

**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.5*

*Point 1*

***of Computer Graphics***

Checked by:

**Olga Grosu,** *university assistant*

DISA, FCIM, UTM

**Chișinău – 2023**

**1. Purpose of the Laboratory Work**

In point A of laboratory Work nr. 5, I had to add to sketch Point B from previous Laboratory Work (ecosystem) force. Main purpose of this point is to make me familiar with Force theory and its application in graphics, specifically for an ecosystem with insects and enhancing the behavior of objects such as insects and predators

**2. Condition of the Laboratory Work**

Incorporate the concept of forces into your ecosystem (you can use the ecosystem from Task 4 point B or create something new). Try introducing other elements into the environment (food, a predator) for the creature to interact with. Does the creature experience attraction or repulsion to things in its world? Can you think more abstractly and design forces based on the creature’s desires or goals?

**3. The program code**

**Main class:**

// Laboratory Work 5

// Point 1

// Student: Gusev Roman

// Group: FAF-222

ArrayList<Insect> insects = new ArrayList<Insect>();

ArrayList<Food> foods = new ArrayList<Food>();

ArrayList<Predator> predators = new ArrayList<Predator>();

int numInsects = 5;

int numPredators = 5;

void setup() {

size(1000, 1000);

for (int i = 0; i < numInsects; i++) {

insects.add(new Insect());

}

for (int i = 0; i < numPredators; i++) {

predators.add(new Predator());

}

}

void draw() {

background(220);

for (Predator predator : predators) {

PVector totalAttraction = new PVector(0, 0);

for (Insect insect : insects) {

PVector force = predator.attract(insect);

totalAttraction.add(force);

}

predator.applyForce(totalAttraction);

predator.update();

predator.display();

}

// Create new food occasionally

if (random(1) < 0.1) {

foods.add(new Food());

}

// Update and display food

for (Food food : foods) {

food.display();

}

// Update and display insects

for (int i = insects.size() - 1; i >= 0; i--) {

Insect insect = insects.get(i);

insect.update();

insect.display();

// Check for collision with food

for (int j = foods.size() - 1; j >= 0; j--) {

Food food = foods.get(j);

if (insect.position.dist(food.position) < 10) {

foods.remove(j); // Remove the food

insect.eat();

}

}

// Check if the insect has reached maximum size

if (insect.isMaxSize()) {

insects.remove(i); // Remove the insect

}

}

// Randomly create new insects

if (random(1) < 0.035) {

insects.add(new Insect());

}

// Randomly create new predators

if (random(1) < 0.002) {

predators.add(new Predator());

}

// Update and display predators

for (Predator predator : predators) {

predator.update();

predator.display();

// Find the nearest insect and attract the predator to it

if (!insects.isEmpty()) {

Insect nearestInsect = findNearestInsect(predator.position);

predator.attractTo(nearestInsect);

}

// Check for collision with insect

for (int j = insects.size() - 1; j >= 0; j--) {

Insect insect = insects.get(j);

if (predator.position.dist(insect.position) < 10) {

insects.remove(j); // Remove the insect

predator.eat();

}

}

// Check for collision with food

for (int j = insects.size() - 1; j >= 0; j--) {

Insect insect = insects.get(j);

if (predator.position.dist(insect.position) < 10) {

insects.remove(j); // Remove the insect

predator.eat();

}

}

// Make insects repel from the predator

for (Insect insect : insects) {

insect.repelFrom(predator);

}

// Check if the predator has reached maximum size

if (predator.isMaxSize()) {

predators.remove(predator); // Remove the predator

}

}

for (int i = 0; i < predators.size(); i++) {

if (predators.get(i).isDead()) {

predators.remove(i);

}

}

textSize(25);

fill(0);

text("Number of Insects: " + insects.size(), 50, 50);

text("Number of Predators: " + predators.size(), 50, 80);

}

Food findNearestFood(PVector insectPosition) {

Food nearestFood = null;

float minDistance = Float.MAX\_VALUE;

for (Food food : foods) {

float distance = insectPosition.dist(food.position);

if (distance < minDistance) {

minDistance = distance;

nearestFood = food;

}

}

return nearestFood;

}

Insect findNearestInsect(PVector predatorPosition) {

Insect nearestInsect = null;

float minDistance = Float.MAX\_VALUE;

for (Insect insect : insects) {

float distance = predatorPosition.dist(insect.position);

if (distance < minDistance) {

minDistance = distance;

nearestInsect = insect;

}

}

return nearestInsect;

}

**Food Class:**

class Food {

PVector position;

Food() {

position = new PVector(random(width), random(height));

}

void display() {

fill(0, 255, 0);

noStroke();

ellipse(position.x, position.y, 10, 10);

}

}

**Insect class:**

class Insect {

PVector position;

PVector velocity;

PVector acceleration;

float maxSpeed = 3;

float maxLimit = 0.1;

float maxSize = 25;

float size = 5;

float mass = 1; // Mass of the insect

Insect() {

position = new PVector(random(width), random(height));

velocity = PVector.random2D();

acceleration = new PVector(0, 0);

}

void update() {

applyJitter();

velocity.add(acceleration);

velocity.limit(maxSpeed);

position.add(velocity);

acceleration.mult(0);

if (position.x > width) position.x = 0;

if (position.x < 0) position.x = width;

if (position.y > height) position.y = 0;

if (position.y < 0) position.y = height;

}

void applyJitter() {

PVector jitter = PVector.random2D();

jitter.mult(0.2);

acceleration.add(jitter);

PVector desired = PVector.sub(position, new PVector(mouseX, mouseY));

float d = desired.mag();

if (d < 100) {

desired.setMag(map(d, 0, 100, 0, maxSpeed));

PVector steer = PVector.sub(desired, velocity);

steer.limit(maxLimit);

acceleration.add(steer);

}

}

void display() {

stroke(0);

fill(127, 127, 0);

ellipse(position.x, position.y, size, size);

fill(255, 255, 0);

float wingLength = 5;

float wingWidth = 2;

pushMatrix();

translate(position.x, position.y);

rotate(velocity.heading());

rect(-wingLength, -wingWidth, wingLength \* 2, wingWidth \* 2);

popMatrix();

}

void eat() {

size += 1;

mass += 2;

maxSpeed -= 0.1;

}

boolean isMaxSize() {

return size >= maxSize;

}

void repelFrom(Predator predator) {

PVector desired = PVector.sub(position, predator.position);

float d = desired.mag();

if (d < 100) { // Adjust the repulsion range

desired.setMag(map(d, 0, 100, 0, maxSpeed));

PVector steer = PVector.sub(desired, velocity);

steer.limit(maxLimit);

acceleration.add(steer);

}

}

}

**Predator class:**

class Predator {

PVector position;

PVector velocity;

PVector acceleration; // Add acceleration

float maxSpeed = 2; // Adjust the maximum speed as needed

float maxLimit = 0.1; // Adjust the maximum force limit

float size = 15;

float mass = 6; // Adjust mass for predator

int hunger = 0; // Initialize hunger level

int maxHunger = 2000; // Set the maximum hunger level

int timeWithoutFood = 0; // Initialize time without eating

int maxTimeWithoutFood = 1000; // Set the maximum time without food (adjust as needed)

int maxSize = 30;

float wingLength = 5;

float wingWidth = 2;

Predator() {

position = new PVector(random(width), random(height));

velocity = PVector.random2D();

acceleration = new PVector(0, 0);

}

void update() {

applyJitter();

// Apply acceleration to velocity

velocity.add(acceleration);

// Limit velocity to maximum speed

velocity.limit(maxSpeed);

// Update position based on velocity

position.add(velocity);

if (position.x > width) position.x = 0;

if (position.x < 0) position.x = width;

if (position.y > height) position.y = 0;

if (position.y < 0) position.y = height;

// Increment the hunger without food

increaseHunger();

// Reset acceleration at the beginning of each frame

acceleration.set(0, 0);

}

void applyJitter() {

PVector jitter = PVector.random2D();

jitter.mult(0.2);

// Apply jitter to acceleration

acceleration.add(jitter);

}

void display() {

stroke(0);

fill(255, 0, 0);

ellipse(position.x, position.y, size, size);

fill(0, 255, 0);

pushMatrix();

translate(position.x, position.y);

rotate(velocity.heading());

rect(-wingLength, -wingWidth, wingLength \* 2, wingWidth \* 2);

popMatrix();

}

void attractTo(Insect insect) {

PVector desired = PVector.sub(insect.position, position);

desired.setMag(maxSpeed); // Set the desired speed

PVector steer = PVector.sub(desired, velocity);

steer.limit(maxLimit); // You can adjust the limit for smoother turning

// Apply attraction force to acceleration

acceleration.add(steer);

}

// Check if the predator dies from hunger

boolean isDead() {

return timeWithoutFood >= maxTimeWithoutFood;

}

void eat() {

size += 1;

mass += 1;

maxSpeed -= 0.1;

hunger = 0;

wingLength += 0.2;

}

void applyForce(PVector force) {

// Adjust the force according to the predator's mass

PVector f = force.copy();

f.div(mass);

acceleration.add(f);

}

PVector attract(Insect insect) {

PVector force = PVector.sub(insect.position, position);

float distance = force.mag();

distance = constrain(distance, 5.0, 25.0);

force.normalize();

float strength = (mass \* insect.mass) / (distance \* distance);

force.mult(strength);

return force;

}

// Implement a hunger mechanism

void increaseHunger() {

hunger++;

if (hunger >= maxHunger) {

// Predator is very hungry, slow it down

timeWithoutFood++;

maxSpeed = 0.5;

}

}

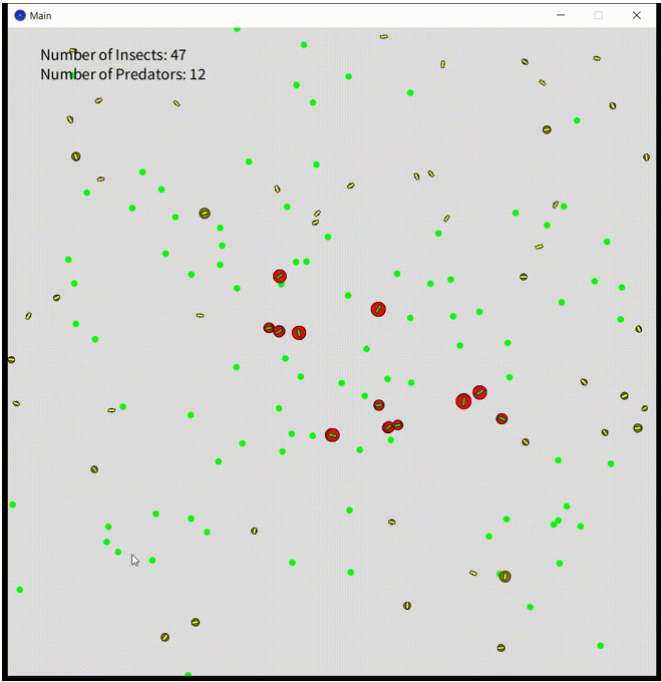
boolean isMaxSize() {

return size >= maxSize;

}

}

**4. Screen printing of program execution**

****

**5. Conclusion**

In point A of Laboratory Work number 5, I successfully integrated the fundamental concept of forces into the existing ecosystem, building upon the foundation laid in Task 4 point B. The primary objective of this stage was to gain a deeper understanding of the principles of Force theory and its practical application in the realm of computer graphics, particularly within the context of an ecosystem inhabited by insects. To achieve this, I introduced various additional elements into the ecosystem, such as food sources and a predator. These elements served as interactive entities, further enriching the behavior of the creatures within the ecosystem. Notably, I explored the concepts of attraction and repulsion, observing how these forces influenced the creatures' movements and interactions. I have not encountered big problems here. As a sum of the above, I can denote that this point I accomplished with success.