MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA Universitatea Tehnică a Moldovei Facultatea Calculatoare, Informatică şi Microelectronică Departamentul Inginerie Software și Automatică

Copta Adrian | FAF-223

Report

Laboratory work n.3

of Computer Graphics

Checked by:

Olga Grosu, *university assistant*

DISA, FCIM, UTM

Chișinău – 2023

1. **Purpose of the task work:**

Task 1:

- To make a sketch using 3D Primitives.

Task 2:

- To make array objects with 2D and (or) 3D primitives.

You can combine point a and b task 2 and do one sketch or separately

2. **Condition:**

Task 1:

Make a sketch using 3D Primitives SPHERE, sphereDetail(), BOX, createShape(), Quad(), pushmatrix(), popmatrix(). Rotate them using rotateX(), rotateY(), rotateZ() or rotate(). Move some of the figures horizontally, some of them vertically and also move the figure on the main and secondary diagonal of the window sketch.

Task 2:

Make array objects with 2D and (or) 3D primitives You can combine point a and b task 2 and do one sketch or separately

3. **Program code:**

```
/*
MADE BY ADRIAN COPTA | FAF-223
*/


// Declare an array to hold PShape objects and the number of shapes
PShape[] shapes;
int numShapes = 10;

void setup() {
```

```
  // Create a 3D canvas of size 400x400 pixels
  size(400, 400, P3D);

  // Initialize the array to hold PShape objects
  shapes = new PShape[numShapes];

  // Create PShape objects and store them in the array
  for (int i = 0; i < numShapes; i++) {
    int choice = int(random(3));
    if (choice == 0) {
      shapes[i] = createSphere();
    } else if (choice == 1) {
      shapes[i] = createBox();
    } else {
      shapes[i] = createQuad();
    }
  }
}

void draw() {
  // Set the background color to light gray
  background(200);

   // Translate the coordinate system to the center of the canvas
  translate(width / 2, height / 2, 0);

   // Calculate the length of the main diagonal of the canvas
```

```
  float diagonal = sqrt(sq(width) + sq(height));

  // Loop through the array of shapes
  for (int i = 0; i < numShapes; i++) {
      // Calculate the x, y, and z positions for each
shape based on time and index
     float x = map(sin(radians(frameCount + i * 15)), -1,
1, -width / 2, width / 2);
     float y = map(cos(radians(frameCount + i * 20)), -1,
1, -height / 2, height / 2);
     float z = map(sin(radians(frameCount + i * 10)), -1,
1, -diagonal / 2, diagonal / 2);

      // Push the current transformation matrix onto the
stack
    pushMatrix();

    // Translate and rotate each shape
    translate(x, y, z);
    rotateX(radians(frameCount));
    rotateY(radians(frameCount));

    // Display the current shape from the array
    shape(shapes[i]);

    // Restore the previous transformation matrix
    popMatrix();
  }
}
```

```
// Function to create a random colored sphere
PShape createSphere() {
  PShape s = createShape(SPHERE, 50);
           s.setFill(color(random(255),     random(255),
random(255)));
  return s;
}

// Function to create a random colored box
PShape createBox() {
  PShape b = createShape(BOX, 80);
           b.setFill(color(random(255),     random(255),
random(255)));
  return b;
}

// Function to create a random colored quad
PShape createQuad() {
   PShape  q  =  createShape(QUAD,  -40,  -40,  40,  -40,  40,
40, -40, 40);
           q.setFill(color(random(255),     random(255),
random(255)));
  return q;
}
```

4. **Program execution:**

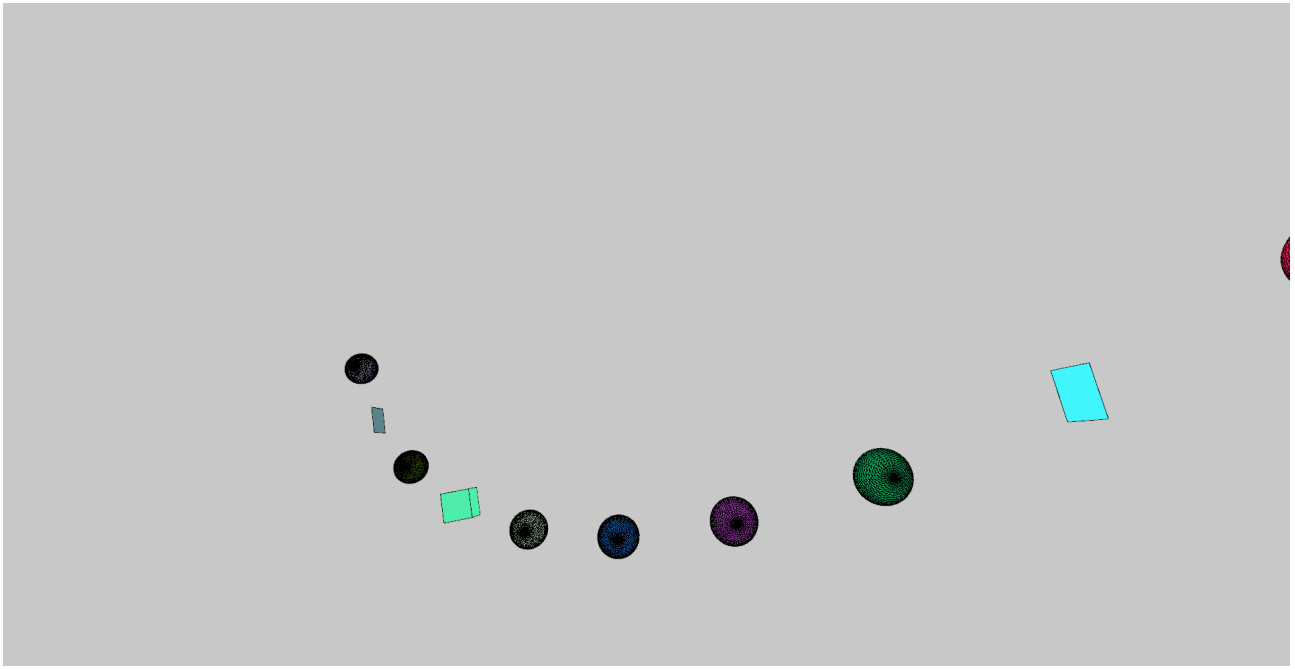

fig.(1) The array of shapes moving on the main diagonal of the canvas

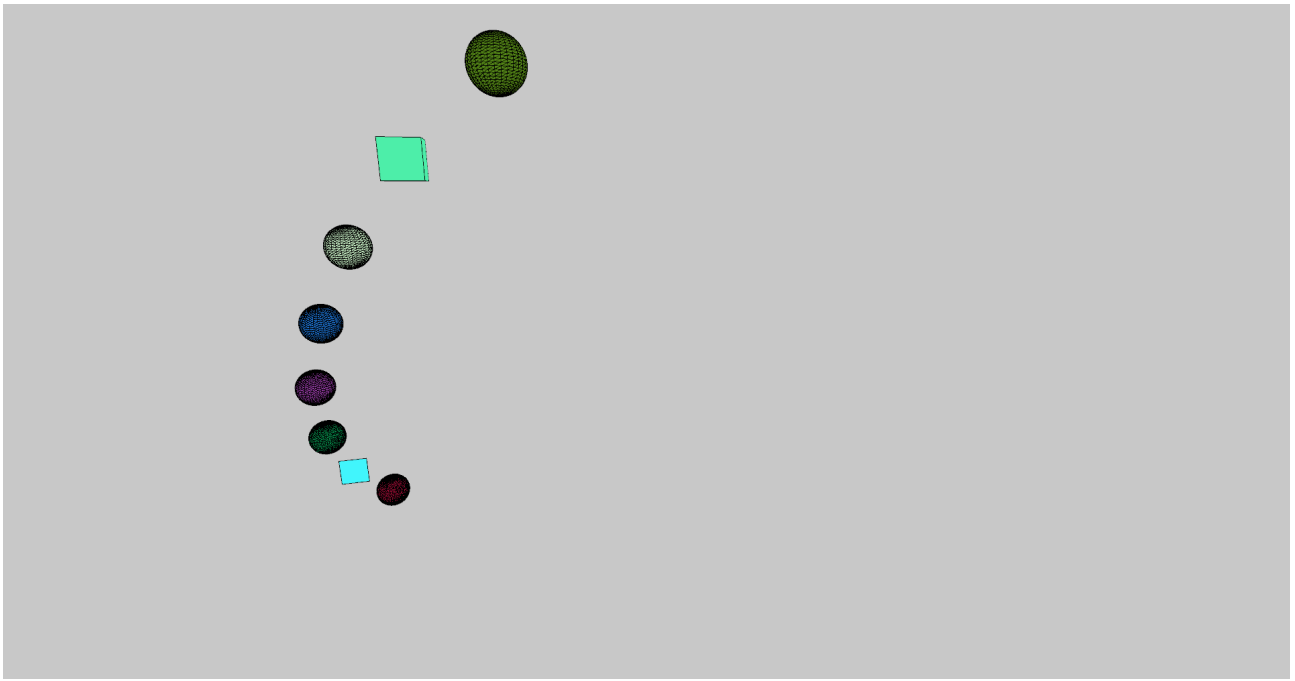

fig.(2) The array of shapes coming back on the main diagonal of the canvas

5. **Conclusion:**

In conclusion, the provided code is a creative and visually engaging example of utilizing the Processing programming language to generate a dynamic 3D art display. The code accomplishes the following key tasks. It declares an array of shapes to hold

PShape objects and defines the number of shapes numShapes to be displayed, which is initially set to 10. In the setup() function, a 3D canvas of size 400x400 pixels is created using the size() function. It initializes the shapes array to hold PShape objects and populates it with a variety of shapes (spheres, boxes, and quads) using the createSphere(), createBox(), and createQuad() functions. The shapes are randomly assigned colors. The draw() function is where the magic happens. It continuously updates the display. It starts by setting the background color to light gray and translating the coordinate system to the center of the canvas. Within a loop, it calculates the positions (x, y, and z) of each shape based on time and index. The shapes move and rotate in a mesmerizing way due to the use of trigonometric functions. This creates a dynamic and visually appealing effect. Before translating and rotating each shape, the code uses pushMatrix() to save the current transformation matrix state and popMatrix() to restore it afterward. This ensures that transformations do not affect subsequent shapes. It uses the shape() function to display each shape from the array at its calculated position and rotation. The result is a constantly changing and visually intriguing 3D art composition. Overall, this code showcases the power of creative coding in generating dynamic and visually captivating art using Processing's 3D graphics capabilities. It combines randomness, trigonometry, and a variety of shapes to create an ever-changing visual experience for the viewer. The result is a digital artwork that continuously evolves and captivates the audience's attention.