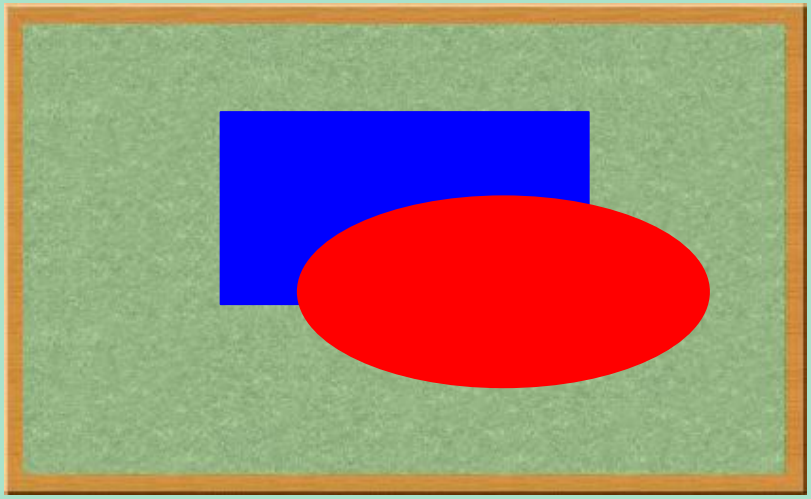
Interactive Graphics

Abraham Gebrekidan
EIP 01
August 09, 2024

Review: The Collage Model

- SJS uses the same graphics model that we have used for the last decade, which is based on the metaphor of a ***collage***.
- A collage is similar to a child's felt board that serves as a backdrop for colored shapes that stick to the felt surface. As an example, the following diagram illustrates the process of adding a blue rectangle and a red oval to a felt board:



- Note that newer objects can obscure those added earlier. This layering arrangement is called the ***stacking order***.

The **GWindow** Class Revisited

The following expanded set of methods are available in the **GWindow** class:

add(*object***)**	Adds the object to the canvas at the front of the stack
add(*object***, *x*, *y*)**	Moves the object to (*x*, *y*) and then adds it to the canvas
remove(*object***)**	Removes the object from the canvas
removeAll()	Removes all objects from the canvas
getElementAt(*x***, *y*)**	Returns the frontmost object at (*x*, *y*), or **null** if none
getWidth()	Returns the width in pixels of the entire canvas
getHeight()	Returns the height in pixels of the entire canvas
setBackground(*c***)**	Sets the background color of the canvas to *c*.

The Two Forms of the **`add`** Method

- The **`add`** method comes in two forms. The first is simply

```
add(object);
```

  which adds the object at the location currently stored in its internal structure. You use this form when you have already set the coordinates of the object, which usually happens at the time you create it.

- The second form is

```
add(object, x, y);
```

  which first moves the object to the point (*x*, *y*) and then adds it there. This form is useful when you need to determine some property of the object before you know where to put it.

Methods Common to All `GObjects`

`setLocation(`*x*`, `***y*`)`**	Resets the location of the object to the specified point
`move(`*dx*`, `***dy*`)`**	Moves the object *dx* and *dy* pixels from its current position
`movePolar(`*r*`, `***theta*`)`**	Moves the object *r* pixel units in direction *theta*
`getX()`	Returns the *x* coordinate of the object
`getY()`	Returns the *y* coordinate of the object
`getWidth()`	Returns the horizontal width of the object in pixels
`getHeight()`	Returns the vertical height of the object in pixels
`contains(`*x*`, `***y*`)`**	Returns **`true`** if the object contains the specified point
`setColor(`*c*`)`**	Sets the color of the object to the color *c*
`getColor()`	Returns the color currently assigned to the object
`scale(`*sf*`)`**	Scales the shape by the scale factor *sf*
`rotate(`*theta*`)`**	Rotates the shape counterclockwise by *theta* degrees
`sendToFront()`	Sends the object to the front of the stacking order
`sendToBack()`	Sends the object to the back of the stacking order
`sendForward()`	Sends the object forward one position in the stacking order
`sendBackward()`	Sends the object backward one position in the stacking order

Additional Methods for `GOval` and `GRect`

Fillable shapes (`GOval` and `GRect` [and later `GArc` and `GPolygon`])

`setFilled(`*flag***`)`**	Sets the fill state for the object (`false`=outlined, `true`=filled)
`isFilled()`	Returns the fill state for the object
`setFillColor(`*c***`)`**	Sets the color used to fill the interior of the object to *c*
`getFillColor()`	Returns the fill color

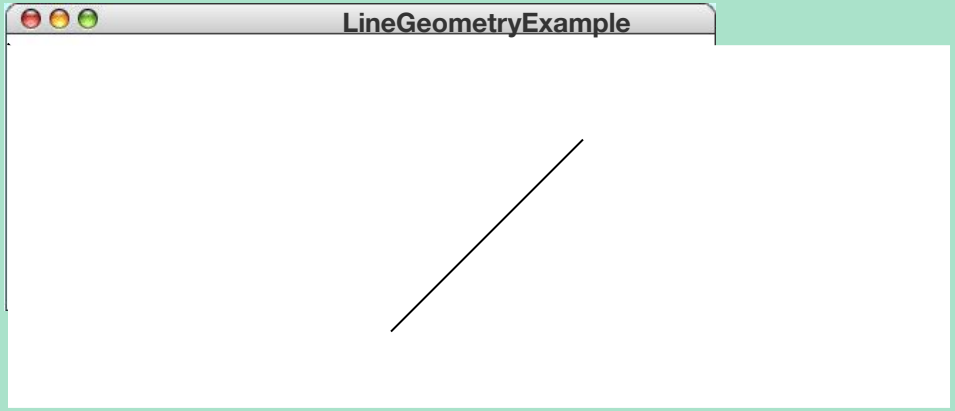
Resizable shapes (`GOval` and `GRect` [and later `GImage`])

`setSize(`*width*`,` *height***`)`**	Sets the dimensions of the object as specified
`setBounds(`*x*`,` *y*`,` *width*`,` *height***`)`**	Sets the location and dimensions together

Additional Methods for `GLine`

`setStartPoint(`*x*`,` *y*`)`	Sets the start point without changing the end point
`setEndPoint(`*x*`,` *y*`)`	Sets the end point without changing the start point

```
function LineGeometryExample() {
   var gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);
   var line = GLine(0, 0, 100, 100);
   gw.add(line);
   line.setLocation(200, 50);
   line.setStartPoint(200, 150);
   line.setEndPoint(300, 50);
}
```



The End