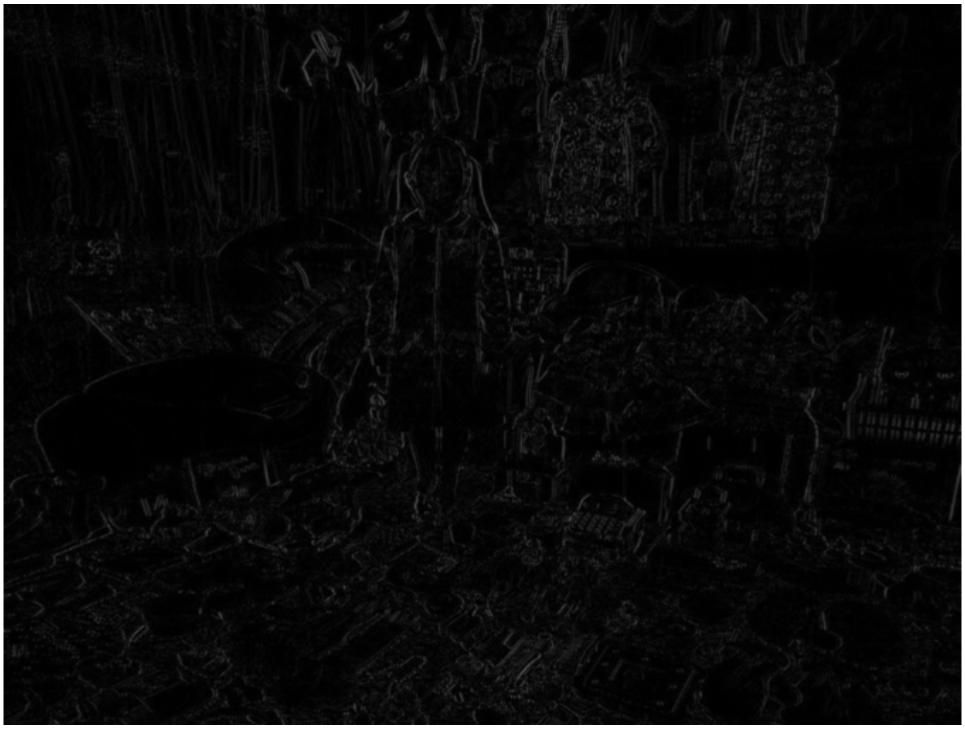


```
[ [ -1, -1, -1 ] ,
[ -1,  8, -1 ] ,
[ -1, -1, -1 ] ]
```

Knowing that the set of kernels would increase pixels that have great difference with its neighbouring pixel, I want to experiment with an image that does not have a range of pixel colors to see if the matrix could still pick up the difference and detect the edge.

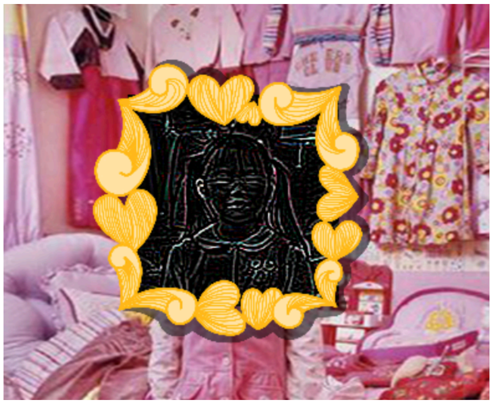
Experimentation

Below is the result of the chosen image after it is processed. Although faint the edges are still detectable.



Idea Modification and Building Interactivity

To add interactivity I utilise the user’s mouse to see sections of the processed image.



This is achieved by implementing this:

```
var start_x = constrain(mouseX- conv_width/2,0,img.width);
var end_x = constrain(mouseX +  conv_width/2,0,img.width);

var start_y = constrain(mouseY- conv_width/2,0,img.height);
var end_y = constrain(mouseY +  conv_width/2,0,img.height);
```

As the convolution is achieved by this:

```
function convolution(x, y, matrix, matrix_size, img) {
    var r_total = 0.0;
    var g_total = 0.0;
    var b_total = 0.0;
    var offset = floor(matrix_size / 2);

    // Loop matrix
    for (var i = 0; i < matrix_size; i++ ) {
      for (var j = 0; j < matrix_size; j++ ) {

        var placement_x = x + i-offset;
        var placement_y = y + j-offset;
        var placement = (placement_x + img.width * placement_y) * 4;

        placement = constrain( placement , 0, img.pixels.length-1);

        // Add all the pixels that are beside each other, multiplied by the values in
the convolution matrix.
        r_total += img.pixels[ placement    ] * matrix[i][j];
        g_total += img.pixels[ placement + 1] * matrix[i][j];
        b_total += img.pixels[ placement + 2] * matrix[i][j];
        //console.log(img.pixels[ placement    ], matrix[i][j]);
      }
    }
    r_total = constrain(r_total,0,255);
    g_total = constrain(g_total,0,255);
    b_total = constrain(b_total,0,255);

    // Return an array with the three color values
    return [r_total,g_total,b_total];
  }
```