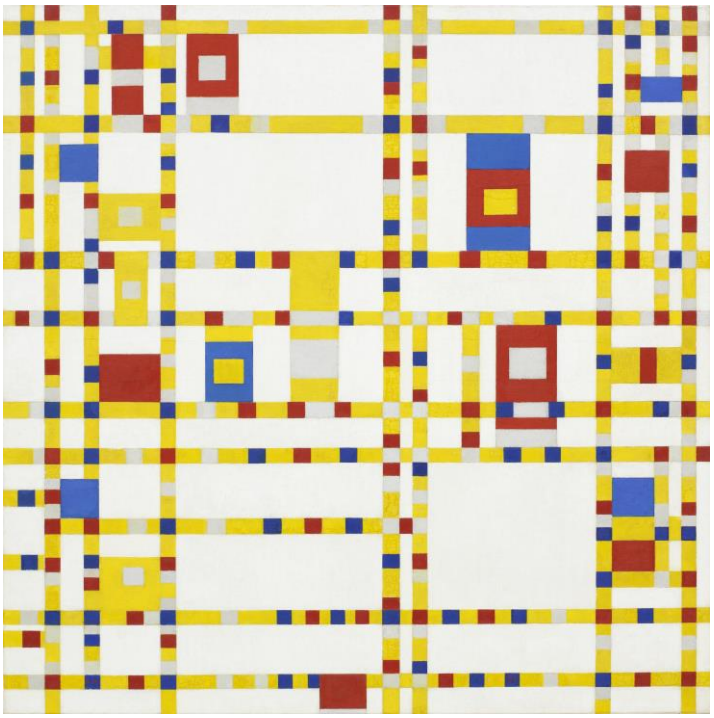


Creative Coding Major Project

IDEA9103 CREATIVE CODING

Major Project

Tutor: Ryan Van Dyk

Team: Tut 10 Group D

Student Name and Unikey:
Lingxiao Zhang_lzha0080
Jingyi Wang_jwan0961
Ruisi Zeng_rzen0962
Yanling Chen_yche4317

Section 1:
Research and inspiration:

The artwork ‘Broadway Boogie Woogie’ (Image 1) was chosen as a source of inspiration due to the respect and love that our group members have for several of Mondrian's master composers. Beyond the most basic use of geometry, Mondrian's work inspired an exploration of inner balance and order. By organising shapes and colours on the canvas, he creates a visual harmony and balance. This inner order can be found in other works of art, such as musical compositions or dance performances. So we combine Kinetic's modern art installations (Image 2), which often include dynamic elements such as moving parts, rotating structures or patterns that change over time, with ‘dynamic composition’ or ‘moving structures’, with the overall idea that The whole is dominated by dynamic elements. Combining the characteristics of both, we decided to combine dynamics in minimalist compositions, for example by arranging colour blocks and lines on the canvas, and by combining local colour blocks, choosing to change with the window. We hope to present a calm and dynamic visual effect

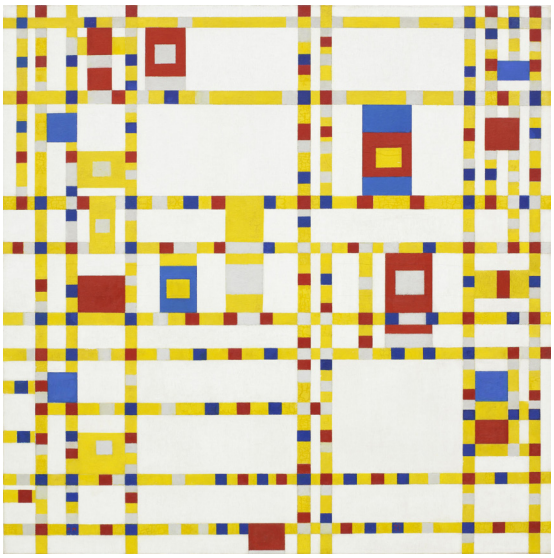


Image 1

Image 2

Meanwhile, looking back at the artwork itself, Piet Mondrian 'Broadway Boogie Woogie' is an abstract work in the De Stijl style created by Piet Mondrian in the context of the new metropolitan life he saw in New York. It contains an exploration of geometric forms and primary colors, including a large number of rectangles, the three primary colors of red, yellow and blue, and non-colors such as black, white and gray. The vitality of the city of New York has brought infinite inspiration to the author, so the author used a combination of red, blue and yellow lines and small squares to express the hustle and bustle of the city and the rhythm of music. So we were also inspired by this part when we reproduced this artwork. We hope to use the yellow long rectangle ratio as the main street of the city, the large rectangle ratio as the city buildings, and the small squares as the traffic shuttling between the cities, so as to reflect the vibrant beauty under pure geometry.

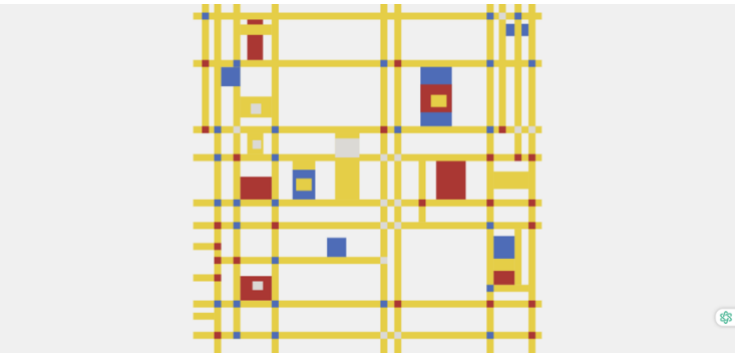
Section 2:
Technical Planning:

First, we can use the lecture class to partition the entire code into global variables. Global global variables are used throughout the page code to calculate the height and width of the gain and can be called in the windowsize function. Use the setup function to let the system calculate the partition partition for the window. Use the draw function to draw the frame background and draw the graphics of the artwork. We can draw still, moving, combined, or randomly generate images, or use the for loop function to make the graphics move randomly. We can also use class functions to define a class or define a function to classify graphics, etc.

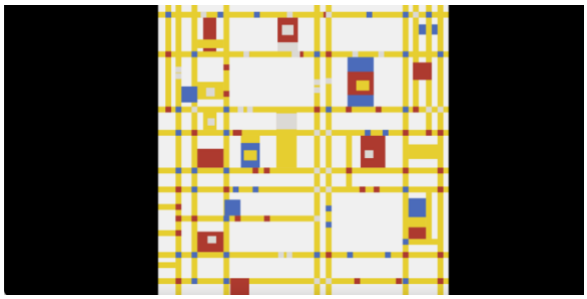
For the entire code, we hope to use the basic element code logic let rects to lay the yellow long rectangle at the top of the layer, and then use the for loop element to reflect the small square under random conditions, that is, we want to express the pattern of urban traffic flow. Then randomly change the position of the large rectangle to represent the changes of buildings in the city. Finally, add imgDrwPrps and other codes to ensure that the image can change according to the size of the browser window

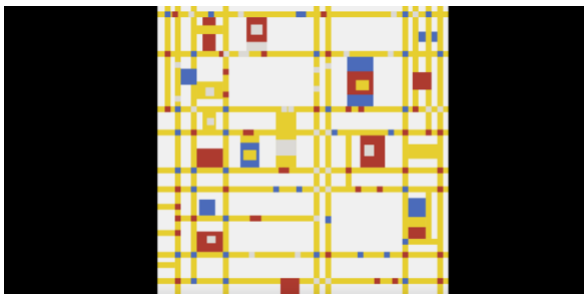
Section 3 – Implementation: Iterations of our process to achieve the outcome.

In this group task, our group had two main iterations in the process of representing the chosen artwork. As shown in the figure below, in the first iteration, all the graphic elements were fixed. We first set the background colour through function draw ( ) and drew many vertical and horizontal yellow rectangular lines. Then, we drew squares of different colours at the intersection of the fixed yellow rectangular lines. After then, we filled the interior of the yellow rectangular lines with rectangles of different sizes and colours. However, in this iteration, we did not realise that we had to make the picture change synchronously with the browser window size.



Therefore, as shown in the figure below, in the second iteration, we corrected the carelessness of the picture adaptive size problem in the previous iteration version through function windowResized ( ) and function calculateCanvasProps ( ). We also added black filling as the background to both sides of the artwork to make it look clearer. In this iteration, we added many randomly moving rectangles and rectangular lines. They will be randomly repositioned within a certain range when the user refreshes the page.





Section 4:

The first part of the group's code involves setting some basic constants, such as the number of rectangles, height, width, and other parameters of the selected artwork.

```
let imgDrwPrps = {aspect: 0, width: 0, height: 0, xOffset: 0, yOffset: 0};
let canvasAspectRatio = 0;
let numRandomRects; // number of ramdom rects
```

Code 1: parameter settings

The second part is the setup, where a canvas is created, and the ratio of window width and height is calculated. Based on calculated ratio, resize the canvas.

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  calculateCanvasProps();
  noLoop();
}
```

Code 2: setup

```
function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
  calculateCanvasProps();
  redraw(); // 重新绘制画布
}
```

Code 3: resize the canvas based on calculated ratio

```
function calculateCanvasProps() {
  // Calculate the aspect ratio of the canvas
  canvasAspectRatio = windowWidth / windowHeight;

  // Set imgDrwPrps to match the window aspect ratio
  if (canvasAspectRatio >= 1) {
    // Landscape or square
    imgDrwPrps.width = windowHeight;
    imgDrwPrps.height = windowHeight;
    imgDrwPrps.xOffset = (windowWidth - windowHeight) / 2;
    imgDrwPrps.yOffset = 0;
  } else {
    // Portrait
    imgDrwPrps.width = windowWidth;
    imgDrwPrps.height = windowWidth;
    imgDrwPrps.xOffset = 0;
    imgDrwPrps.yOffset = (windowHeight - windowWidth) / 2;
  }
}
```

Code 4: calculation of window width and height ratio

The third part is the draw() function. First, a background is created, and then the x, y, width, and height of each rectangle are defined. To allow some rectangles to move, we also define a buffer zone using the random function. After defining all the rectangles, we start drawing each one and adding color. We first draw the yellow horizontal rectangles, followed by the vertical fixed rectangles. Then we draw the red, blue, and gray rectangles. Finally, we use a function to define the width and height ratio of the window, ensuring that the pattern maintains the same proportions across different window sizes.

```
let rect1X = imgDrwPrps.xOffset;
let rect1Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.024;
let rect1W = imgDrwPrps.width;
let rect1H = imgDrwPrps.height * 0.02;

let rect2X = imgDrwPrps.xOffset;
let rect2Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.16;
let rect2W = imgDrwPrps.width;
let rect2H = imgDrwPrps.height * 0.02;

let rect3X = imgDrwPrps.xOffset;
let rect3Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.35;
let rect3W = imgDrwPrps.width;
let rect3H = imgDrwPrps.height * 0.02;

let rect4X = imgDrwPrps.xOffset;
let rect4Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.43;
let rect4W = imgDrwPrps.width;
let rect4H = imgDrwPrps.height * 0.02;

let rect5X = imgDrwPrps.xOffset;
let rect5Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.56;
let rect5W = imgDrwPrps.width;
let rect5H = imgDrwPrps.height * 0.02;

let rect6X = imgDrwPrps.xOffset;
let rect6Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.625;
let rect6W = imgDrwPrps.width;
let rect6H = imgDrwPrps.height * 0.02;

let rect7X = imgDrwPrps.xOffset;
let rect7Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.685;
let rect7W = imgDrwPrps.width * 0.06;
let rect7H = imgDrwPrps.height * 0.02;

let rect8X = imgDrwPrps.xOffset;
let rect8Y = imgDrwPrps.yOffset + imgDrwPrps.height * 0.775;
let rect8W = imgDrwPrps.width * 0.06;
let rect8H = imgDrwPrps.height * 0.02;
```

Code 6: define each rectangle's parameters, including x, y, width and height

```
let rect115X = imgDrwPrps.xOffset + imgDrwPrps.width * 0.407;
let randomRect5 = random(0.37, 0.505);
let rect115Y = imgDrwPrps.yOffset + imgDrwPrps.height * randomRect5;
let rect115W = imgDrwPrps.width * 0.07;
let rect115H = imgDrwPrps.height * 0.055;

let randomRect6 = random(0.71, 0.73);
let rect116X = imgDrwPrps.xOffset + imgDrwPrps.width * randomRect6;
let randomRect7 = random(0.48, 0.51);
let rect116Y = imgDrwPrps.yOffset + imgDrwPrps.height * randomRect7;
let randomRect8 = random(0.025, 0.04);
let rect116W = imgDrwPrps.width * randomRect8;
let randomRect9 = random(0.025, 0.05);
let rect116H = imgDrwPrps.height * randomRect9;
```

Code 7: draw some rectangles that can change location each time by using random() function

```
fill(230, 207, 48); // yellow
//horizontal lines
rect(rect1X, rect1Y, rect1W, rect1H);
rect(rect2X, rect2Y, rect2W, rect2H);
rect(rect3X, rect3Y, rect3W, rect3H);
rect(rect4X, rect4Y, rect4W, rect4H);
rect(rect5X, rect5Y, rect5W, rect5H);
rect(rect6X, rect6Y, rect6W, rect6H);
rect(rect7X, rect7Y, rect7W, rect7H);
rect(rect8X, rect8Y, rect8W, rect8H);
rect(rect9X, rect9Y, rect9W, rect9H);
rect(rect10X, rect10Y, rect10W, rect10H);
rect(rect11X, rect11Y, rect11W, rect11H);
rect(rect12X, rect12Y, rect12W, rect12H);
rect(rect13X, rect13Y, rect13W, rect13H);
```

Code 8: draw each rectangles and fill with selected color.

Section 5: GitHub links:
Tut 10 Group D:

Name:	**Unikey:**	**Github Link:**
Lingxiao Zhang	lzha0080	https://github.com/Rale1gh123/lzha0080_9103_tut07.git
Jingyi Wang	jwan0961	https://github.com/YiiiiioO/jwan0961_9103_tut7.git
Ruisi Zeng	rzen0962	https://github.com/RuisiZeng/rzen0962_9103_tut7.git
Yanling Chen	yche4317	https://github.com/YLChen7/yche4317_9103_tut7.git