

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică Departamentul Inginerie Software și Automatică**

**Gusev Roman FAF-222**

Report

*Laboratory work n.4*

*Tasks 1.1 – 1.8*

***of Computer Graphics***

Checked by:

**Olga Grosu,** *university assistant*

DISA, FCIM, UTM

**Chișinău – 2023**

**1. Purpose of the Laboratory Work**

I had to accomplish and solve several exercises before the midterm I. Those exercises are mainly focused on the usage of Vectors, that in Processing are called PVectors, and accommodation in working with them. Those exercises covered the initialization of a PVector, work with their 2D space: x and y values, that can be further extended to 3D space. Another exercises covered the operations with Vectors, such as multiplication, addition, substraction and other operations. Also, those exercises covered several physics concepts, such as acceleration, velocity and location in 2D space.

**2. Condition of the Laboratory Work**

1.1) Find something you’ve previously made in Processing using separate x and y variables and use PVectors instead.

1.2) Take one of the walker examples from the introduction and convert it to use PVectors.

1.3) Extend the bouncing ball with vectors example into 3D. Can you get a sphere to bounce around a box?

1.4) Write the limit() function for the PVector class.

void limit(float max) {

if (\_\_\_\_\_\_\_ > \_\_\_\_\_\_\_) {

\_\_\_\_\_\_\_\_\_();

\_\_\_\_(max);

}

}

1.5) Create a simulation of a car (or runner) that accelerates when you press the up key and brakes when you press the down key.

1.6) Referring back to the Introduction, implement acceleration according to Perlin noise.

1.7) Translate the following pseudocode to code using static or non-static functions where appropriate.

The PVector v equals (1,5).

The PVector u equals v multiplied by 2.

The PVector w equals v minus u.

Divide the PVector w by 3.

PVector v = new PVector(1,5);

PVector u = \_\_\_\_\_\_\_\_.\_\_\_\_\_(\_\_,\_\_);

PVector w = \_\_\_\_\_\_\_\_.\_\_\_\_\_(\_\_,\_\_);

\_\_\_\_\_\_\_\_\_\_\_;

1.8) Try implementing the above example with a variable magnitude of acceleration, stronger when it is either closer or farther away.

**3. The program code**

**1.1)**

/\* Exercise 1.1 \*/

// Student: Gusev Roman

// Group: FAF-222

// Create Vectors for location and velocity

// of the square and size of the square

PVector location;

PVector velocity;

int sizeSquare;

void setup() {

// Canvas size and background color

size(400, 400);

background(255);

// Initialize Vectors for location

// and velocity and size of the ball

sizeSquare = 25;

location = new PVector(random(sizeSquare, width - sizeSquare), random(sizeSquare, height - sizeSquare));

velocity = new PVector(3, 1.1);

}

void draw() {

// Reset the background every iteration

background(255);

// Move ball in direction

location.add(velocity);

// If ball touches borders, reverse direction

if ((location.x > width - sizeSquare/2) || (location.x < 0 + sizeSquare/2)) {

velocity.x \*= -1;

fill(color(random(0, 255), random(0, 255), random(0, 255)));

}

if ((location.y > height - sizeSquare/2) || (location.y < 0 + sizeSquare/2)) {

velocity.y \*= -1;

fill(random(0, 255), random(0, 255), random(0, 255));

}

rectMode(CENTER);

square(location.x, location.y, sizeSquare);

}

**1.2)**

**Main Class:**

/\* Exercise 1.2 \*/

// Student: Gusev Roman

// Group: FAF-222

// Convert Perlin Noise walker class

// to using PVectors

Walker walker;

PVector locationBall = new PVector();

void setup() {

size(500, 500);

walker = new Walker(locationBall);

}

void draw() {

background(255);

walker.step();

fill(255, 0, 0);

circle(locationBall.x, locationBall.y, 25);

}

**Walker Class:**

class Walker {

PVector location;

PVector time = new PVector();

Walker(PVector location) {

time.x = 0;

time.y = 10000;

this.location = location;

}

void step() {

location.x = map(noise(time.x), 0, 1, 0, width);

location.y = map(noise(time.y), 0, 1, 0, height);

time.x += 0.01;

time.y += 0.01;

}

}

**1.3)**

/\* Exercise 1.3 \*/

// Student: Gusev Roman

// Group: FAF-222

// Create Vectors for location and velocity

// of the box and size of the box

PVector location;

PVector velocity;

int sizeBox;

// Draw the canvas borders in 3D

void drawBorders() {

stroke(0); // Set the stroke color to black

// Lines along the edges of the canvas

line(0, 0, 0, 0, 0, -width);

line(width, 0, 0, width, 0, -width);

line(width, height, 0, width, height, -width);

line(0, height, 0, 0, height, -width);

line(0, 0, -width, width, 0, -width);

line(width, 0, -width, width, height, -width);

line(width, height, -width, 0, height, -width);

line(0, height, -width, 0, 0, -width);

}

void setup() {

// Canvas size and background color

size(400, 400, P3D);

background(255);

// Initialize Vectors for location

// and velocity and size of the box

sizeBox = 50;

location = new PVector(random(sizeBox, width - sizeBox), random(sizeBox, height - sizeBox), random(-sizeBox, sizeBox - width ));

velocity = new PVector(3, 1.1, 2);

fill(color(random(0, 255), random(0, 255), random(0, 255)));

}

void draw() {

// Reset the background every iteration

background(255);

drawBorders();

// Move box in direction

location.add(velocity);

if ((location.x > width - sizeBox/2) || (location.x < 0 + sizeBox/2)) {

velocity.x \*= -1;

fill(color(random(0, 255), random(0, 255), random(0, 255)));

}

if ((location.y > height - sizeBox/2) || (location.y < 0 + sizeBox/2)) {

velocity.y \*= -1;

fill(random(0, 255), random(0, 255), random(0, 255));

}

if ((location.z < - height + sizeBox/2) || (location.z > -sizeBox/2)) {

velocity.z \*= -1;

fill(random(0, 255), random(0, 255), random(0, 255));

}

rectMode(CENTER);

translate(location.x, location.y, location.z);

box(sizeBox);

}

**1.4)**

**Main Class:**

// Exercise 1.4

// Student: Gusev Roman

// Group: FAF-222

Mover mover;

void setup() {

size(400, 400);

mover = new Mover();

}

void draw() {

background(255);

mover.update();

mover.checkEdges();

mover.display();

System.out.println(mover.velocity);

}

**Mover Class:**

class Mover {

PVector location;

PVector velocity;

PVector acceleration;

float maxSpeed;

Mover() {

location = new PVector(width/2, height/2);

velocity = new PVector(0, 0);

acceleration = new PVector(-0.001, 0.01);

maxSpeed = 10;

}

void limit(float maxSpeed) {

if (this.acceleration.mag() > maxSpeed) {

this.acceleration.normalize();

this.acceleration.mult(maxSpeed);

}

}

void update() {

velocity.add(acceleration);

velocity.limit(maxSpeed);

location.add(velocity);

}

void display() {

fill(255, 0, 0);

square(location.x, location.y, 25);

}

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;

}

}

}

**1.5)**

**Main Class:**

// Exercise 1.5

// Student: Gusev Roman

// Group: FAF-222

PVector car1Location;

PVector car2Location;

Car car1;

Car car2;

void setup() {

size(200, 400);

car1Location = new PVector(150, height/2);

car2Location = new PVector(25, height/3);

car1 = new Car(-0.01, car1Location);

car2 = new Car(0.01, car2Location);

fill(0);

rect(80, 400, 400, 200);

}

void draw() {

background(0);

fill(255, 255, 0);

rect(95, 0, 10, 400);

car1.update();

car1.checkEdges();

car1.display();

car2.update();

car2.checkEdges();

car2.display();

}

**Car Class:**

class Car {

PVector location;

PVector velocity;

PVector acceleration;

PVector deceleration;

float accelerationValue;

float maxSpeed;

float minSpeed;

boolean isAccelerating;

Car(float accelerationValue, PVector location) {

this.location = location;

velocity = new PVector(0, 0);

this.accelerationValue = accelerationValue;

acceleration = new PVector(0, this.accelerationValue);

deceleration = new PVector(0, -this.accelerationValue);

maxSpeed = 10;

minSpeed = 0;

isAccelerating = false;

}

void update() {

if (keyPressed && ((key == CODED && keyCode == UP) || key == 'w')) {

velocity.add(acceleration);

velocity.limit(maxSpeed);

}

else if (keyPressed && ((key == CODED && keyCode == DOWN) || key == 's')) {

velocity.add(deceleration);

if (velocity.mag() < 0.1) {

velocity.limit(minSpeed);

}

else {

velocity.limit(max(velocity.mag(), minSpeed));

}

}

location.add(velocity);

}

void display() {

fill(255, 0, 0);

square(location.x, location.y, 25);

}

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;

}

}

}

**1.6)**

**Main Class:**

// Exercise 1.6

// Student: Gusev Roman

// Group: FAF-222

Walker walker;

PVector locationBall = new PVector(500, 355);

void setup() {

size(400, 500);

walker = new Walker(locationBall);

//frameRate(10);

}

void draw() {

background(255);

walker.step();

fill(255, 0, 0);

circle(walker.location.x, walker.location.y, 25);

}

**Walker Class:**

class Walker {

PVector location;

PVector velocity;

PVector acceleration;

float maxSpeed;

float accelerationX;

float accelerationY;

PVector time = new PVector();

Walker(PVector location) {

this.location = location;

this.velocity = new PVector();

this.acceleration = new PVector();

this.maxSpeed = 5; // Reduced max speed for smoother wrapping

this.time.x = 0;

this.time.y = 10000;

}

void step() {

accelerationX = map(noise(time.x), 0, 1, -0.2, 0.2);

accelerationY = map(noise(time.y), 0, 1, -0.2, 0.2);

acceleration.set(accelerationX, accelerationY);

velocity.add(acceleration);

velocity.limit(maxSpeed);

location.add(velocity);

time.x += 0.01;

time.y += 0.01;

// Wrap the walker to the opposite side if it goes out of bounds

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;

}

}

}

**1.7)**

// Exercise 1.7

// Student: Gusev Roman

// Group: FAF-222

PVector v;

void setup() {

size(400, 400);

background(255);

v = new PVector(100, 125);

rectMode(CENTER);

translate(10, 10);

line(0,0,v.x,v.y);

PVector u = PVector.mult(v, 2);

translate(15, 0);

stroke(255, 0, 0);

line(0, 0, u.x, u.y);

PVector w = PVector.sub(v, u);

translate(width/2, height/2);

stroke(0, 255, 0);

line(0, 0, w.x, w.y);

w.div(3);

translate(width/4, 0);

stroke(0, 255, 100);

line(0, 0, w.x, w.y);

}

**1.8)**

**Main Class:**

// Exercise 1.8

// Student: Gusev Roman

// Group: FAF-222

Mover mover;

void setup() {

size(1000, 1000);

background(255);

mover = new Mover();

}

void draw() {

background(255);

mover.update();

mover.display();

}

**Mover Class:**

class Mover {

PVector location;

PVector velocity;

PVector acceleration;

float maxSpeed;

Mover() {

location = new PVector(width/2, height/2);

velocity = new PVector(0, 0);

acceleration = new PVector(0, 0);

maxSpeed = 10;

}

void update() {

PVector mouseLocation = new PVector(mouseX, mouseY);

PVector direction = PVector.sub(mouseLocation, location);

// calculate distance from mouse to ball

float distance = direction.mag();

direction.normalize();

// determine the force of the attraction based on

// distance

float strength = map(distance, 0, height, 1, 30);

// divide by that force strength

direction.div(strength);

acceleration = direction;

velocity.add(acceleration);

velocity.limit(maxSpeed);

location.add(velocity);

}

void display() {

stroke(0);

fill(175);

circle(location.x, location.y, 25);

textSize(50);

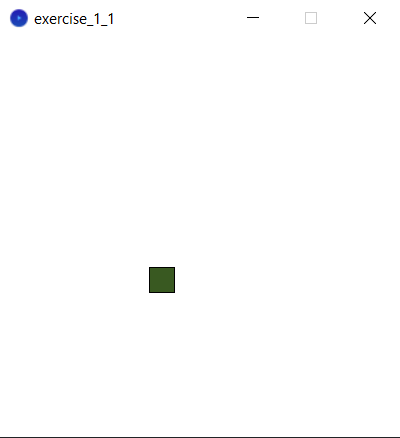
fill(0);

text("Acceleration: " + acceleration.mag(), 50, 50);

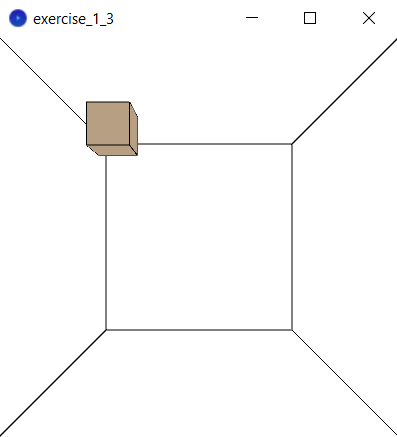
}

}

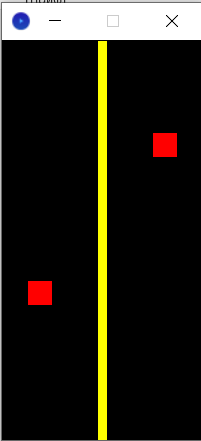
**4. Screen printing of program execution**

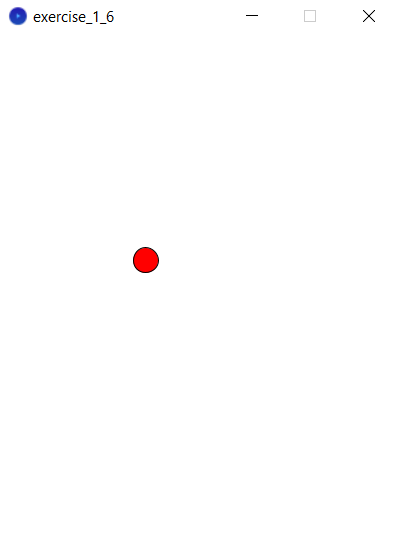
****

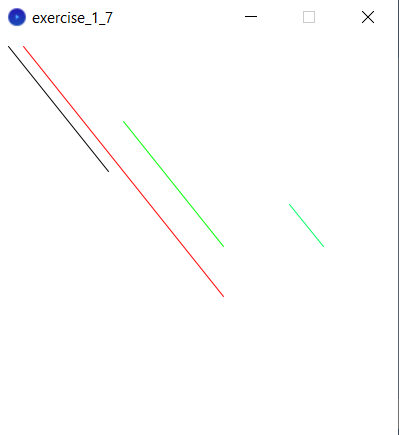
****

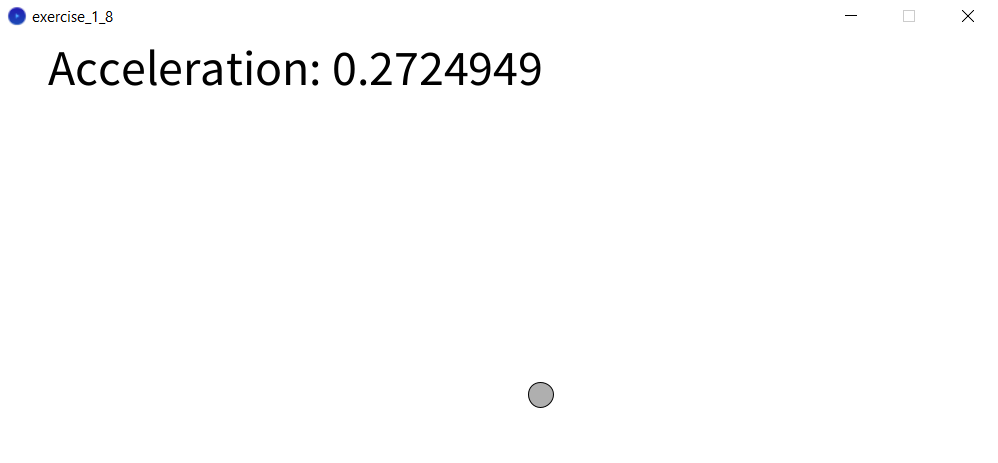
****

****









**5. Conclusion**

By the end of this Laboratory Work nr. 4 exercises 1.1-1.8, I have familiarized myself with several useful notions in the Processing – Pvectors, operations with Vectors in Processing, physics applications of Vectors, Random generation – Perlin Noise algorithm and others. Also, I have encountered new methods that previously have not been known for me, such as the mentioned above PVectors and constructors, that initialize the objects. I have encountered several problems, but quickly managed to solve them. I can say that this part of the laboratory work has been done completely and finished with success.