MAT 2170: Graphic Coordinates Exercise

Introduction

When writing graphics programs, we need to determine the placement of objects in the graphics window. When we develop Java programs which utilized the `acmLibrary` graphics tools, we want to plan ahead and think about a uniform method for placing graphic objects, since it can be very frustrating to use "trial and error" to do this. What we're interested in is using *scaling* and *translation* to position graphic objects correctly. These operations require us to do multiplication and addition by hand, but after we cover Chapter 3, you will be able to solve these problems in a much better manner.

Preparations

For each exercise, use one of the planning grids provided in this handout to sketch the desired graphic. For each sketch, provide a full "inventory" of the graphical components and their attributes. For example, if a circle appears, we need to know the following details:

- Where is the "bounding box" for this circle? (x =?, y =?, $width$ =?)
- What color is to be used? (What is the predefined Java color name?)
- Should the circle be filled? (True or false?)

Example Exercise

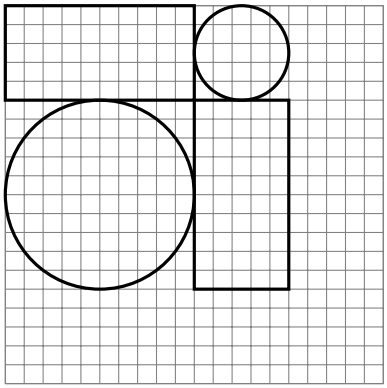
As an example, consider the graphic shown at the top of the next page. To keep things simple, we will choose convenient units for the sketch, then scale and translate as necessary to obtain the actual coordinates we desire. For example, the location and size of each of the circles and rectangles below has been summarized as a table of x, y, w, and h values. Each square of the grid is considered to be one unit, representing 20 pixels.

We have used the same conventions of the ACM library to describe the location and size of rectangles and circles: a rectangle is described by the coordinate (x, y) of its upper-left corner, its width w and height h. Circles are defined by a square bounding box.

We wish to use a scaling factor of 20 so the larger circle has a diameter of 10 units $\times$ 20 or 200 pixels. Observe how the table used for the planning grid has a direct impact on the values used in the Java program: multiplying each number by 20 provides the necessary scaling.

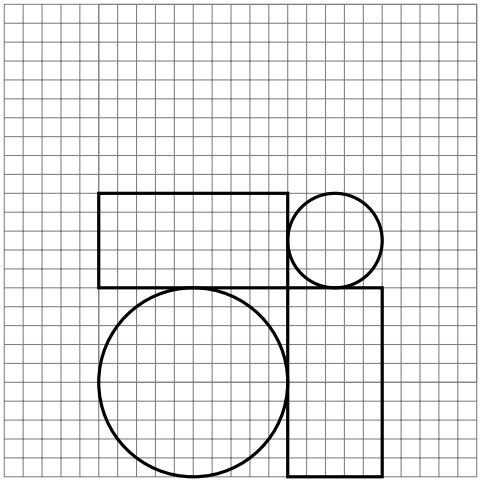
It is important to realize that, with careful planning, no "trial-and-error" is needed to obtain the desired graphic image. In this example, the scaled image will be flush to the upper left-hand corner of the output window. A horizontal and/or vertical translation can be easily obtained by adding appropriate Δx and Δy values:

In–class Practice:



Position & Dimensions	x	y	w	h
upper-left rectangle	0	0	10	5
upper-right circle	10	0	5	5
lower-left circle	0	5	10	10
lower-right rectangle	10	5	5	10

object	x	y	w	h	filled?	color name
Upper Left GRect						
Upper Right GOval						
Lower Left GOval			200	200		
Lower Right GRect						



$\Delta x = 5$

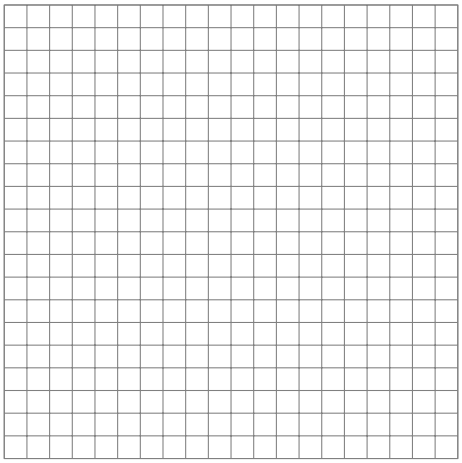
$\Delta y = 10$

Original Values	x	y	w	h
upper-left rectangle	0	0	10	5
upper-right circle	10	0	5	5
lower-left circle	0	5	10	10
lower-right rectangle	10	5	5	10

object	x	y	w	h	filled?	color name
Upper Left GRect						
Lower Left GOval						
Upper Right GOval						
Lower Right GRect						

For use with Lab #2:

DrawHouse - flush to top left corner

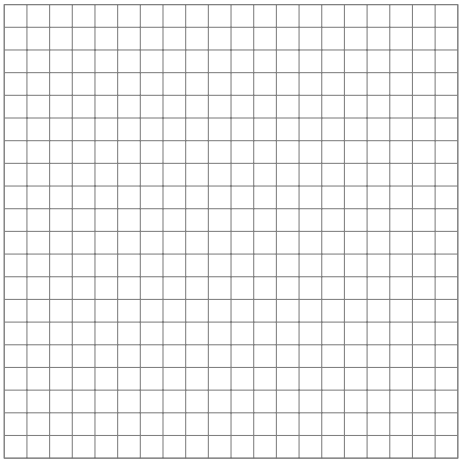


	x	y	w	h
Base				
Roof - left				
Roof - right				
Left window				
Right window				
Door				
Doorknob				

DrawHouse - moved 4 units right and 5 units down

$\Delta x =$

$\Delta y =$

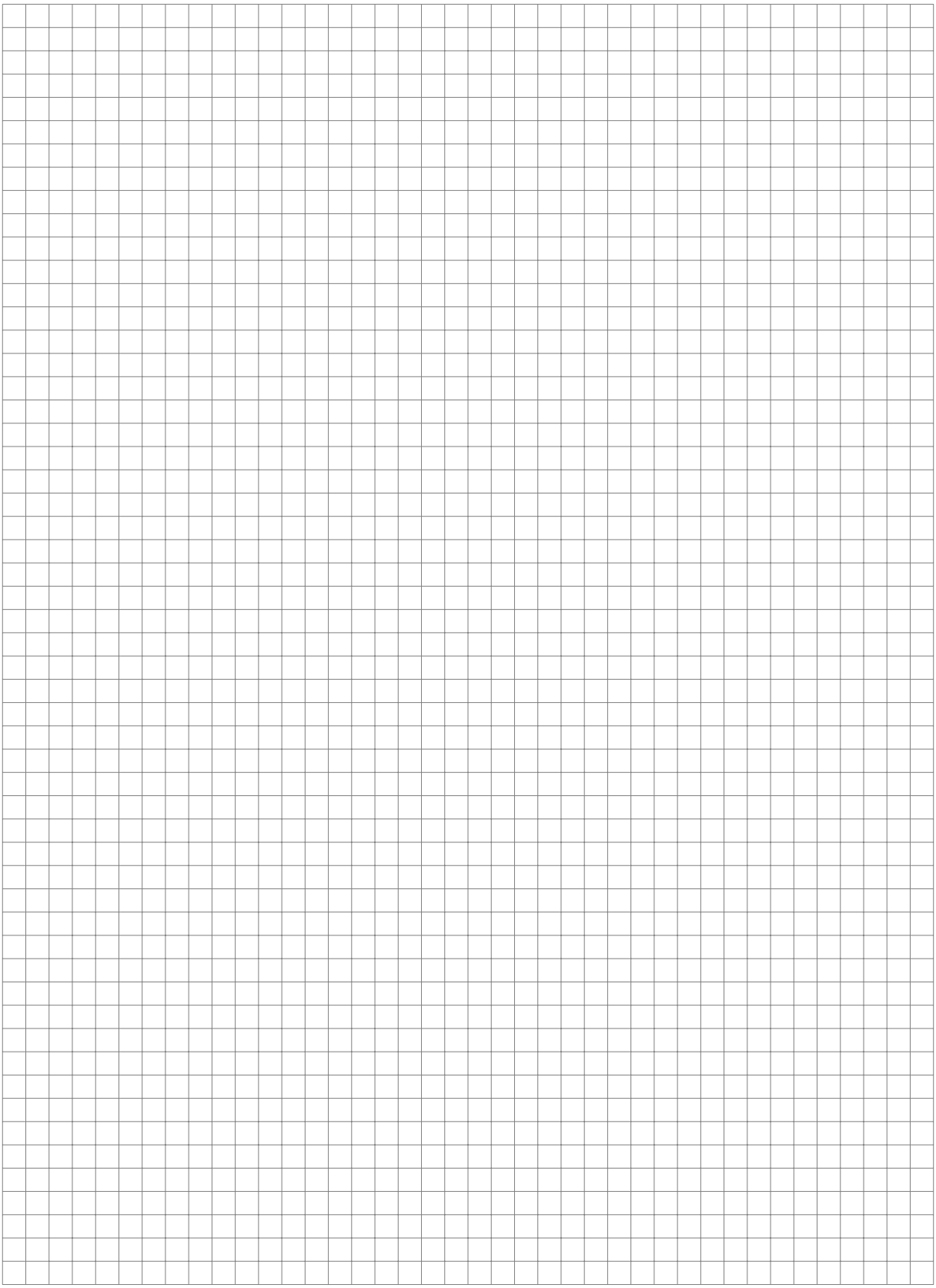


	x	y	w	h
Base				
Roof-left				
Roof-right				
Left window				
Right window				
Door				
Doorknob				

Scaled by 20	x	y	w	h	filled?	color name
Base GRect						
Roof-left GLine						
Roof-right GLine						
Left-window GRect						
Right-window GRect						
Door GRect						
Doorknob GOval						

Group Project: Robot / Alien / Bug

Robot Sketch — Flush to Origin $\Delta x =$ _____ $\Delta y =$ _____



Scaled by 20	x	y	w	h	filled?	color name
Head						
AntenaeLeft						
EyeLeft						
Mouth						
Neck						
Body						