Sound and Image Processing
Project Process Log
Project: Game of Life Simulation

Idea
A simulation of Cornway's Game of Life using the conditionals techniques used in class.

Starter code
The plan was to have different function for the initial state, a function to spawn new life, details of the screen that revolves around making the sketch aesthetic, conditions of the game of life and a population counter. This is to make the code easier to read and build to avoid the coding inefficiency in the first project.

T simulation will be in the form of a "grid layout" where each box represent either a living or dead cell. Hence the function would have to consider the canvas's center, corner and edges as the calculation of living or dead cell would be different depending on the position. The function bellow is the starter code to create the conditions for the simulation.

```
function gol_conditions() {
    //condition function for center, corner and edges

    for (var i = 0; i < elements_; i++) {
      for (var j = 0; j < elements_; j++) {
        var turnip = 0

        // Corners condition
        if (i == 0 && j == 0) {
          if (main_screen[i][j + 1]) turnip++;
          if (main_screen[i + 1][j]) turnip++;
          if (main_screen[i + 1][j + 1]) turnip++;
        } else if (i == 0 && j == (elements_ - 1)) {
          if (main_screen[i][j - 1]) turnip++;
          if (main_screen[i + 1][j]) turnip++;
          if (main_screen[i + 1][j - 1]) turnip++;
        } else if (i == (elements_ - 1) && j == 0) {
          if (main_screen[i][j + 1]) turnip++;
          if (main_screen[i - 1][j]) turnip++;
          if (main_screen[i - 1][j + 1]) turnip++;
        } else if (i == (elements_ - 1) && j == (elements_ - 1)) {
          if (main_screen[i][j - 1]) turnip++;
          if (main_screen[i - 1][j]) turnip++;
          if (main_screen[i - 1][j - 1]) turnip++;
        }
        // Border conditions
        else if (i == 0 && (j > 0 && j < (elements_ - 1))) {
          if (main_screen[i][j - 1]) turnip++;
          if (main_screen[i][j + 1]) turnip++;
          if (main_screen[i + 1][j - 1]) turnip++;
          if (main_screen[i + 1][j]) turnip++;
          if (main_screen[i + 1][j + 1]) turnip++;
        } else if (i == (elements_ - 1) && (j > 0 && j < (elements_ - 1))) {
          if (main_screen[i][j - 1]) turnip++;
          if (main_screen[i][j + 1]) turnip++;
          if (main_screen[i - 1][j - 1]) turnip++;
          if (main_screen[i - 1][j]) turnip++;
          if (main_screen[i - 1][j + 1]) turnip++;
        } else if ((i > 0 && i < (elements_ - 1)) && j == 0) {
          if (main_screen[i - 1][j]) turnip++;
          if (main_screen[i + 1][j]) turnip++;
          if (main_screen[i - 1][j + 1]) turnip++;
          if (main_screen[i][j + 1]) turnip++;
          if (main_screen[i + 1][j + 1]) turnip++;
        } else if ((i > 0 && i < (elements_ - 1)) && j == (elements_ - 1)) {
          if (main_screen[i - 1][j]) turnip++;
          if (main_screen[i + 1][j]) turnip++;
          if (main_screen[i - 1][j - 1]) turnip++;
          if (main_screen[i][j - 1]) turnip++;
          if (main_screen[i + 1][j - 1]) turnip++;
        }
        // center/general condition
        else {
          if (main_screen[i - 1][j - 1]) turnip++;
          if (main_screen[i - 1][j]) turnip++;
          if (main_screen[i - 1][j + 1]) turnip++;
          if (main_screen[i][j - 1]) turnip++;
          if (main_screen[i][j + 1]) turnip++;
          if (main_screen[i + 1][j - 1]) turnip++;
          if (main_screen[i + 1][j]) turnip++;
          if (main_screen[i + 1][j + 1]) turnip++;
        }

        // Applying changes to a secondary array.
        if (main_screen[i][j]) {
          if (turnip < 2) new_screen[i][j] = 0;
          else if (turnip > 3) new_screen[i][j] = 0;
        } else {
          if (turnip == 3) new_screen[i][j] = 1;
        }
      }
    }
  }
```

Building Interactivity Element

To add interactivity I utilize the mousePressed function that allows users to restart the simulation.

```
function mousePressed() {

    //let user restart generation
    let screen_dist = dist(mouseX, mouseY, 600, 1100);
    if (screen_dist < 90) {
      click.play();
      initial_var(screen_detail);
    }


    //instruction pop up
    if (a >= 1) {
      a = -1;
    }

    let inst_dist = dist(mouseX, mouseY, 30, 25);
    if (inst_dist < 200) {
      a++
      click.play();
      condition = loadImage(inst_array[a]);
    }

  }
```

Idea Modification

Initially I just wanted to display the number of population, however I decided to display the generations to better represent the game of life.

This is the function I made to represent the generation. It utilized the same variables I use to count the number of population.

```
function emptyGrid() {
    generation = 0;

    for (var i = 0; i < elements_; i++) {
      for (var j = 0; j < elements_; j++) {
        new_screen[i][j] = 0;
      }
    }

  }
```