MINISTERUL EDUCAȚIEI, CULTURII ȘI

CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi

Microelectronică Departamentul Inginerie Software și

Automatică

Cuzmin Simion Faf-221

Report

Laboratory work n. 5.2

of Computer Graphics

Checked by:

Olga Grosu, *university assistant*

DISA, FCIM, UTM

Chișinău – 2023

Take one of your creatures and incorporate oscillation into its motion. You can use the Oscillator class from Example 3.7 (Chapter 3. Oscillation) as a model. The Oscillator object, however, oscillates around a single point (the middle of the window). Try oscillating around a moving point. In other words, design a creature that moves around the screen according to location, velocity, and acceleration. But that creature isn't just a static shape, it's an oscillating body. Consider tying the speed of oscillation to the speed of motion. Think of a butterfly's flapping wings or the legs of an insect. Can you make it appear that the creature's internal mechanics (oscillation) drive its locomotion? For a sample, check out the "AttractionArrayWithOscillation" example with the code download.

The program code with relevant comments:

```
Creature creature;


void setup() {
  size(800, 600);
  creature = new Creature(width / 2, height
/ 2, 50, 0.02, 50); // 6 wings
}


void draw() {
  background(224,158,118);
  creature.update();
  creature.display();
}


void keyPressed() {
  if (keyCode == UP) {
    creature.setDirection(0, -1);
  } else if (keyCode == DOWN) {
    creature.setDirection(0, 1);
  } else if (keyCode == LEFT) {
```

```
    creature.setDirection(-1, 0);
  } else if (keyCode == RIGHT) {
    creature.setDirection(1, 0);
  } else if (key == 'B' || key == 'b') {
    creature.changeSize(1.1); // Increase
size
  } else if (key == 'S' || key == 's') {
    creature.changeSize(0.9); // Decrease
size
  }
}

void keyReleased() {
  creature.setDirection(0, 0);
}

class Creature {
  PVector center;
  float radius;
  float angle;
```

```
  float angleVelocity;
  float speed;
  PVector acceleration;
  PVector velocity;
  PVector direction;
  ArrayList<Wings> wingsList;

  Creature(float x, float y, float radius,
float angleVelocity, int numWings) {
    center = new PVector(x, y);
    this.radius = radius;
    this.angle = 0;
    this.angleVelocity = angleVelocity;
    this.speed = 2;
    this.acceleration = new PVector(0.1,
0.1);
    this.velocity = new PVector(this.speed,
0);
    this.direction = new PVector(0, 0);
    wingsList = new ArrayList<Wings>();
```

```
    for (int i = 0; i < numWings; i++) {
      wingsList.add(new Wings());
    }
  }

  void update() {
    // Update speed based on acceleration
    this.speed += this.acceleration.mag();

    // Update position based on speed and
direction
    this.velocity.set(this.direction.x *
this.speed, this.direction.y * this.speed);
    this.center.add(this.velocity);

    // Oscillate the angle of the
creature's wings based on speed
    this.angle += this.speed *
this.angleVelocity;

    // Update wings position
```

```
    for (Wings wings : wingsList) {
      wings.update(this.center,
this.angle);
    }

    // Check boundaries and reverse
direction if needed
    if (this.center.x > width -
this.radius) {
      this.center.x = width - this.radius;
    } else if (this.center.x < this.radius)
{
      this.center.x = this.radius;
    }

    if (this.center.y > height -
this.radius) {
      this.center.y = height - this.radius;
    } else if (this.center.y < this.radius)
{
      this.center.y = this.radius;
    }
```

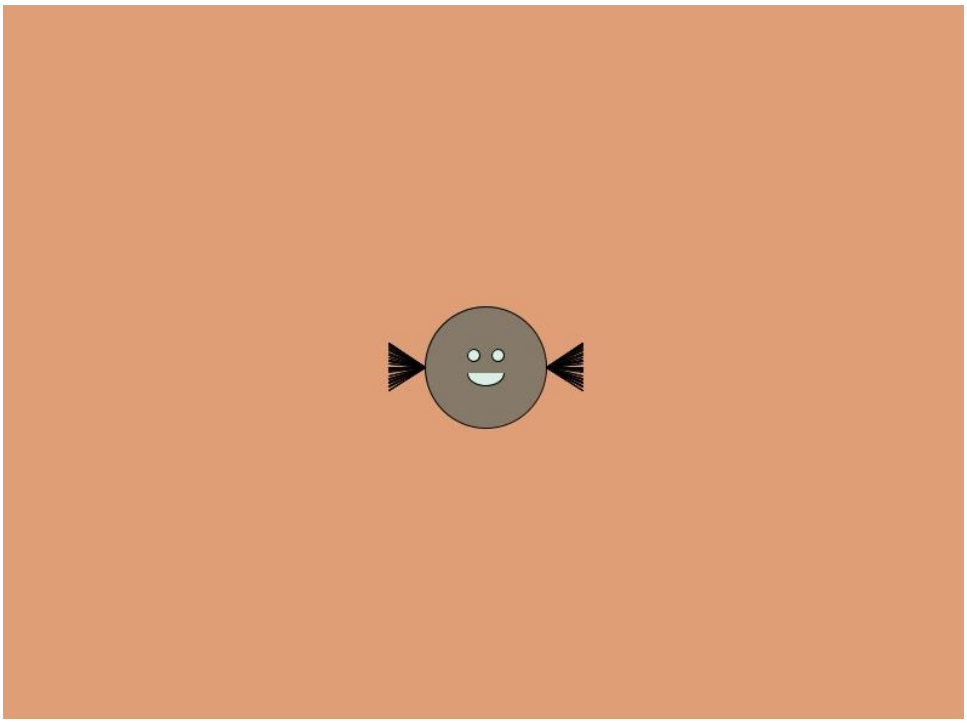
```
  }

  void display() {
    // Draw the creature
    pushMatrix();
    translate(this.center.x,
this.center.y);

    // Draw body
    ellipse(0, 0, this.radius * 2,
this.radius * 2);

    // Draw eyes
    fill(216,240,232);
    ellipse(-10, -10, 10, 10);
    ellipse(10, -10, 10, 10);


    // Draw smile
    arc(0, 5, 30, 20, 0, PI);
```

```
    fill(133,122,106);

    // Draw wings
    for (Wings wings : wingsList) {
      wings.display(this.radius); // Pass
the radius to the display method
    }

    popMatrix();
  }

  void setDirection(float x, float y) {
    direction.set(x, y);
  }

  void changeSize(float scaleFactor) {
    this.radius *= scaleFactor;
    for (Wings wings : wingsList) {
      wings.changeSize(scaleFactor);
    }
```

```
  }
}

class Wings {
  float wingLength;
  float oscillationAmplitude;
  float wingAngleOffset;

  Wings() {
    wingLength = 30;
    oscillationAmplitude = 20;
    wingAngleOffset = random(TWO_PI); //
Randomize initial wing angles
  }

  void update(PVector center, float angle)
{
    // Wings update logic, if needed
  }
```

```
  void display(float creatureRadius) {
    // Oscillate wings
    float wingOscillation = sin(frameCount
* 0.05 + wingAngleOffset) *
oscillationAmplitude;

    // Draw wings
    line(creatureRadius, 0, creatureRadius
+ wingLength, wingOscillation);
    line(-creatureRadius, 0, -
creatureRadius - wingLength,
wingOscillation);
  }

  void changeSize(float scaleFactor) {
    this.wingLength *= scaleFactor;
    this.oscillationAmplitude *=
scaleFactor;
  }
}
```

- Screen printing of program execution:



- Student's conclusions and reflections:

In conclusion, the integration of oscillation into the motion of a creature adds a dynamic and visually engaging dimension to its behavior. By incorporating the Oscillator class and allowing the creature to oscillate around a moving point, we achieve a more lifelike and intriguing movement pattern. The oscillation, tied to the creature's speed of motion, creates a symbiotic relationship between internal mechanics and locomotion, reminiscent of the fluttering wings of a butterfly or the rhythmic leg movements of an insect. This not only enhances the aesthetic appeal of the creature's motion but also simulates a connection between its internal dynamics and external movement, contributing to a more organic and captivating simulation. The "AttractionArrayWithOscillation" example serves as inspiration, showcasing

the potential of this approach to imbue creatures with a sense of life and purpose in their traversal of the screen.