MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRIIAL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi MicroelectronicăDepartamentul Inginerie Software și Automatică

Cuzmin Simion Faf-221

Report

Laboratory work n.4.1

of Computer Graphics

Checked by:

Olga Grosu, *university assistant*

DISA, FCIM, UTM

Chișinău – 2023

Task A

- Do the sketch using the function:
  Exercise 1-8

- The program code with relevant comments:

```
PVector position;  // Declare a PVector for the position
PVector dimensions; // Declare a PVector for the dimensions

void setup() {
  size(400, 400);
  background(51);
  position = new PVector(200, 200);  // Center of the rectangle
  dimensions = new PVector(180, 150); // Half of the rectangle's width and
height
  rectMode(RADIUS);
  rect(position.x, position.y, dimensions.x, dimensions.y);
  // Draw the diagonal lines
  line(position.x - dimensions.x, position.y - dimensions.y, position.x +
dimensions.x, position.y + dimensions.y);
  line(position.x - dimensions.x, position.y + dimensions.y, position.x +
dimensions.x, position.y - dimensions.y);
  // Draw horizontal center line
  line(position.x - dimensions.x, position.y, position.x + dimensions.x,
position.y);
  // Draw vertical center line
  line(position.x, position.y - dimensions.y, position.x, position.y +
dimensions.y);
}

void draw() {
  strokeWeight(2);

  // Draw an arc in the first quadrant
  stroke(0, 0, 255);
```

```
  arc(position.x, position.y, dimensions.x * 2, dimensions.y * 2, -PI/2, 0);

  // Draw an arc in the middle of quadrant II to the end of quadrant III
  stroke(255, 0, 0);
  arc(position.x, position.y, dimensions.x * 2, dimensions.y * 2, PI/2, PI + PI/4,
CHORD);

  // Draw an arc at the end of quadrant IV and the middle of quadrant III
  stroke(0, 255, 0);
  arc(position.x, position.y, (dimensions.x - 10) * 2, dimensions.y * 2, 0, PI +
PI/4, PIE);
}
```

Ex2

```
PVector position; // PVector for the walker's position

void setup() {
  size(400, 400);
  background(255);
  position = new PVector(width / 2, height / 2); // Start at the center of the
canvas
}

void draw() {
  stroke(0);
  point(position.x, position.y); // Draw the walker as a point at its position

  // Generate a random vector for the walker to move
  PVector step = PVector.random2D();

  // Scale the step size (you can adjust the step size here)
  step.mult(10);

  // Add the step to the walker's position
  position.add(step);

  // Constrain the walker within the canvas
```

```
  position.x = constrain(position.x, 0, width);
  position.y = constrain(position.y, 0, height);
}
```

Ex3

```
PVector position;
PVector velocity;
float radius = 25;
float xdir, ydir, zdir;
float boxSize = 200;

void setup() {
  size(400, 400, P3D);
  position = new PVector(random(radius, width - radius), random(radius, height
- radius), random(radius, boxSize - radius));
  velocity = new PVector(random(-2, 2), random(-2, 2), random(-2, 2));
  xdir = 1;
  ydir = 1;
  zdir = 1;
}

void draw() {
  background(220);
  lights();
  translate(width / 2, height / 2, 0);

  position.add(velocity);

  if (position.x > width / 2 - radius || position.x < -width / 2 + radius) {
    velocity.x *= -1;
  }

  if (position.y > height / 2 - radius || position.y < -height / 2 + radius) {
    velocity.y *= -1;
  }

  if (position.z > boxSize / 2 - radius || position.z < -boxSize / 2 + radius) {
```

```
    velocity.z *= -1;
  }

  noStroke();
  fill(255, 0, 0);
  pushMatrix();
  translate(position.x, position.y, position.z);
  sphere(radius);
  popMatrix();
}
```

Ex4

```
void limit(float max) {
    if (mag() > max) {
      normalize();
      mult(max);
    }
}
```

Ex5

```
float carX;
float carSpeed;
float acceleration = 0.1;
float braking = 0.2;
int carWidth = 80;

void setup() {
  size(800, 200);
  carX = 0; // Start the car at the left edge
  carSpeed = 0;
}

void draw() {
  background(220);

  // Draw the road
```

```
  fill(100);
  rect(0, height / 2, width, height / 2);

  // Draw the car
  fill(255, 0, 0);
  rect(carX, height / 2 - 30, carWidth, 20);

  // Update car position
  carX += carSpeed;

  // Accelerate when the up arrow key is pressed
  if (keyPressed && keyCode == UP) {
    carSpeed += acceleration;
  }

  // Brake when the down arrow key is pressed
  if (keyPressed && keyCode == DOWN) {
    carSpeed -= braking;
  }

  // Limit car speed
  carSpeed = constrain(carSpeed, 0, 5);

  // Wrap the car to the left when it reaches the right border
  if (carX > width) {
    carX = -carWidth;
  }
}
```

Ex6

```
float carX;
float carSpeed;
float maxSpeed = 5.0;
float t = 0.0;
float tIncrement = 0.01;

void setup() {
  size(800, 200);
```

```
  carX = width / 2;  // Start the car at the center
  carSpeed = 0;
  noiseSeed(10); // Set a consistent noise seed for repeatability
}

void draw() {
  background(220);

  // Draw the road
  fill(100);
  rect(0, height / 2, width, height / 2);

  // Draw the car
  fill(255, 0, 0);
  rect(carX, height / 2 - 30, 80, 20);

  // Apply Perlin noise to acceleration
  float n = noise(t);
  float acceleration = map(n, 0, 1, -0.1, 0.1); // Adjust the range and scale for
acceleration

  // Increment time for Perlin noise
  t += tIncrement;

  // Apply acceleration to speed
  carSpeed += acceleration;

  // Limit the speed
  carSpeed = constrain(carSpeed, -maxSpeed, maxSpeed);

  // Wrap the car to the left when it reaches the right border
  if (carX > width) {
    carX = -80; // Car width is 80
  }

  // Wrap the car to the right when it disappears to the left
  if (carX < -80) {
    carX = width;
  }
```

```
  // Brake when the down arrow key is pressed
  if (keyPressed && keyCode == DOWN) {
    carSpeed -= 0.2;
  }

  // Accelerate when the up arrow key is pressed
  if (keyPressed && keyCode == UP) {
    carSpeed += 0.2;
  }

  // Update car position
  carX += carSpeed;
}
```

Ex7

```
PVector v = new PVector(1, 5);
PVector u = PVector.mult(v, 2); // Multiply v by 2
PVector w = PVector.sub(v, u);  // Subtract u from v
w.div(3);                       // Divide w by 3
```

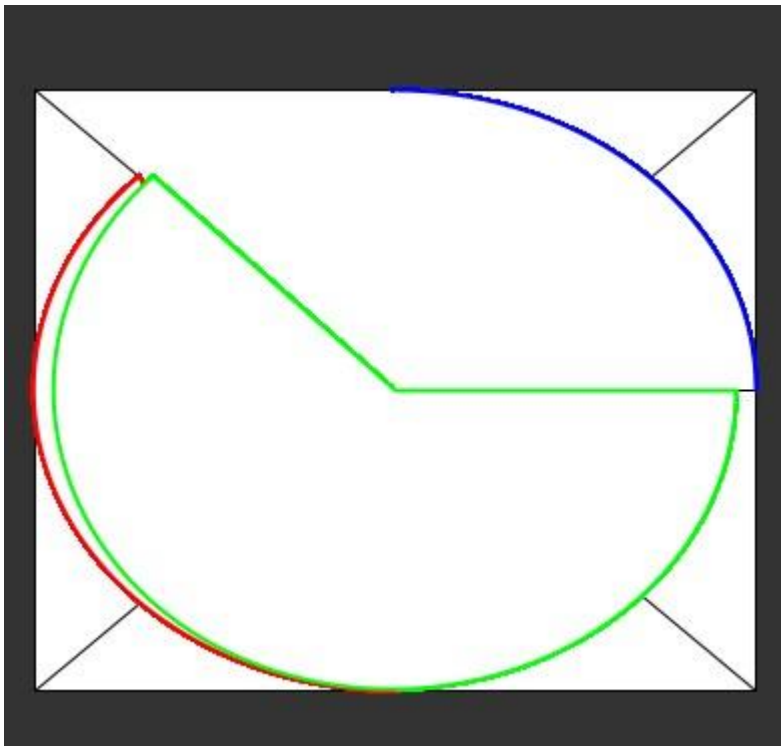
Ex8

```
Mover[] movers = new Mover[20];

void setup() {
  size(640, 360);
  background(255);
  for (int i = 0; i < movers.length; i++) {
    movers[i] = new Mover();
  }
}

void draw() {
  background(255);
  for (int i = 0; i < movers.length; i++) {
    movers[i].update();
```

```
    movers[i].checkEdges();
    movers[i].display();
  }
}

class Mover {
  PVector location;
  PVector velocity;
  PVector acceleration;
  float topspeed;
  float radius = 32; // Adjust the size of the movers

  Mover() {
    location = new PVector(random(width), random(height));
    velocity = new PVector(0, 0);
    topspeed = 4;
  }

  void update() {
    PVector mouse = new PVector(mouseX, mouseY);
    PVector dir = PVector.sub(mouse, location);

    float d = dir.mag();
    float mappedAcceleration = map(d, 0, width, 0, 0.5);

    dir.normalize();
    dir.mult(mappedAcceleration);
    acceleration = dir;

    velocity.add(acceleration);
    velocity.limit(topspeed);

    location.add(velocity);
  }

  void display() {
    stroke(0);
    fill(175);
```

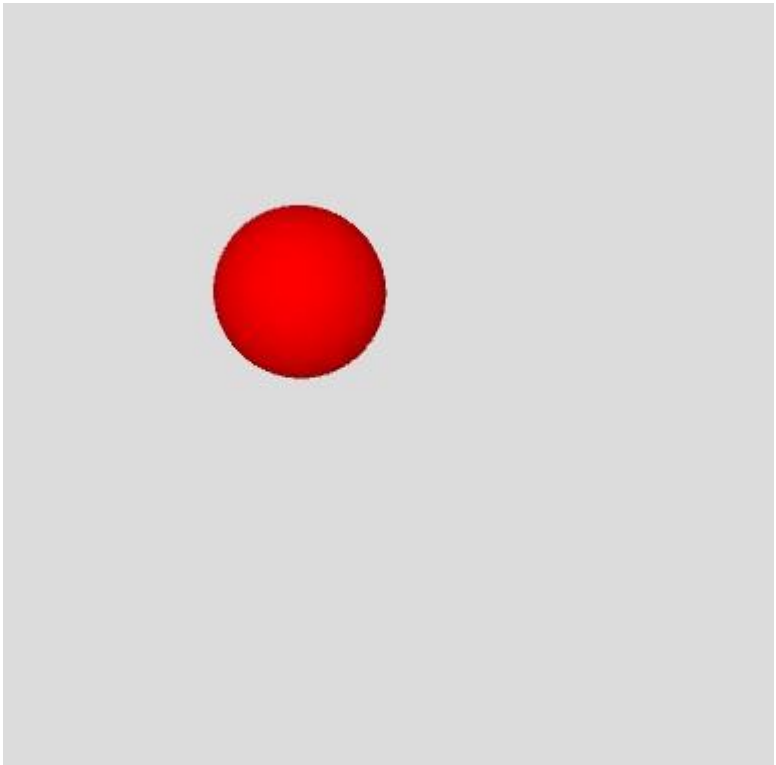
```
    ellipse(location.x, location.y, radius * 2, radius * 2); // Use the 'radius'
variable for size
  }

  void checkEdges() {
    if (location.x > width) {
      location.x = 0;
    } else if (location.x < 0) {
      location.x = width;
    }

    if (location.y > height) {
      location.y = 0;
    } else if (location.y < 0) {
      location.y = height;
    }
  }
}
```

- Screen printing of program execution:

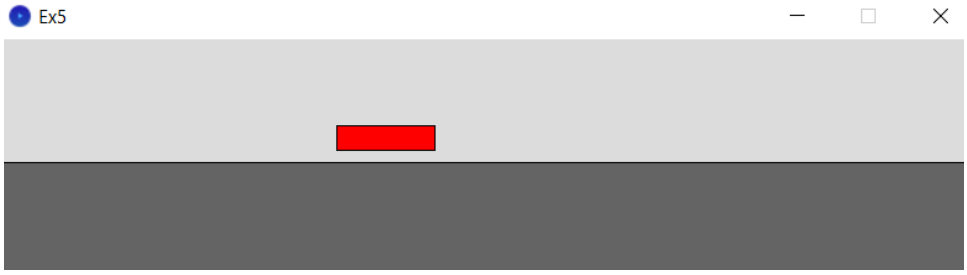


Ex1

Ex2

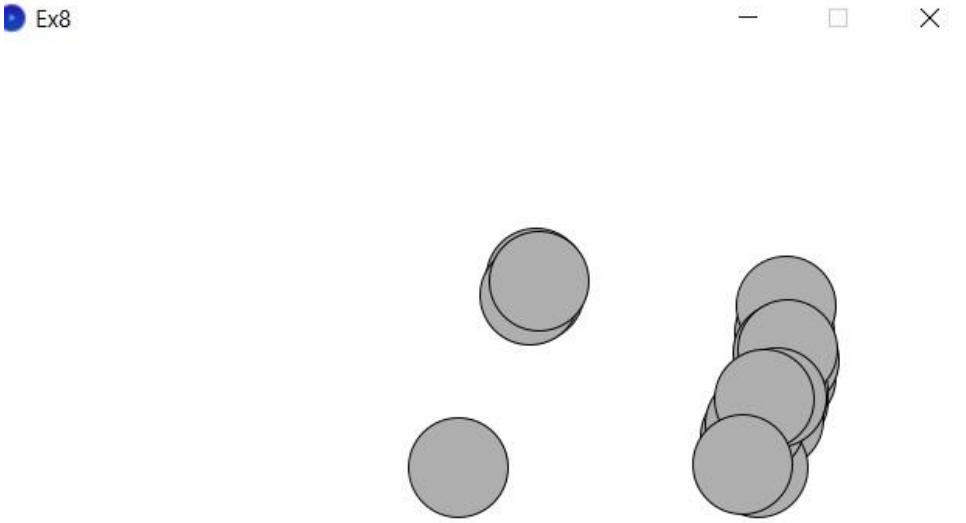


Ex3



Ex5

Ex6



Ex8

- Student's conclusions and reflections:

After this exercise i can conclude that i develop my skills in processing, regarding perlin noise tool, aplying it in different forms and different ways. As well i used mouse and arrows to move my object properly. I think this exercise were very usefull for me.