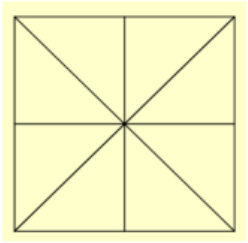
1. Purpose of the Laboratory Work

According to the solved problem, I can formulate the purpose of this laboratory work was: to get used with arc shapes, to formulate better the code and to begin use of such things as: variables, conditionals and loops.

2. Condition of the Laboratory Work

Task 1) I had Point number 4 to do

Task I  Picture 1	Make a sketch: - the window dimension for sketch is 400x400, use background for the window – choose the color of it. - use function setup() and draw() - Draw a rectangle and the line inside like in a Picture 1 - modify the position of x and y coordinates (first two coordinates of the rectangle) in a center of the sketch for drawing it. Next to coordinates rx and ry (half of the width and a half of the height) is in your variant (point). - Declarate x,y flat variable and rx, ry int variable. - Use the key word width and height to determinate the center of the sketch. The thickness of the line is 2 for rectangle and the line. Put all this code (Picture1) inside setup() function. Change the color to draw lines and borders around rectangle.

Point 4	- rx=190 ry=145 (is a half of the width and a half of the height) - Put next code in a function draw()

	- The thickness of the line is 4. Change the color for each shape. a) Draw an arc inside the rectangle in **quadrant III** (see the Picture 2) b) Draw a chord inside the rectangle. - angle to start the chord is **in quadrant IV** - angle to end the chord is in **the middle of quadrant I** c) Draw a Pie inside the rectangle. - angle to start the pie **at the beginning of quadrant II** - angle to end the pie is **the middle of quadrant IV** Change the RX for 7 points before pie function (rx-12)

Task 2)

Make sketch with 2d primitives function and move it.

- you can use random function or increase the coordonate of your sketch combination)

- use VARIABLES, CONDITIONALS, LOOPS function and create your own function and call it in Draw function

3. The program code

Task 1)

```
// Student: Gusev Roman
// Group: FAF-222
// Variant 4

// Variables
int rx;
int ry;
float x;
float y;

void setup() {
  // Setting the size of the canvas
  size(400, 400);

  // Setting the color for background
  background(200);

  // Variables initialization
```

```
  x = width / 2;
  y = height / 2;
  rx = 190;
  ry = 145;

  // Using rectMode with parameter CENTER, to use rect
center x and y
  rectMode(CENTER);

  // Setting the width of the line to 2
  strokeWeight(2);

  // Drawing rect in center of the canvas
  fill(255, 204, 204);
  rect(x, y, 2 * rx, 2 * ry);

  // Drawing lines inside of rect
  line(x - rx, y - ry, x + rx, y + ry);
  line(x - rx, y + ry, x + rx, y - ry);
  line(x - rx, y, x + rx, y);
  line(x, y - ry, x, y + ry);
}

void draw() {
  // Setting the width of the arcs to 4
  strokeWeight(4);

  // a) Drawing an arc in III Quadrant
```

```
  noFill();  // setting empty design for arcs
  stroke(100, 100, 255);  // setting color of the arc
  arc(x,  y,  rx  *  2,  ry  *  2,  HALF_PI,  PI);    //
constructing the arc
  //stroke(0);  // setting color back to black

  // b) Drawing a chord (angle to start the chord in
Quadrant IV and angle to
  //    end the chord in the middle of Quadrant I)

  stroke(255, 0 , 0);  // setting color of the chord
  arc(x, y, rx * 2, ry * 2, -QUARTER_PI, HALF_PI - 0.4,
CHORD);  // constructing the chord
  //stroke(0);  // setting color back to black

  // c) Drawing a Pie (angle to start the pie at the
beginning of quadrant II
  //      and angle to end the pie is the middle of
quadrant IV)

  stroke(255, 100, 100);  // setting color of the pie
  arc(x, y, rx * 2 - 50, ry * 2 - 50, QUARTER_PI, PI +
HALF_PI, PIE);  // constructing the pie
  stroke(0);  // setting color back to black
}
```

Task 2)

```
// Trajectory of Moon and Sun
float horizontalTraj = 400;
float verticalTraj = 250;

// Position on unit circle of Sun and Moon (initial)
float sunAngle = 0;
float moonAngle = PI;

// Color of Day and Night
color dayColor = color(0, 155, 255);
color nightColor = color(0, 15, 15);

// Color of the grass
color dayGrass = color(0, 153, 0);
color nightGrass = color(7, 53, 30);

// Color of the tree
color dayTree = color(196, 92, 0);
color nightTree = color(67, 33, 0);

// Color of the rope
color ropeDay = color(131, 71, 19);
color ropeNight = color(71, 39, 11);

// Position on canvas of Sun and Moon
float sunX;
float sunY;
```

```
float moonX;
float moonY;

// Position of the rope
float ropeNum1InitX = 525;
float ropeNum1InitY = 265;
float ropeNum1EndX;
float ropeNum1EndY;
float ropeNum2X = ropeNum1InitX + 15;
float ropeNum2Y = ropeNum1InitY - 5;
float ropeNum2EndX;
float ropeNum2EndY;

// Swing variables
float ropeLength = 100;
float ropeAmplitude = 20;
float ropeFrequency = 0.01;
float ropeAngle = 0;
boolean swinging = true;
int direction = 1;

// Position of leaves
float leafNum1X = 465;
float leafNum1Y = 180;
float leafNum2X = 600;
float leafNum2Y = 230;
float leafNum3X = 345;
float leafNum3Y = 280;
```

```
// Leaves Shake Variables
float leafShakeX = 0;
float leafShakeY = 0;
float leafShakeRange = 1;

// Function that add Stars during Night
void drawStars(float aux) {
  if (aux > 0.5) {
    for (int i = 0; i < 15; i++) {
      float starX = random(0, width);
      float starY = random(0, 299);
      float starSize = random(2, 4);
      strokeWeight(starSize);
      stroke(255);
      point(starX, starY);
    }
  }
  else if (aux >= 0.3 && aux <= 0.5) {
    for (int i = 0; i < 4; i++) {
      float starX = random(0, width);
      float starY = random(0, 299);
      float starSize = random(2, 4);
      strokeWeight(starSize);
      stroke(255);
      point(starX, starY);
    }
  }
```

```
  strokeWeight(1);
  stroke(0);
}

// Function to draw a tree in the field
void drawTree(float aux, color grassColor) {
  color treeColor = lerpColor(dayTree, nightTree, aux);
  fill(treeColor);
  noStroke();

  // Tree body
  quad(420, 400, 440, 320, 490, 320, 510, 400);
  quad(440, 350, 440, 210, 490, 210, 490, 350);

  // Tree branches
  quad(490, 260, 565, 240, 580, 255, 490, 280);
  quad(565, 240, 590, 210, 610, 210, 580, 255);
  quad(440, 300, 380, 280, 365, 295, 440, 320);
  quad(380, 280, 355, 260, 335, 260, 365, 295);

  // Leaves
  fill(grassColor);
  ellipseMode(CENTER);
  stroke(0);

  leafNum1X += leafShakeX;
  leafNum1Y += leafShakeY;
  leafNum2X += leafShakeX;
```

```
  leafNum2Y += leafShakeY;
  leafNum3X += leafShakeX;
  leafNum3Y += leafShakeY;

  ellipse(leafNum1X, leafNum1Y, 140, 130);
  ellipse(leafNum2X, leafNum2Y, 90, 80);
  ellipse(leafNum3X, leafNum3Y, 90, 80);

  // Function to draw the Swing
  drawSwing(aux);

  // Reset stroke color and strokeWeight
  stroke(0);
  strokeWeight(1);

  // Change leaf position
  leafShakeX = random(-leafShakeRange, leafShakeRange);
  leafShakeY = random(-leafShakeRange, leafShakeRange);
}

// Function to draw the Swing on the branch
void drawSwing(float aux) {
  strokeWeight(3);
  color swingColor = lerpColor(ropeDay, ropeNight, aux);
  stroke(swingColor);

  if (swinging) {
    // Calculate where Rope ends
```

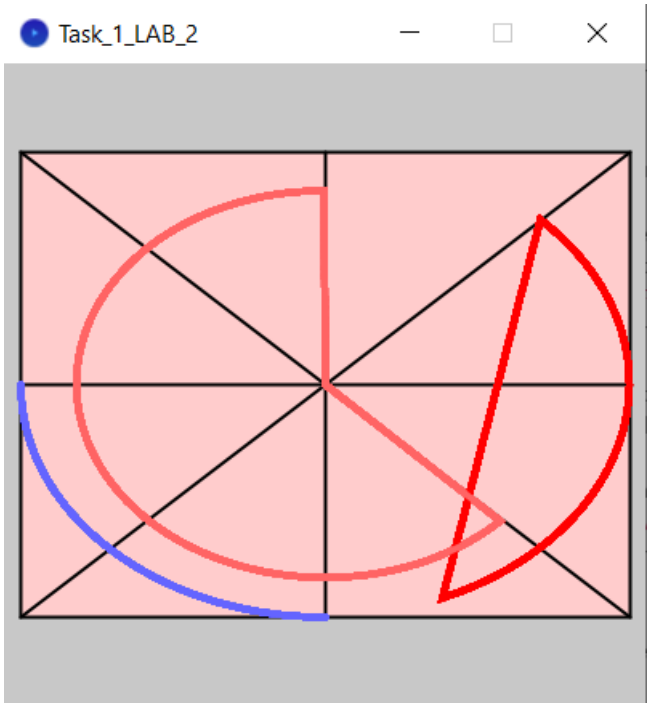
```
    ropeNum1EndX = ropeNum1InitX + ropeLength *
sin(ropeAngle);
    ropeNum1EndY = ropeNum1InitY + ropeLength *
cos(ropeAngle) - 15;
    ropeNum2EndX = ropeNum2X + ropeLength *
sin(ropeAngle);
    ropeNum2EndY = ropeNum2Y + ropeLength *
cos(ropeAngle);

    // Draw the rope
    line(ropeNum1InitX, ropeNum1InitY, ropeNum1EndX,
ropeNum1EndY);
    line(ropeNum2X, ropeNum2Y, ropeNum2EndX,
ropeNum2EndY);

    // Draw the plank
    fill(swingColor); // Use the same color as the rope
    noStroke();
    rect(ropeNum1EndX - 20, ropeNum1EndY, 50, 20);

    // Update the rope angle for swinging motion
    ropeAngle += ropeFrequency * direction;

    // Reverse the swing direction when reaching 15
degrees
    if (ropeAngle >= radians(15) || ropeAngle <= -
radians(15)) {
      direction *= -1;
```

```
    }
  }
}

void setup() {
  // Setting up the sizes of canvas
  size(800, 600);
}

void draw() {
  // Sun and Moon position being calculated
  sunX = width / 2 + horizontalTraj * cos(sunAngle);
  sunY = height / 2 + verticalTraj * sin(sunAngle);
  moonX = width / 2 + horizontalTraj * cos(moonAngle);
  moonY = height / 2 + verticalTraj * sin(moonAngle);

  // Changing the color of sky based on position of Sun
and Moon
  float aux = map(sunY, height / 2 - verticalTraj, height
/ 2 + verticalTraj, 0, 1);
  color bgColor = lerpColor(dayColor, nightColor, aux);
  background(bgColor);

  // Sun
  fill(255, 255, 0);
  ellipse(sunX, sunY, 50, 50);

  // Function drawStars()
```

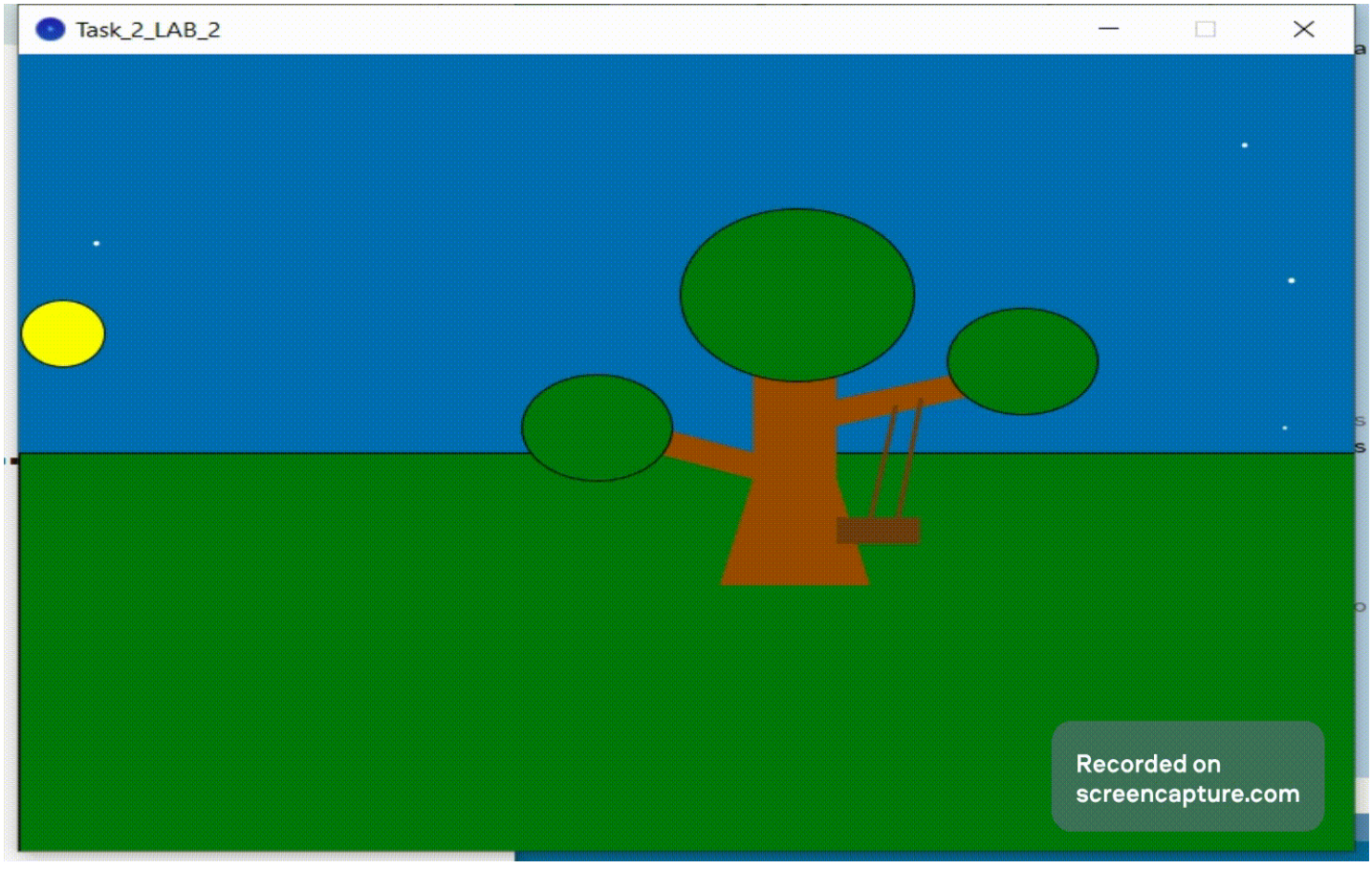
```
  drawStars(aux);

  // Moon
  fill(180);
  ellipse(moonX, moonY, 50, 50);

  // Changing the color of grass based on positions of
Sun and Moon
  color grassColor = lerpColor(dayGrass, nightGrass,
aux);
  // Horizon
  fill(grassColor);
  rect(0, 300, 800, 350);
  // Updating the angles for motion on the trajectory
(speed of Sun and Moon)
  sunAngle += radians(1);
  moonAngle += radians(1);
  // Drawing a tree in the field
  drawTree(aux, grassColor);
}
```

4. Screen printing of program execution

Task 1)



Task 2)



5. Conclusion

By the end of this Laboratory Work nr. 2, I accomplished given task exactly with the requirements. I did task 1 during the class, I accomplished it easily. I used several shapes and initialized variables. Also, I used several colors to distinct every element: pie, arc, chord etc. The most difficult part was to figure how the start and end angles of arcs works. Task 2 was much more difficult, it contains several new functions, especially loops, conditionals and variables initialized outside the methods. I used several shapes, such as rect, circles, ellipses. I also created several auxiliar functions that made draw() function much more shorter and easier to read. I did a scene of a plain with Day Night cycle and a tree that shakes by the wind and a swing that is disturbed by the same wind. I accomplished those tasks with the use of for loops, conditionals and variable change, that is executed every frame. The most difficult part of this task was to understand that draw function is executed in a while loop, but once I understood that, it became much easier to do.