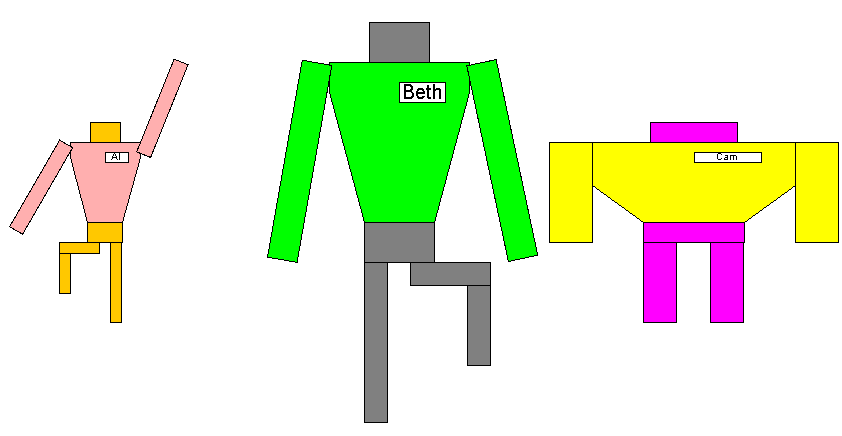
MovingRobot Lab

In this lab, you will be creating a program to draw a robot. The robot can move its arms and legs. Here are some sample pictures of robots:



Exercise 1 – Instance Variables (In class)

Your first task is to identify the different attributes that each robot has. What makes each robot different from its peers? Hint: there are nine attributes. For each attribute, what type of variable do you need? When you have a complete list of the attributes and types, add instance variables to the MovingRobot class.

Exercise 2 – Constructor (In class)

Five of the nine attributes for the robot need to be specified at the time that the robot is created. Four of the attributes can be set to default values. Identify which attributes are in which groups. Hint: Cam (above) is still in her starting position… which of the attributes can be set to defaults and what are the default values for these attributes?

Write a constructor for class MovingRobot:

1. What is the name of the constructor function?
2. Make the constructor public. It should take five values as input to initialize the five attributes.
3. The body of the function should initialize the five specified attributes to the supplied values.
4. The body of the function should also initialize the four other attributes to their default values.

Exercise 3 – Getters (In class)

Provide a getter for each of the nine attributes. The name of each getter should be getXXX where XXX is the name of the attribute, unless the attribute is a boolean value in which case use isXXXYYY where XXX is the attribute name and YYY is an adjective that describes that attribute when the attribute value is true. How many arguments should each getter take as input?

Exercise 4 – Additional Getters (Start in class)

Once we know the over-all dimensions of the robot, we can compute several other quantities. For example, the height of the torso should be 40% of the height of the robot. Write some additional getters for the following attributes. How many arguments should each of these getters take as input?

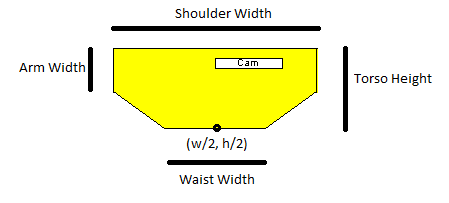
1. Torso Height – 40% of the total height
2. Leg Height – 40% of the total height
3. Pelvis Height – half the height less the leg height (hint: use the leg height function to help out here)
4. Head Height – half the height less the torso height
5. Shoulder Width – 70% of the total width
6. Waist Width – half the shoulder width
7. Arm Width – half the remaining width after the shoulder width is removed
8. Arm Length – 125% of the torso length
9. Leg Width – one third the width of the waist
10. Head Width – 30% of the total width
11. Name Tag Width – one third the width of the shoulders
12. Name Tag Height – one eighth of the torso height

Exercise 5 – Drawing the Robot

Class MovingRobotView has several functions to paint various parts of the robot.

5a – paintTorso

This function should paint the torso of the robot. The torso is a hexagon, so you’ll need to use the draw/fill Polygon functions in the Graphics class. These functions consume two arrays of integers and a number. The number indicates the number of sides of the polygon. The arrays indicate the x coordinates and the y coordinates of the vertices of the polygon. You’ll need to create the two arrays and then set each entry to the coordinates of the vertices of the hexagon. The center of the window (w/2, h/2) should be at the middle of the bottom edge of the hexagon (ie the waist of the robot). All other vertices can be calculated from the getters that you defined in exercises 3 and 4 above.



To test, uncomment the lines in launchSingleRobot and then run the launchSingleRobot function in the main function. You will see only the torso.

5b – paintPelvis and paintHead

The pelvis and the head are a rectangles. The robot should supply the dimensions and you can figure out the upper left corner. To test, run launchSingleRobot again. You should see the torso, pelvis and head.

5c – paintNameTag

The name tag is a rectangle with the robot’s name in it. The tag should be 1 tag-height down from the shoulders and the left side of the tag should be in the center of the torso. The dimensions can be retrieved from the robot. Don’t forget that class GraphicsUtilityFunctions provides two functions which can be used to get a Font object and draw text centered on a rectangle. To test, run launchSingleRobot again to see the name tag.

5d – paintLegs (leg down)

Each of the legs can be in one of two positions (up or down). If the leg is down, paint a vertical rectangle. Use functions from the robot to get the dimensions and location of each leg. Later, we’ll worry about the leg being up.

5e – paintArms

I’ve provided a helper function that paints a rectangle at a given angle: paintRotatedRectangle. The inputs are as follows:

g - A Graphics object

c - A Color

ULCx, ULCy - The coordinates of the upper left hand corner of the rectangle before it is rotated

width, height – the dimensions of the rectangle

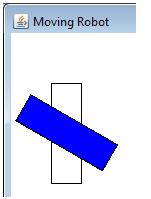
cOffX, cOffY – the location of the point of rotation (relative to the upper left hand corner)

angle – the angle in radians to rotate the rectangle.

For example, to rotate a blue rectangle with width 30 and height 100, with ULC at (40, 50) 30 degrees clockwise around its center, you would use

paintRotatedRectangle( g, Color.BLUE, 40, 50, 30, 100, 15, 50, 30\*Math.PI/180 );

Here’s a picture of the original and rotated rectangles. Note that the cOffX and cOffY give the point of rotation RELATIVE to the ULC rather than in absolute screen coordinates. That is, the point of rotation is actually (40 + 15, 50 + 50), not (15, 50).



The length, width and angle of each arm can be obtained from the robot. The rotation point for each arm is arm width/2 pixels down and right from the ULC of the arm. For right now, the arms should be at the robot’s sides.

Exercise 6 – Moving the Robot

The robot can do eight things: raise/lower each arm/leg. In class MovingRobot, provide the following functions:

6a – Arms

raiseRightArm – raise the right arm by 1 degree. Make sure that the angle does not exceed 90 degrees.

raiseLeftArm – same as raiseRight arm but for the left arm.

lowerRightArm – lower the right arm by 1 degree. Make sure that the angle is at least 0 degrees.

lowerLeftArm – same as lowerRight arm but for the left arm.

To test these changes, we actually need to call these functions. Class MovingRobotView has a function called handleKeyPress. This function consumes a MovingRobot, and int (which you will ignore), and a KeyEvent. The KeyEvent has information about what key is pressed. Specifically, it has a function called getKeyCode. Each key on your keyboard has a number associated with it. These numbers have constants within the KeyEvent class for convenience. For example, the ‘A’ key on your keyboard has code KeyEvent.VK\_A (VK stands for “Virtual Key”). In the handleKeyPress function, if the key pressed is I (ie the key code is KeyEvent.VK\_I), raise the right arm of the robot. If the key pressed is k, lower it. Use the W and S keys to raise/lower respectively the left arm. Make these changes and then test your program using the launchSingleRobot function.

6b – Legs (up)

Add four functions to the robot class to raise or lower the legs. Note that the legs are either up or down… there is not in-between here. Also, if one leg is already up, then raising the other leg should do nothing (we don’t want our robots to fall down!).

Next, change paintLegs in the MovingRobotView class so that, if a leg is up, it paints two rectangles of length Leg Length / 2 as shown in the picture at the top of this document. Finally, change handleKeyPress so that if the user presses U or J, the robot raises or lowers his right leg. If the user presses Q or A the robot should raise or lower his left leg.