2023/24

Building Grids with p5.js

Level 5 Diploma in Creative Computing

Goal of the workshop

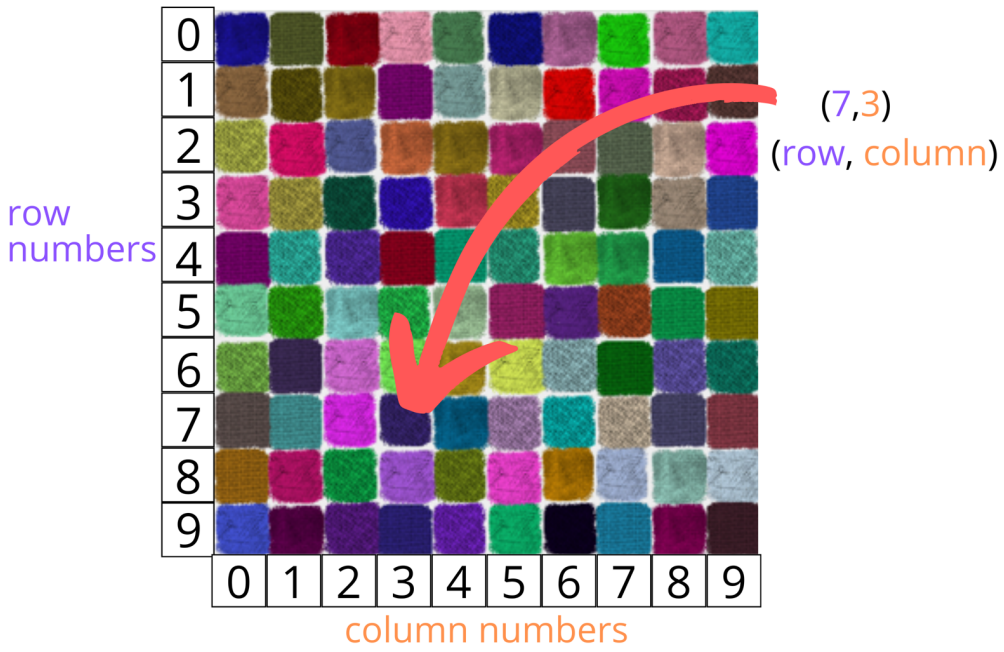
Our goal today is to use p5.js to build a grid with following characteristics:

- It has **ten rows** and **ten columns.**
- Each position, defined by a **row** and **column** position, contains a cell made up of:
    - A background texture png – randomly selected from a set of png images provided.

- A a filter color applied to this texture – randomly selected from all possible RGB, i.e., randomly selected from the 16,777,216 possibilities that RGB coding offers. (Zola, 2023)

- The images are not fully positioned in the centre of the cell. The exact position is determined by two random small offsets in the vertical and horizontal direction respectively.

Goal of the workshop - Exemplified



Starting Point

After setting up your canvas, we can start working on the grids project. Let's have a starting JavaScript as follows:

```
//set size to 800
const CANVAS_SIZE = 800;

function preload() {

}

function setup() {
  // create a canvas of size CANVAS_SIZE by CANVAS_SIZE
  createCanvas(CANVAS_SIZE, CANVAS_SIZE);

  // set up background to greyscale 220
  background(220);
}
```

As you can see, it's the same code we had after setting up p5.js with an added function named `preload`.

Starting Point – The `preload` and `setup` functions

"Called directly before setup(), the preload() function is used to handle asynchronous loading of external files in a blocking way. If a preload function is defined, setup() will wait until any load calls within have finished."
https://p5js.org/reference/#/p5/preload

"The setup() function is called once when the program starts. It's used to define initial environment properties such as screen size and background color and to load media such as images

and fonts as the program starts."
https://p5js.org/reference/#/p5/setup

Coding a grid with random colors

Since the Canvas is 800 px by 800 px, if we want 10 rows and 10 columns, each column will be 80 px by 80 px. To save this, create the following constants below the CANVAS_SIZE definition:

```
const NUMBER_OF_CELLS = 10;

const CELL_SIZE = CANVAS_SIZE / NUMBER_OF_CELLS;
```

In the setup function, you can iterate through rows and columns to fill in each cell with a color like so:

```
// loop through canvas to draw grid
  for (let row = 0; row < NUMBER_OF_CELLS; row++) {
    for (let column = 0; column < NUMBER_OF_CELLS; column++) {
      fill(20, 100, 200);
      rect(row * CELL_SIZE, column * CELL_SIZE, CELL_SIZE, CELL_SIZE);
    }
  }
```

In this code:

- Two nested for loops iterate through the rows and the columns of the grid
- In each pair (row, column), a color is chosen – for now it's always the color with RGB code (20, 100, 200).
- A rectangle is created which has the top – left corner at the position x = row * CELL_SIZE and y = column * CELL_SIZE, with sides CELL_SIZE by CELL_SIZE.

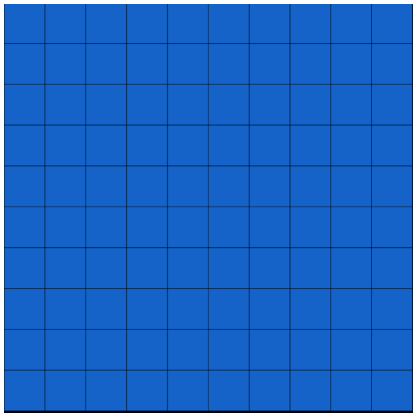
To practice, calculate what those numbers would be for:

- row = 0, column = 0
- row = 5, column = 5
- row = 9, column = 9

Your code should now look like this:

```
const CANVAS_SIZE = 800;
const NUMBER_OF_CELLS = 10;
const CELL_SIZE = CANVAS_SIZE / NUMBER_OF_CELLS;

function preload() {
}

function setup() {
  //create a canvas of 800 x 800 pixels
  createCanvas(CANVAS_SIZE, CANVAS_SIZE);

  // define a background color for the canvas
  background(220);

  // loop through canvas to create grid
  for (let row = 0; row < NUMBER_OF_CELLS; row++) {
    for (let column = 0; column < NUMBER_OF_CELLS; column++) {
      fill(20, 100, 200);
      rect(row * CELL_SIZE, column * CELL_SIZE, CELL_SIZE, CELL_SIZE);
    }
  }
}
```

And your grid should now look like this:



Now, let's create a random color for each cell. To do this, JavaScript provides a random function that can generate a random number from zero to a desired number. RGB codes are generated with a number that can go from 0 to 255. To randomize this, we can generate three numbers – one for red, one for green and one for blue – and use that as our RGB code like so:

Inside the two nested loops, write the following code:

```
const red = random(255);
const green = random(255);
const blue = random(255);
```

And then, replace the fill line with:

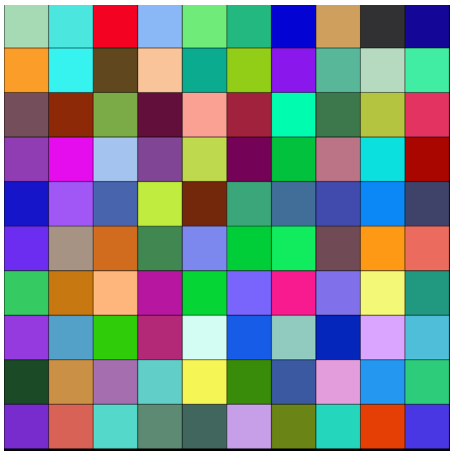
```
fill(red, green, blue);
```

Your code should now look like this:

```
const CANVAS_SIZE = 800;
const BACKGROUND_COLOR = 238;
const NUMBER_OF_CELLS = 10;
const CELL_SIZE = CANVAS_SIZE / NUMBER_OF_CELLS;
```

```
function preload() {}

function setup() {
  //create a canvas of 800 x 800 pixels
  createCanvas(CANVAS_SIZE, CANVAS_SIZE);

  // define a background color for the canvas
  background(220);

  // loop through canvas to create grid
  for (let row = 0; row < NUMBER_OF_CELLS; row++) {
    for (let column = 0; column < NUMBER_OF_CELLS; column++) {
      // calculate random color
      const red = random(255);
      const green = random(255);
      const blue = random(255);

      // set color as the fill color
      fill(red, green, blue);
      // draw square in correct place with correct size
      rect(row * CELL_SIZE, column * CELL_SIZE, CELL_SIZE, CELL_SIZE);
    }
  }
}
```

And your grid should look like this (with some different random colors):



Adding the textures

Head to the VLE and download the set of textures that we will use for our final grid. Save them in a subfolder inside the directory named images.

Commented [d1]: TEXTURES TO VLES!

In your JavaScript file, create an empty array for the textures at the top like so:

```
const BACKGROUND_TEXTURES = [];
```

Inside your preload function, make a loop that will load the textures into our new array like so:

```
// use a loop to load the similarly names images into an array
  for (let i = 1; i <= 4; i++) {
    // backgroundTextures.push(loadImage("images/texture-trans"+i+".png")); // old
school concatenation
    BACKGROUND_TEXTURES.push(loadImage(`images/texture-trans${i}.png`)); // ES6
template literal with string interpolation
  }
```

In the setup loop, after generating the red, green, and blue color, add a new line to also select a random texture from the BACKGROUND_TEXTURES array like so:

```
//get a random texture
const img = random(BACKGROUND_TEXTURES);
```

Replace the function fill with the function tint, so that instead of filling the square with a color it just acts as a filter:

```
tint(red, green, blue);
```

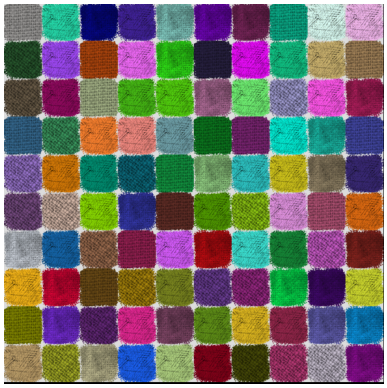
Draw the image with the following function:

```
image(img, row * CELL_SIZE, column * CELL_SIZE, CELL_SIZE, CELL_SIZE);
```

Your code should now look like this:

```
const CANVAS_SIZE = 800;
const NUMBER_OF_CELLS = 10;
const CELL_SIZE = CANVAS_SIZE / NUMBER_OF_CELLS;
const BACKGROUND_TEXTURES = [];

function preload() {
  // use a loop to load the similarly names images into an array
  for (let i = 1; i <= 4; i++) {
    // backgroundTextures.push(loadImage("images/texture-trans"+i+".png")); // old
school concatenation
    BACKGROUND_TEXTURES.push(loadImage(`images/texture-trans${i}.png`)); // ES6
template literal with string interpolation
  }
}

function setup() {
  //create a canvas of 800 x 800 pixels
  createCanvas(CANVAS_SIZE, CANVAS_SIZE);

  // define a background color for the canvas
  background(220);

  // loop through canvas to create grid
  for (let row = 0; row < NUMBER_OF_CELLS; row++) {
    for (let column = 0; column < NUMBER_OF_CELLS; column++) {
      // calculate random color
      const red = random(255);
      const green = random(255);
      const blue = random(255);

      //get a random texture
      const img = random(BACKGROUND_TEXTURES);

      //set tint
      tint(red, green, blue);
      // draw square in correct place with correct size
      image(img, row * CELL_SIZE, column * CELL_SIZE, CELL_SIZE, CELL_SIZE);
    }
  }
}
```

And your grid should look (with different randomizations) like this:



Generating a random offset

We will now generate a small random horizontal and vertical offset in each of the cells in terms of position and size, so that the textures look more natural, not as tidy. To do so, start by creating the following constants at the top:

```
const X_VARIATION = 2;
const Y_VARIATION = 3;
```

In the setup function, inside the two nested loops, after setting the tint, we will now generate a random horizontal offset which will be a random number between – X_VARIATION and X_VARIATION (in this case -2 and 2). We will do the same with the vertical offset. You can do this like so:

```
const xOffset = random(-X_VARIATION, X_VARIATION);
const yOffset = random(-Y_VARIATION, Y_VARIATION);
```

Add these changes to the result by modifying the image line like so:

```
image(img, row * CELL_SIZE + xOffset, column * CELL_SIZE + yOffset, CELL_SIZE,
CELL_SIZE);
```

Your cells should now be slightly uncentered.

To do the same for the width and hight, add two new constants at the top with following names:

```
const WIDTH_VARIATION = 5;
const HEIGHT_VARIATION = 2;
```

Add these lines after creating the x and y offsets:

```
const widthOffset = random(-WIDTH_VARIATION, WIDTH_VARIATION);
const heightOffset = random(-HEIGHT_VARIATION, HEIGHT_VARIATION);
```

And, finally, apply the changes to the image creation like so:

```
image(img, row * CELL_SIZE + xOffset, column * CELL_SIZE + yOffset, CELL_SIZE +
widthOffset, CELL_SIZE + heightOffset);
```

Since this image creation now is very convoluted, we can now tidy this last line up by replacing it like so:

```
//create positional and size variables
const xPosition = row * CELL_SIZE + xOffset;
const yPosition = column * CELL_SIZE + yOffset;
const width = CELL_SIZE + widthOffset;
const height = CELL_SIZE + heightOffset;

// draw square in correct place with correct size
image(img, xPosition, yPosition, width, height);
```

The final code of this exercise is the following:

```
const CANVAS_SIZE = 800;
const NUMBER_OF_CELLS = 10;
const CELL_SIZE = CANVAS_SIZE / NUMBER_OF_CELLS;
const BACKGROUND_TEXTURES = [];
const X_VARIATION = 2;
const Y_VARIATION = 3;
const WIDTH_VARIATION = 5;
const HEIGHT_VARIATION = 2;

function preload() {
  // use a loop to load the similarly names images into an array
  for (let i = 1; i <= 4; i++) {
```

```
    // backgroundTextures.push(loadImage("images/texture-trans"+i+".png")); // old
school concatenation
    BACKGROUND_TEXTURES.push(loadImage(`images/texture-trans${i}.png`)); // ES6
template literal with string interpolation
  }
}

function setup() {
  //create a canvas of 800 x 800 pixels
  createCanvas(CANVAS_SIZE, CANVAS_SIZE);

  // define a background color for the canvas
  background(220);

  // loop through canvas to create grid
  for (let row = 0; row < NUMBER_OF_CELLS; row++) {
    for (let column = 0; column < NUMBER_OF_CELLS; column++) {
      // calculate random color
      const red = random(255);
      const green = random(255);
      const blue = random(255);

      //get a random texture
      const img = random(BACKGROUND_TEXTURES);

      //set tint
      tint(red, green, blue);

      //generate position offset
      const xOffset = random(-X_VARIATION, X_VARIATION);
      const yOffset = random(-Y_VARIATION, Y_VARIATION);

      //generate width and height offsets
      const widthOffset = random(-WIDTH_VARIATION, WIDTH_VARIATION);
      const heightOffset = random(-HEIGHT_VARIATION, HEIGHT_VARIATION);

      //create positional and size variables
      const xPosition = row * CELL_SIZE + xOffset;
      const yPosition = column * CELL_SIZE + yOffset;
      const width = CELL_SIZE + widthOffset;
      const height = CELL_SIZE + heightOffset;

      // draw square in correct place with correct size
      image(img, xPosition, yPosition, width, height);
    }
  }
}
```