DESC9103 Creative coding
Major Project

Tut 1 (Russian) Group E
jzha6441
xzha0156
jwhi6372

Section1: Research and inspiration

The artwork that has been chosen by the group is ‘Apple Tree’ by Anwar Jalal Shemza. ‘Apple Tree was created in 1962, in this piece Shemza explored organic themes and its repeating nature. This is seen in Shemza’s use of repetitive circular and linear patterns which are used to represent the structure of a common tree. Through this Shemza has created a sense of rhythm and balance which is experienced in nature.

This artwork is shown below:

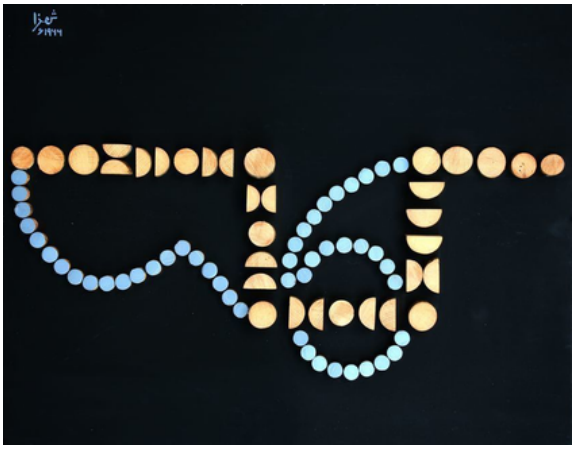
Shemza took strong inspiration from the Biblical Genesis story of the Garden of Eden. The tree in his artwork thus represents the Tree of Knowledge which Eve takes the forbidden fruit from. This represents the two sides of the human experience, one being the pursuit of knowledge and the freedom which it can give and its potential downfall. This is represented in the fruit being two colours, one which represents the calmness and security which was offered by God and the other symbolising the downfall which the Devil offered to the couple. The same characteristics also shown in other artworks by Anwar

Section 1: Research and inspiration

These same characteristics of repetition of linear and circular patterns is also seen in his other works. These works are shown below:

Composition in Red and Green, 1963.

Meem Two 1967.



Composition with a Number Six, 1966

Apple Trees, 1970.

Section 1: Research and inspiration

The reason that Apple Tree has been chosen by the group is because it can be created in the same method with JavaScript and specifically p5.js, which leads itself to use simple geometric shapes to create natural artworks like what Shemza has created. This has inspired the group to follow a similar structure by using simple geometric shapes, such as lines and circles, to create nature-based artworks in code. Also, this concept helps the group understand how to use JavaScript to generate the structural layout of branches and leaves observed in Shemza's work. Similarly, we will use the concept of generative art to repeat the lines of branches and leaves to build the complexity of the work. Moreover, the nature elements such as trees has a potential of growth, allowing the opportunity to create the iteration of the form of our individual works.Moreover, the nature elements such as trees has a potential of growth, allowing the opportunity to create the iteration of the form of our individual works.

Inspiration for the final piece can be shown in the links below. These pieces of artwork show the growth of a tree which has given us inspiration to use moving featuring in our work to make the final piece movement. This will give the effect of making the final piece to feel active and alive.

Link to Precedents1
Link to Precedents2
Link to Image and artworks information

Section 2: Technical Planning

To be able to recreate Shemza's ‘Apple Tree’ using p5.js it is planned to segregate the code into distinct sections which will allow for more succinct writing and reading of the entire script. The code will be split into three main sections:

- drawTree
- drawCircles
- drawBaseStructure

Within these three sections of the code there will be nested functions which will allow for the group to be able to complete each of the sections with ease. drawTree will include two functions which will use for loops which will draw the branches of the tree. Due to there being both vertical and horizontal lines it is will be more convenient for the drawTree section to be split into these two sections. For the branches to grow and shrink with the size of the window the windowResize function will need to be used here.

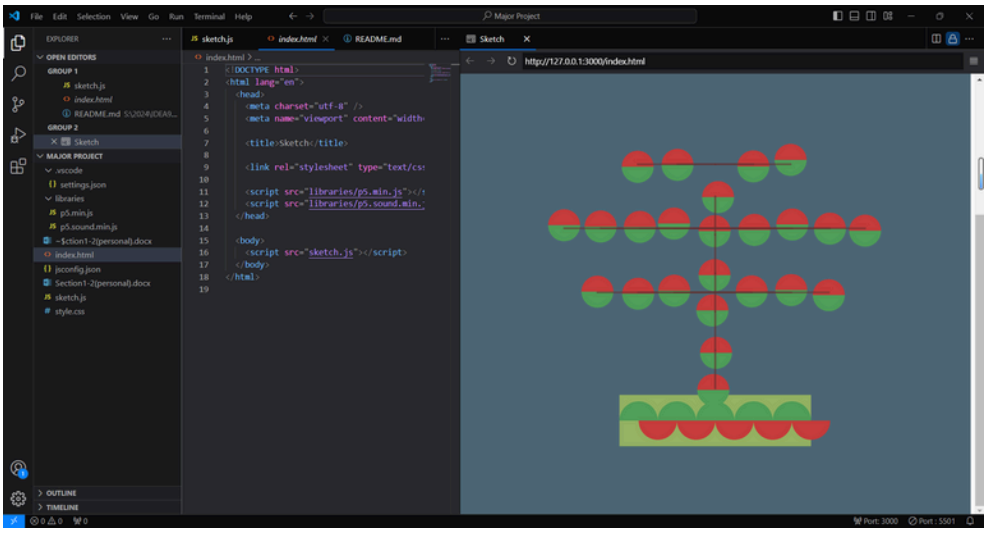
The next section is the drawCircles which will be used to draw semicircles using the built in arc function. These semicircles will represent the fruits which Shemza similarly used in his artwork. Within this section for loops will also be used to easily draw said semicircles. Using for loops will also make it simpler to have these semicircles grow and shrink and the window is resized. For this the windowResize function will also need to be used here.

The final section will be the drawBaseStructure which will be used to draw the base which the tree will be placed on. This will be drawn in two parts, one being a rectangle using the rect function and the second being the array of semicircles which are included in the base. The rectangle will be simple; however, the semicircles will be drawn with a for loop. Similarly, to the other sections this whole section will need to grow and shrink with the size of the window so the windowResize function will be implemented here.

Section 3: Implementation

Initial idea

This is the initial idea of our group coding, which is using the same characteristics as Anwar to create another abstract art piece about apple tree.
The work was organized by repeated geometric patterns such as circles, rectangular, also the line.
The Perlin noise is also used in the coding to make the apple swing.



Section 3: Implementation

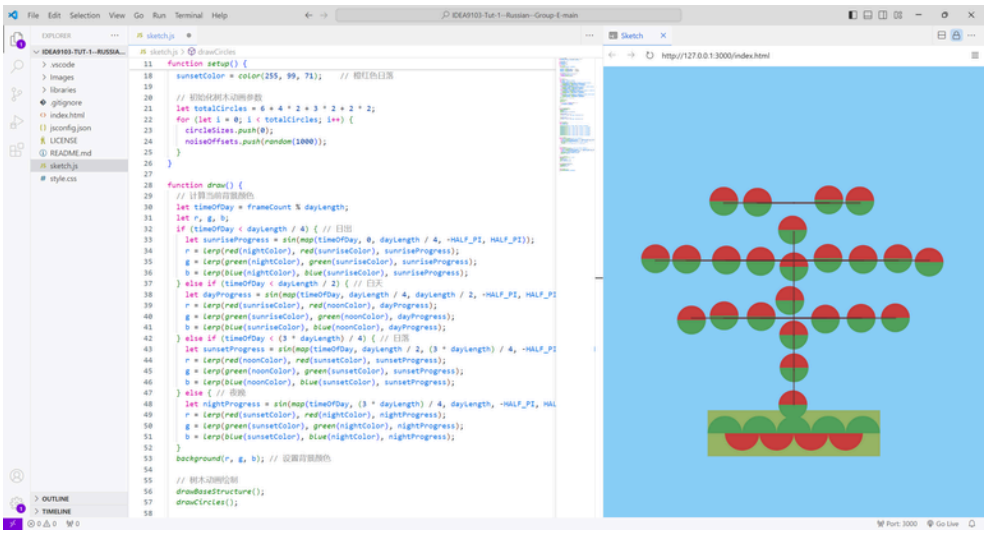
Iteration 1

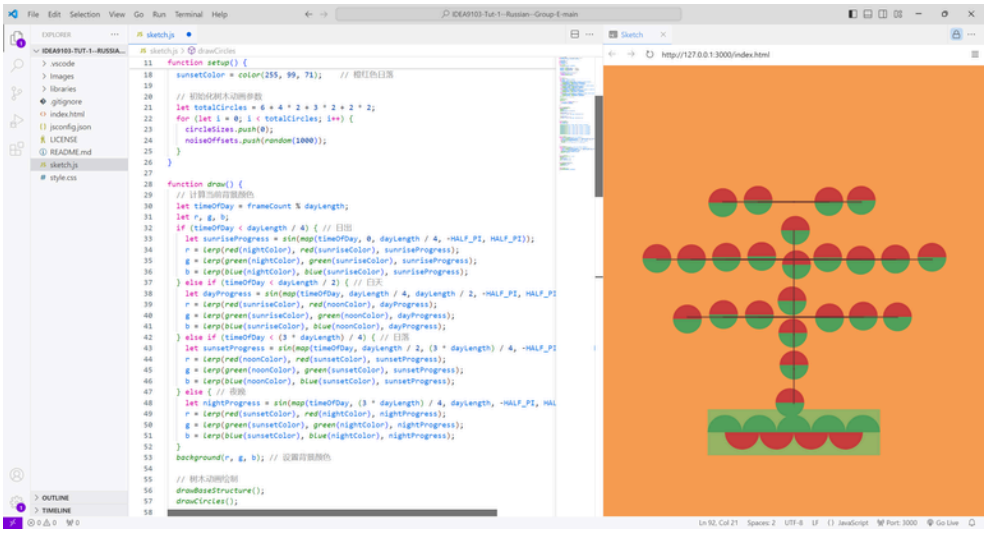
The background has been set into different colors in this stage, which refers to the four time periods of the day:

- Night
- Sunrise
- Daytime
- Sunset

The color of the background will be automatically changed with time.

The size of apple tree can be scaled with the size of window.

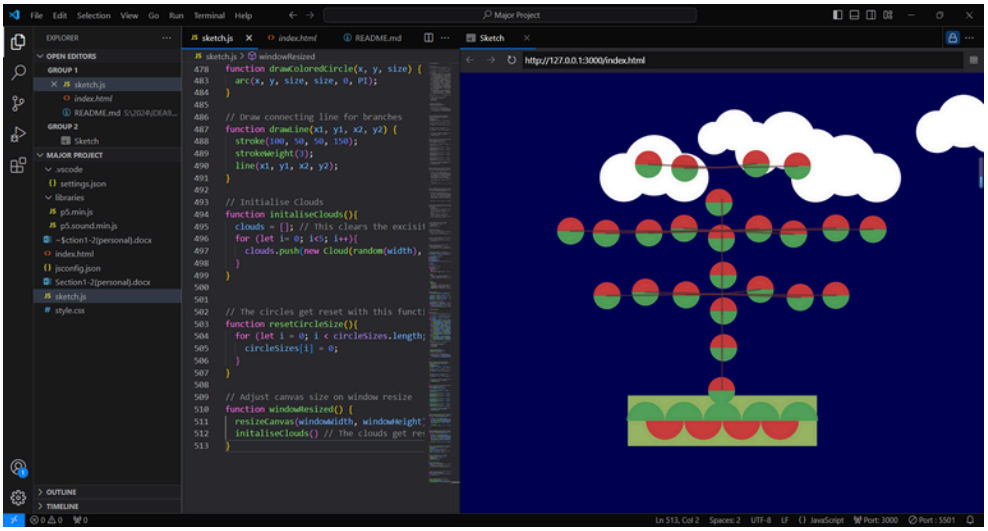




Section 3: Implementation

Iteration 2

The clouds were added which can move across the screen, the clouds are generated randomly.



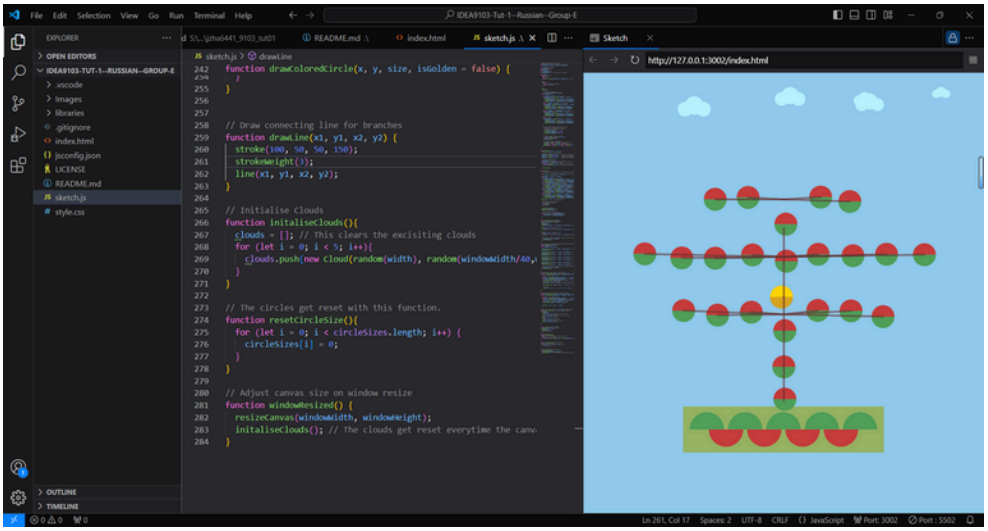
Section 3: Implementation

Iteration 3

Considered the religion background of the inspiration of Anwar's work, the color of one of the apples into gold, which refers to the forbidden fruit.

The color of clouds can now be changed with the color of background. The size of clouds can be scaled with the size of window.

Slow down the refresh frequency



Section 4 – Technical overview

Set up the key parameters that will be used in function

dayLength: the length of the day

sunriseColor, **noonColor**, **sunsetColor**, **nightColor**: the color of background in different time.

growthSpeed, **maxSize**: control the speed of apple grown and the maximum size of apple.

circleSize, **noiseOffsets**, **swayNoiseOffsets**: control the movement of every apple.

```
// Dynamic background color variables
let dayLength = 1000; // Duration of a day (in frames)
let sunriseColor, noonColor, sunsetColor, nightColor;

// Tree animation variables
let growthSpeed = 0.4;
let maxSize = 40;
let circleSizes = [];
let noiseOffsets = [];
let swayNoiseOffset; // Noise offset for sway effect

```

Identify a Cloud class, which allows each cloud to have its own shape. Use **move()** to let clouds move across the window.

```
class Cloud {
  constructor(x, y) {
    this.x = x;
    this.y = y;
    this.size = random(windowWidth/40, windowWidth/20); //The size of the clouds changes with the size of window
  }

  move() {
    this.x += 1;
    if (this.x > width + this.size) {
      this.x = -this.size;
    }
  }
```

Section 4 – Technical overview

In this part, use **(sin)** and **(lerp)** to achieve the color interpolation and day-night circle.

```
show() {
  // Get current background color
  let timeOfDay = frameCount % dayLength;
  let currentBgColor;

  if (timeOfDay < dayLength / 4) { // Sunrise
    let sunriseProgress = sin(map(timeOfDay, 0, dayLength / 2, -HALF_PI, HALF_PI));
    currentBgColor = color(
      lerp(red(nightColor), red(sunriseColor), sunriseProgress),
      lerp(green(nightColor), green(sunriseColor), sunriseProgress),
      lerp(blue(nightColor), blue(sunriseColor), sunriseProgress)
    );
  } else if (timeOfDay < dayLength / 2) { // Daytime
    let dayProgress = sin(map(timeOfDay, dayLength / 4, dayLength / 2, -HALF_PI, HALF_PI));
    currentBgColor = color(
      lerp(red(sunriseColor), red(noonColor), dayProgress),
      lerp(green(sunriseColor), green(noonColor), dayProgress),
      lerp(blue(sunriseColor), blue(noonColor), dayProgress)
    );
  } else if (timeOfDay < (3 * dayLength) / 4) { // Sunset
    let sunsetProgress = sin(map(timeOfDay, dayLength / 2, (3 * dayLength) / 4, -HALF_PI, HALF_PI));
    currentBgColor = color(
      lerp(red(noonColor), red(sunsetColor), sunsetProgress),
      lerp(green(noonColor), green(sunsetColor), sunsetProgress),
      lerp(blue(noonColor), blue(sunsetColor), sunsetProgress)
    );
  } else { // Night
    let nightProgress = sin(map(timeOfDay, (3 * dayLength) / 4, dayLength, -HALF_PI, HALF_PI));
    currentBgColor = color(
      lerp(red(sunsetColor), red(nightColor), nightProgress),
      lerp(green(sunsetColor), green(nightColor), nightProgress),
      lerp(blue(sunsetColor), blue(nightColor), nightProgress)
    );
  }
```

set up the color of cloud and draw clouds by several circles.

```
  // Create gradient effect for each cloud element
  for (let i = 0; i <= this.size; i += 2) {
    // Make cloud color lighter than background
    let cloudColor = color(
      min(red(currentBgColor) + 40, 255),
      min(green(currentBgColor) + 40, 255),
      min(blue(currentBgColor) + 40, 255)
    );

    // Create gradient from top to bottom
    let alpha = map(i, 0, this.size, 255, 180);
    cloudColor.setAlpha(alpha);
    fill(cloudColor);

    // Draw cloud shapes with gradient
    let yOffset = map(i, 0, this.size, 0, this.size * 0.2);
    ellipse(this.x, this.y + yOffset, this.size - i);
    ellipse(this.x + this.size * 0.5, this.y + this.size * 0.2 + yOffset, (this.size - i) * 0.8);
    ellipse(this.x - this.size * 0.5, this.y + this.size * 0.2 + yOffset, (this.size - i) * 0.8);
  }
}
```

Section 4 – Technical overview

set up the size of canvas as the size of window by use **createCanvas(windowWidth, windowHeight)**

set up the different colors in a day by RGB

Initializes arrays for each circle’s size and noise offset, giving each a unique starting point for growth and sway.

use **initializeClouds()** to create clouds

```
function setup() {
  createCanvas(windowWidth, windowHeight);

  // Initialize dynamic background colors
  nightColor = color(20, 24, 82);      // Night color (deep blue)
  sunriseColor = color(255, 160, 80);  // Sunrise color (orange)
  noonColor = color(135, 206, 250);    // Daytime color (light blue)
  sunsetColor = color(255, 99, 71);    // Sunset color (orange-red)

  // Initialize tree parameters
  let totalCircles = 6 + 4 * 2 + 3 * 2 + 2 * 2;
  swayNoiseOffset = random(1000); // Initialize noise offset for sway effect
  for (let i = 0; i < totalCircles; i++) {
    circleSizes.push(0);          // Initial size of all circles is 0
    noiseOffsets.push(random(1000)); // Random noise offset for each circle
  }

  initaliseClouds();
}
```

Section 4 – Technical overview

Use **draw()** to updates continuously to animate the scene.
Use **frameCount % dayLength** to calculate the time in timeOfDay
Use **lerp()** to change the color of the day
sin(), map() to change the color smoothly.
Use **move()** to let clouds move across the window and use **show()** to display them.
Control the growth speed of apples

```
function draw() {
  // Calculate current background color (transitioning from night to sunrise, daytime, and then sunset)

  let scaleFactor = min(windowWidth, windowHeight) / 700; //scaleFactor is used to reshape the sizes of the fruit and the tree as the window

  let timeOfDay = frameCount % dayLength;
  let r, g, b;
  if (timeOfDay < dayLength / 4) { // Sunrise
    let sunriseProgress = sin(map(timeOfDay, 0, dayLength / 2, -HALF_PI, HALF_PI));
    r = lerp(red(nightColor), red(sunriseColor), sunriseProgress);
    g = lerp(green(nightColor), green(sunriseColor), sunriseProgress);
    b = lerp(blue(nightColor), blue(sunriseColor), sunriseProgress);
  } else if (timeOfDay < dayLength / 2) { // Daytime
    let dayProgress = sin(map(timeOfDay, dayLength / 4, dayLength / 2, -HALF_PI, HALF_PI));
    r = lerp(red(sunriseColor), red(noonColor), dayProgress);
    g = lerp(green(sunriseColor), green(noonColor), dayProgress);
    b = lerp(blue(sunriseColor), blue(noonColor), dayProgress);
  } else if (timeOfDay < (3 * dayLength) / 4) { // Sunset
    let sunsetProgress = sin(map(timeOfDay, dayLength / 2, (3 * dayLength) / 4, -HALF_PI, HALF_PI));
    r = lerp(red(noonColor), red(sunsetColor), sunsetProgress);
    g = lerp(green(noonColor), green(sunsetColor), sunsetProgress);
    b = lerp(blue(noonColor), blue(sunsetColor), sunsetProgress);
  } else { // Night
    let nightProgress = sin(map(timeOfDay, (3 * dayLength) / 4, dayLength, -HALF_PI, HALF_PI));
    r = lerp(red(sunsetColor), red(nightColor), nightProgress);
    g = lerp(green(sunsetColor), green(nightColor), nightProgress);
    b = lerp(blue(sunsetColor), blue(nightColor), nightProgress);
  }
  background(r, g, b); // Set background color
```

```
// Clouds move across the window
for (let cloud of clouds){
  cloud.move();
  cloud.show();
}

// Draw tree animation
drawBaseStructure(scaleFactor);
drawCircles(scaleFactor);

// Control growth of circle sizes
for (let i = 0; i < circleSizes.length; i++) {
  if (circleSizes[i] < maxSize*scaleFactor) {
    circleSizes[i] += growthSpeed*scaleFactor;
  }
}
```

Section 4 – Technical overview

Use **rect()** to draw the ground and add decoration by use **arc()**

Use **if** to check **frameCount % 300 === maxSize**, if it is true, enabling them to grow again.

```
// Draw the base structure (flowerpot)
function drawBaseStructure(scaleFactor) {
  fill(150, 180, 100); // Pot color
  noStroke();
  rectMode(CENTER);
  rect(width / 2, height - 150*scaleFactor, 350*scaleFactor, 80*scaleFactor);

  fill(80, 160, 90); // Green semi-circles
  for (let i = 0; i < 5; i++) {
    arc(width / 2 - 120 + i * 60, height - 150*scaleFactor, 60*scaleFactor, 60*scaleFactor, PI, 0);
  }

  fill(200, 60, 60); // Red semi-circles
  for (let i = 0; i < 4; i++) {
    arc(width / 2 - 90 + i * 60, height - 150*scaleFactor, 60*scaleFactor, 60*scaleFactor, 0, PI);
  }

  // The circle refreshes, to change the speed of this the number after framecount needs to be changed.
  if (frameCount % 300 === maxSize){
    resetCircleSize();
  }
}
```

This part is the code to draw the apples in both vertical and horizontal way by set up **drawVerticalCircles()** and **drawHorizontalCircles().**

also enabling them if **frameCount % 300 === maxSize**

```
// Draw circles for tree trunk and branches with noise-based sway
function drawCircles(scaleFactor) {
  let currentIndex = 0;
  let circleSize = 50*scaleFactor;

  drawVerticalCircles(width / 2, height - 200*scaleFactor, 6, circleSize, currentIndex, scaleFactor);
  currentIndex += 6;

  drawHorizontalCircles(width / 2, height - 450*scaleFactor, 4, circleSize, -1, currentIndex, scaleFactor);
  currentIndex += 4;
  drawHorizontalCircles(width / 2, height - 450*scaleFactor, 4, circleSize, 1, currentIndex, scaleFactor);
  currentIndex += 4;

  drawHorizontalCircles(width / 2, height - 350*scaleFactor, 3, circleSize, -1, currentIndex, scaleFactor);
  currentIndex += 3;
  drawHorizontalCircles(width / 2, height - 350*scaleFactor, 3, circleSize, 1, currentIndex, scaleFactor);
  currentIndex += 3;

  drawHorizontalCircles(width / 2, height - 550*scaleFactor, 2, circleSize, -1, currentIndex, scaleFactor);
  currentIndex += 2;
  drawHorizontalCircles(width / 2, height - 550*scaleFactor, 2, circleSize, 1, currentIndex, scaleFactor);

  // The circle refreshes, to change the speed of this the number after framecount needs to be changed.
  if (frameCount % 300 === maxSize){
    resetCircleSize();
  }
}
```

Section 4 – Technical overview

Draw the apples in both vertical and horizontal

Use Perlin Noise to create the swing of each apple

Set up one of the apples into gold color by tag it **isGolden.**

```
// Draw vertical circles (trunk) with sway effect
function drawVerticalCircles(x, y, count, size, indexStart, scaleFactor) {
  let sway = map(noise(swayNoiseOffset + frameCount * 0.01), 0, 1, -5*scaleFactor, 5*scaleFactor); // Calculate sway

  for (let i = 0; i < count; i++) {
    let noiseX = map(noise(noiseOffsets[indexStart + i] + frameCount * 0.01), 0, 1, -10*scaleFactor, 10*scaleFactor);
    let noiseY = map(noise(noiseOffsets[indexStart + i] + 1000 + frameCount * 0.01), 0, 1, -10*scaleFactor, 10*scaleFactor);
    let circleSize = circleSizes[indexStart + i];
    drawColoredCircle(x + noiseX + sway, y - i * size * 1.2 + noiseY, circleSize);
    // Set the first circle as golden
    let isGolden = (i === 3); // Change this index to make a different circle golden

    drawColoredCircle(x + noiseX + sway, y - i * size * 1.2 + noiseY, circleSize, isGolden);

    if (i > 0) {
      drawLine(x + sway, y - (i - 1) * size * 1.2, x + sway, y - i * size * 1.2);
    }
  }
}
```

```
// Draw horizontal circles (branches) with sway effect
function drawHorizontalCircles(x, y, count, size, direction, indexStart, scaleFactor) {
  let sway = map(noise(swayNoiseOffset + frameCount * 0.01), 0, 1, -5*scaleFactor, 5*scaleFactor); // Calculate sway

  for (let i = 1; i <= count; i++) {
    let noiseX = map(noise(noiseOffsets[indexStart + i - 1] + frameCount * 0.01), 0, 1, -10*scaleFactor, 10*scaleFactor);
    let noiseY = map(noise(noiseOffsets[indexStart + i - 1] + 1000 + frameCount * 0.01), 0, 1, -10*scaleFactor, 10*scaleFactor);
    let xPos = x + i * size * 1.2 * direction + noiseX + sway;
    let circleSize = circleSizes[indexStart + i - 1];
    drawColoredCircle(xPos, y + noiseY, circleSize);


    drawLine(x + sway, y, xPos, y + noiseY);
  }
}
```

Use **drawColoredCircle()** to draw apples.
If **isGolden** is **true**, draw apple in gold color, otherwise, draw apples in red and green.

```
// Draw a circle with alternating red and green halves, and one with gold
function drawColoredCircle(x, y, size, isGolden = false) {
  noStroke();
  if (isGolden) {
    fill(255, 215, 0); // Gold color for the top half
    arc(x, y, size, size, PI, 0);
    fill(218, 165, 32); // Darker gold for the bottom half
    arc(x, y, size, size, 0, PI);
  } else {
    fill(200, 60, 60); // Red top half
    arc(x, y, size, size, PI, 0);
    fill(80, 160, 90); // Green bottom half
    arc(x, y, size, size, 0, PI);
  }
}
```

Section 4 – Technical overview

Draw lines as branches of the tree.

```
// Draw connecting line for branches
function drawLine(x1, y1, x2, y2) {
  stroke(100, 50, 50, 150);
  strokeWeight(3);
  line(x1, y1, x2, y2);
}
```

The coding in this part is support the scene works.

initaliseClouds():
clear existing cloud and create a new one with random size and position

resetCircleSize():
reset the size of apple to 0 to allow them growth again

windowResized():
resize the canvas with the size of window, when the size of window has changed, initialise the clouds.

```
// Initialise Clouds
function initaliseClouds(){
  clouds = []; // This clears the excisiting clouds
  for (let i = 0; i < 5; i++){
    clouds.push(new Cloud(random(width), random(windowWidth/40,windowWidth/10)));
  }
}

// The circles get reset with this function.
function resetCircleSize(){
  for (let i = 0; i < circleSizes.length; i++) {
    circleSizes[i] = 0;
  }
}

// Adjust canvas size on window resize
function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
  initaliseClouds(); // The clouds get reset everytime the canvas gets adjusted.
}
```

Section 5 – GitHub links

Jack Whittles

https://github.com/jwhittles/MajorAssignmentIndividual.git

Selina Zhang

https://github.com/Selina-zxy/xzha0156_9103_Final.git

Jackson Zhao

https://github.com/JacksonZhao5100/Jianbo-Zhao-Major-Assignment.git