Name: Gustavo Lessa
Date: 05/02/2017
Module: MFC2016
Lecturer: Carole McGloughlin
Group: A
Student number: 2016104

Christmas Project

Robot Simulator

1) My code uses if conditional to determine if the robot's front vertex is at the boundaries of my screen. If positive, the robot turns 5 times to the left (doing a circle on the spot at the corner and turning left) and then goes forward again. This if statement is inserted in another if statement, that prevents collision:

```
if (//there's empty space ahead){
   if((direction==1&&((x1+speed)>=parent.width-
   1))||(direction==2&&((y1+speed)>= parent.height-1))||(direction==3&&(x1-
   speed)<=1)||(direction==4&&((y1-speed)<= 1))){
                        turnLeft();
                        turnLeft();
                        turnLeft();
                        turnLeft();
                        turnLeft();
    }
    moveForward();
} else {
    turnLeft();
```

2) While moving forward, the triangle's vertices are simply being translated using the "speed" value. The moveForward() method considers the direction to which the robot is facing to determine white variables are going to be changed (X or Y). When the boundaries or other objects are almost hit, the turnLeft() method is called. This method rotates the robot by the following steps:

   a. Calculates the translation rate needed to translate the approximate center of the triangle to the origin.
   b. Translates all points by the rate calculated.
   c. Rotates all points 90° counterclockwise through the origin: vertex (x, y) becomes (y, -x).
   d. Translates all points back to the correct position by using the inverse translation rate.

   After rotated, the robot moves forward again, keeping it patrolling the screen.

3) Bob moves using a method called randomWalk().The first if statement considers that it's reaching a boundary. If positive, it sets a Boolean *hit* to true. The second if statement considers that if the Boolean is false, the robot moves forward. If false, it moves the robot backwards and then rotates it a random number of times (between 0 and 10) and to a random direction (random between 0 and 1, using an if statement to determine that 0 is left and 1 is right). Then it sets *hit* as false.

```
    public void randomWalk(){
            if((direction==1&&((x1+speed)>=parent.width-
1))||(direction==2&&((y1+speed)>= parent.height-1))||(direction==3&&(x1-
speed)<=1)||(direction==4&&((y1-speed)<= 1))){
                        hit = true;
            }
```

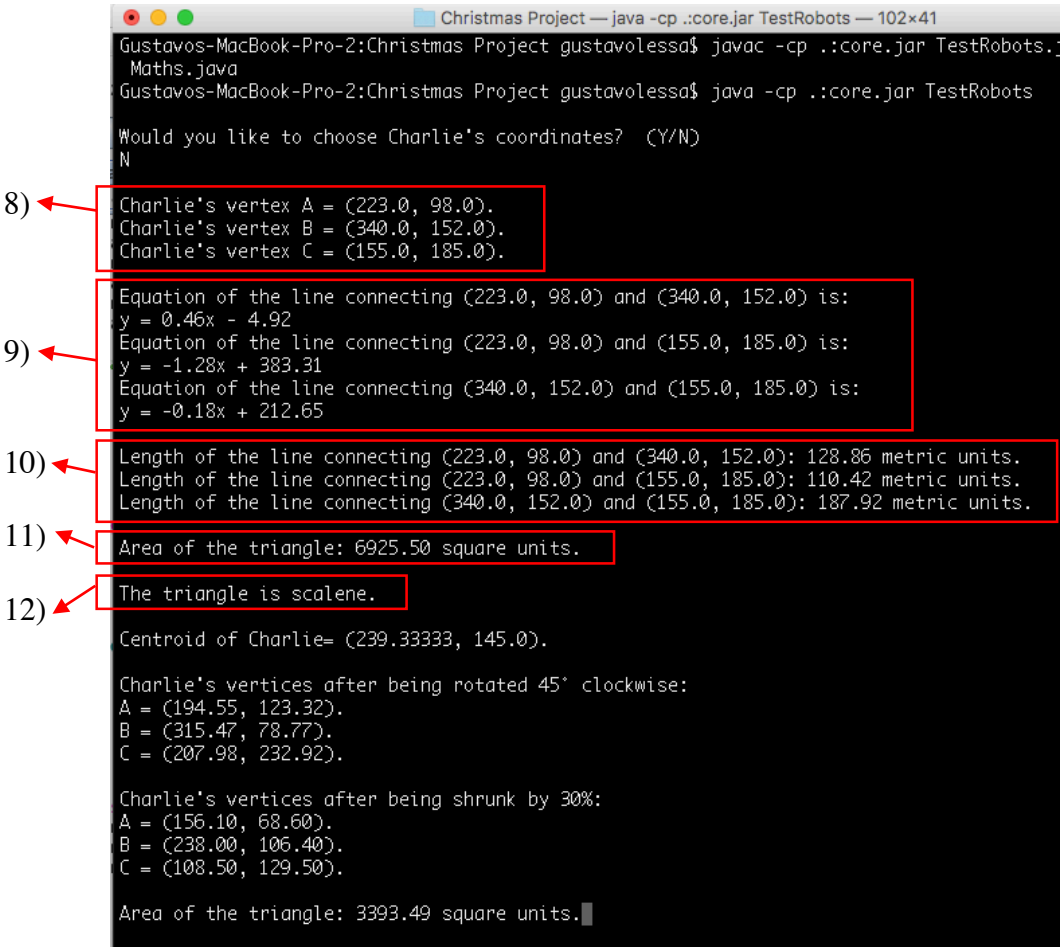
```
        if(hit == false){
                moveForward();
        } else if (hit == true){
                moveBackwards();
                k++;
                if(k >= 8){
                        int turns = parent.round(parent.random(10));
                        for(int i = 0; i==turns; i++){
                                int side = parent.round(parent.random(1));
                                if (side == 0){
                                        turnLeft();
                                        hit = false;
                                        k = 0;
                                } else {
                                        turnRight();
                                        hit = false;
                                        k = 0;
                                }
                        }
                }
        }
}
```

4) If the robot is going to the right AND its x1 coordinate + speed is equal or higher than the screen width OR if the robot is going down AND its y1 coordinate + speed is equal or higher than the screen height OR if the robot is going to the left AND its x1 coordinate + speed is equal or lower than 0 OR if the robot it going up AND its y1 coordinate + speed is equal or lower than 0, THEN hit is true.
If hit is false, THEN the robot goes forward.
If hit is true, THEN the robot goes backwards AND increments a counter by 1 that started at 0.
If this counter is equal or higher than 8 THEN the robot turns a random number of times to a random direction, hit is set to false and the counter is set to 0.

5) A = robot is going to the right.
B = x1 coordinate + speed is equal or higher than the screen width.
C = robot is going down
D = y1 coordinate + speed is equal or higher than the screen height.
E = the robot is going to the left
F = x1 coordinate + speed is equal or lower than 0
G = the robot it going up
H = y1 coordinate + speed is equal or lower than 0
I = it's a hit.
J = robot goes forward
K = the robot goes backwards
L = increments a counter by 1
M = counter is equal or higher than 8.
N = robot turns a random number of times to a random direction, set hit as false, and resets counter to 0.

I = (A ∧ B) v (C ∧ D) v (E ∧ F) ∧ (G ∧ H).
I ➔K ∧ L
~I ➔ J
M ➔ N

The following output comprehends information related to questions 8, 9, 10, 11, 12, 18, 19 and 20.



To create Charlie, the program randomly generates vertex A using as limits the screen length and width. The other vertices are also randomly generated considering A and some boundaries not to create an oddly shaped Charlie.

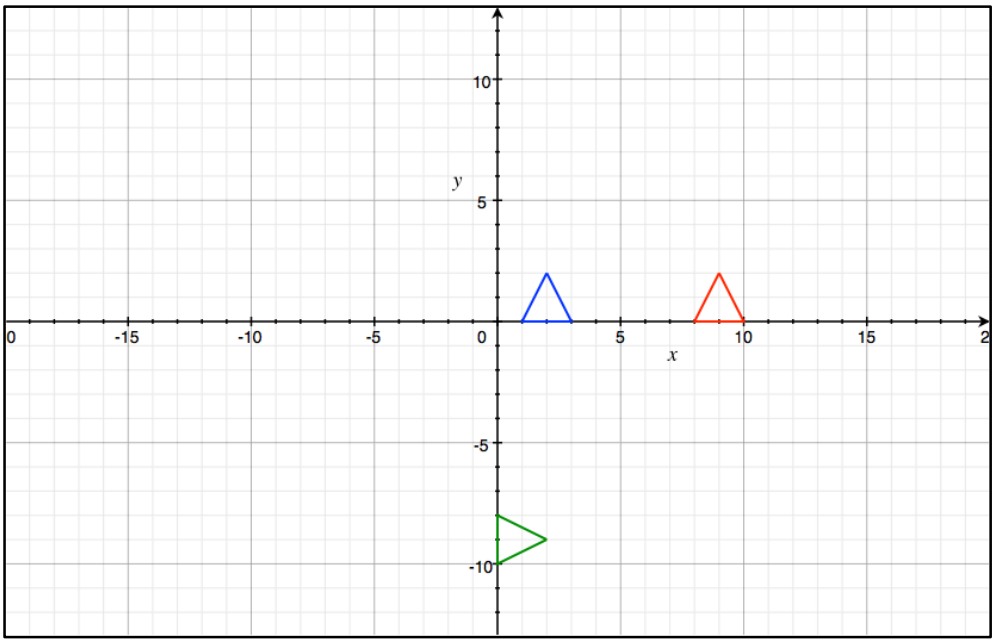
The equations are generated by calculating the slope and using one point to determine the equation through the formula y-y1 = m(x-x1).

The length is calculated using the formula |AB| = $\sqrt{(x1 - x2)^2 - (y1 - y2)^2}$.

The area is calculated using the formula Area = | ½ (x1*y2 - x2*y1) + (x2*y3 - x3*y2) + (x3*y1 - x1*y3)|.

The type of triangle is determined by firstly comparing the length of the lines that make it up. If two are equal, it's an isosceles; if all are equal, it's an equilateral triangle (special case of isosceles); if all are different, it's a scalene. After that it checks if the triangle is right-angled by testing its sides through Pythagoras' Theorem.

13) To move Charlie 7 steps in a positive horizontal direction, all x coordinates should be increased by 7, while keeping the same values for y. To, then, rotate it 90° clockwise about the origin, all points (x, y) should become (y, -x). In the below image, the original Charlie is blue, the translated Charlie is red and the rotated Charlie is green. The new coordinates are: A = (2, -9); B = (0, -8); C = (0, -10)



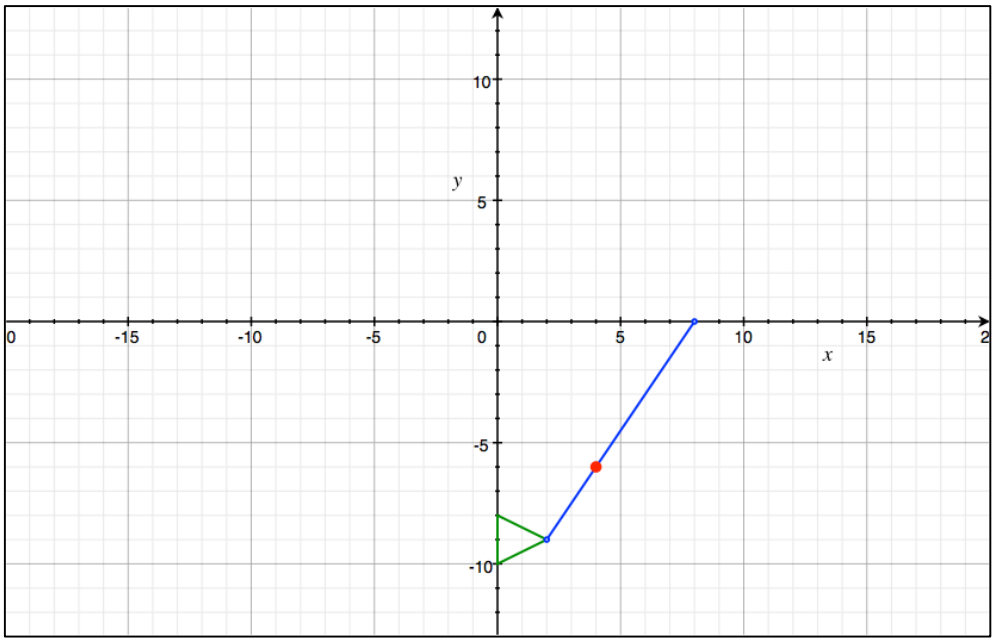
14) Considering the points:
A = (2, -9) and H = (8, 0):
|AH| = $\sqrt{(x1-x2)^2-(y1-y2)^2}$
|AH| = $\sqrt{117} \cong 10.82$.

The point one third of this distance from A, is calculated by

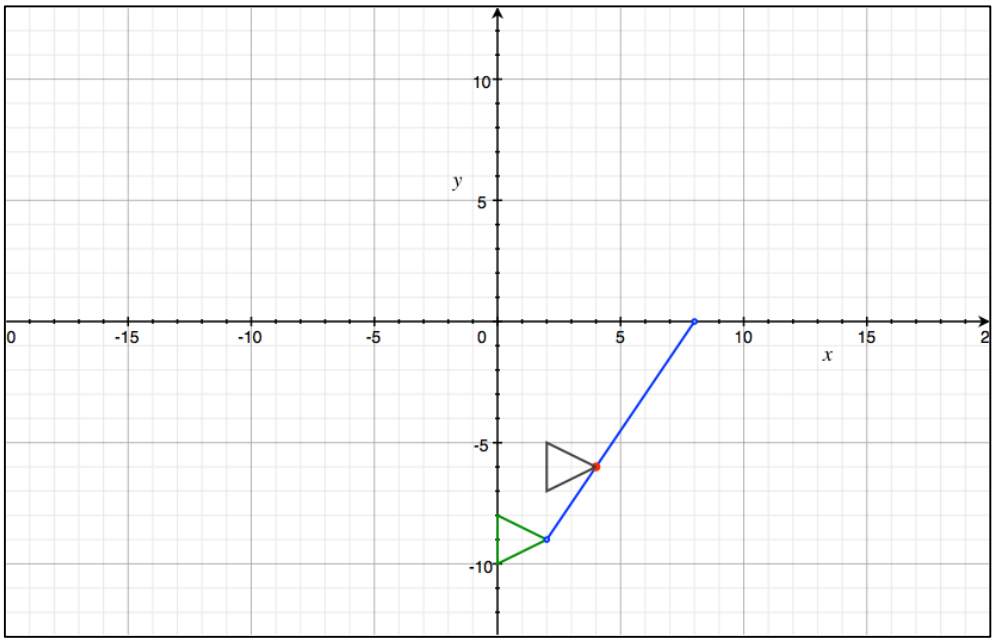
$$x = x1 + \frac{x2-x1}{3} = 4$$

$$y = y1 + \frac{y2-y1}{3} = -6$$

Point = (4, -6)

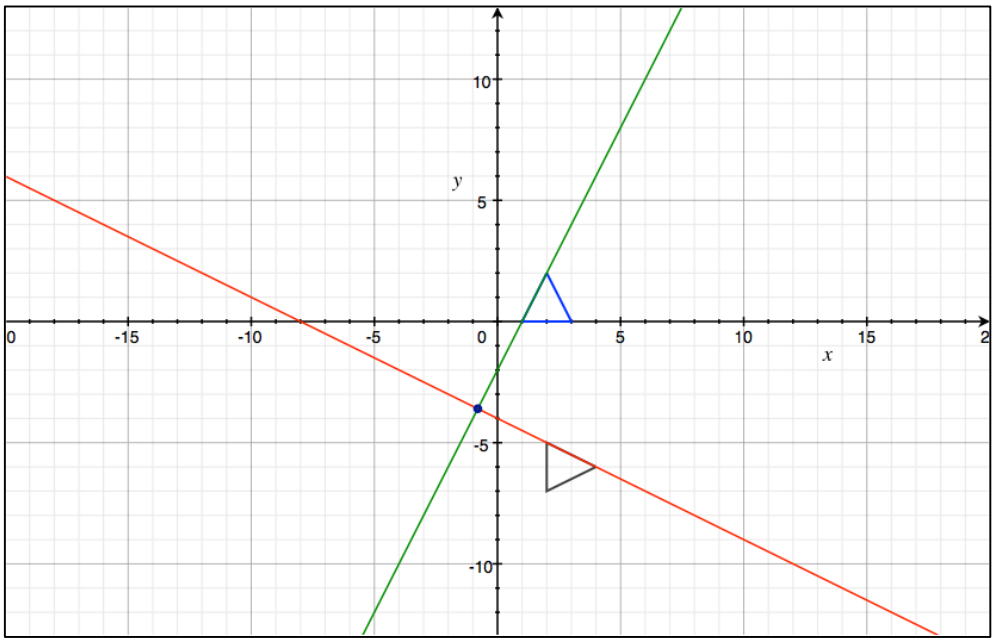


15) Translation needed: $T_{2,3}$.
A = (4, -6).
B = (2, -5).
C = (2, -7).



16) Current line = c / Current A = (4, -6) / Current B = (2, -5).
Original line = o / Original A = (2, 2) / Original B = (1, 0).

$Slope_c = \Delta y/\Delta x = (-6 - (-5))/(4-2) = -½$.
Equation C: y-y1=m(x-x1)



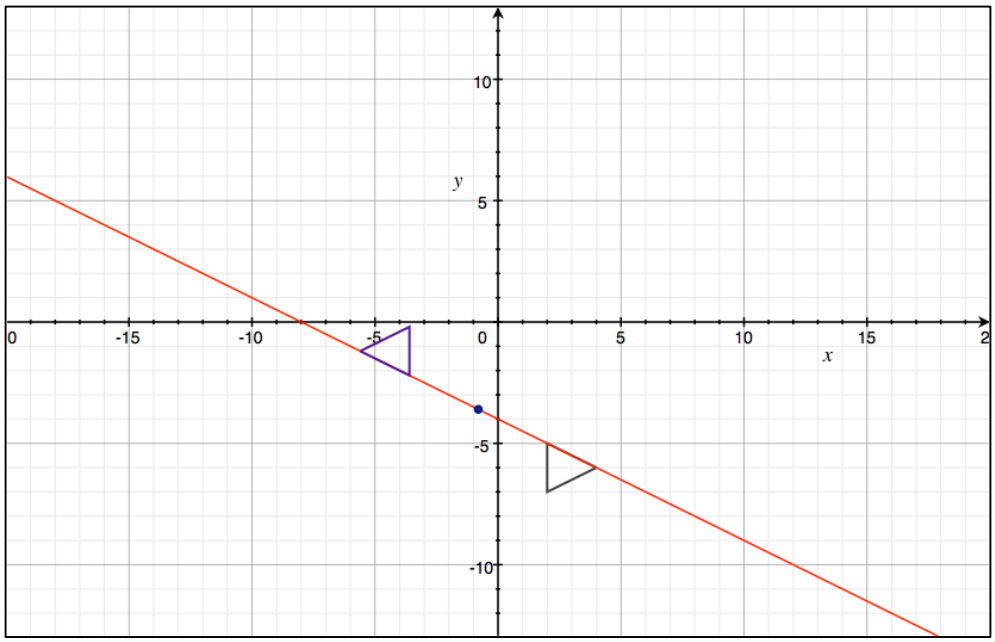
y-(-6) = -½(x-4).
y + 6 = -x/2 + 2
y = -x/2 – 4

$Slope_o = \Delta y/\Delta x = (2-0)/(2-1) = 2$.
Equation O: y – 2 = 2(x – 2)
y – 2 = 2x – 4
y = 2x – 2

POI: 2x – 2 = -x/2 – 4
2x + x/2 = -4 + 2
2.5x = -2
x = -0.8
y = 2 X 0.8 – 2 = -1.6 – 2 = -3.6

POI = (-0.8, -3.6)

17) Central symmetry about a point $C_{a, b}$ = (x, y) → (-x+2a, -y+2b).
Central symmetry about $POI_{-0.8, -3.6}$ = (-x+2 X -0.8, -y + 2 X -3.6)
A = (4, -6) → (-4-1.6, 6-7.2) = (-5.6, -1.2)
B = (2, -5) → (-2-1.6, 5-7.2) = (-3.6, -2.2)
C = (2, -7) → (-2-1.6, 7-7.2) = (-3.6, -0.2)



18)

19)

20)

```
Centroid of Charlie= (239.33333, 145.0).

Charlie's vertices after being rotated 45° clockwise:
A = (194.55, 123.32).
B = (315.47, 78.77).
C = (207.98, 232.92).

Charlie's vertices after being shrunk by 30%:
A = (156.10, 68.60).
B = (238.00, 106.40).
C = (108.50, 129.50).

Area of the triangle: 3393.49 square units.
```

The coordinates of the centroid were calculated using the formula:
Cx = (x1+x2+x3)/3
Cy = (y1+y2+y3)/3
Centroid = (Cx, Cy)

To rotate 45° clockwise, vertex A was translated to the origin, all points were translated using the same translation. Then, the points were rotated using:
x'= x cos θ + y sin θ
y'= -x sin θ + cos θ.

The last step was to translate each point back to its position using the inverse translation used before.

Finally, to shrink Charlie, each coordinate was shrunk by the same chosen rate and the area recalculated using the same method used on question 11.