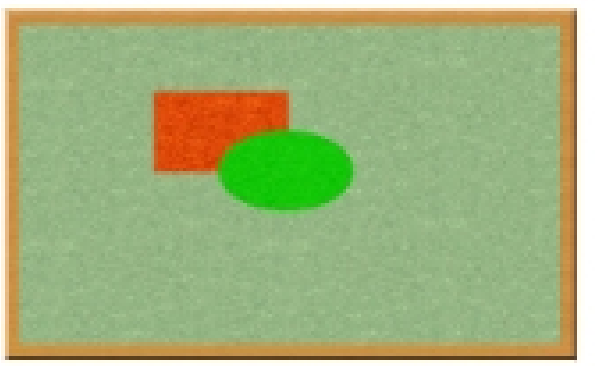
Interactive Graphics

Eric Roberts
CS 106A
January 27, 2010

The `acm.graphics` Model

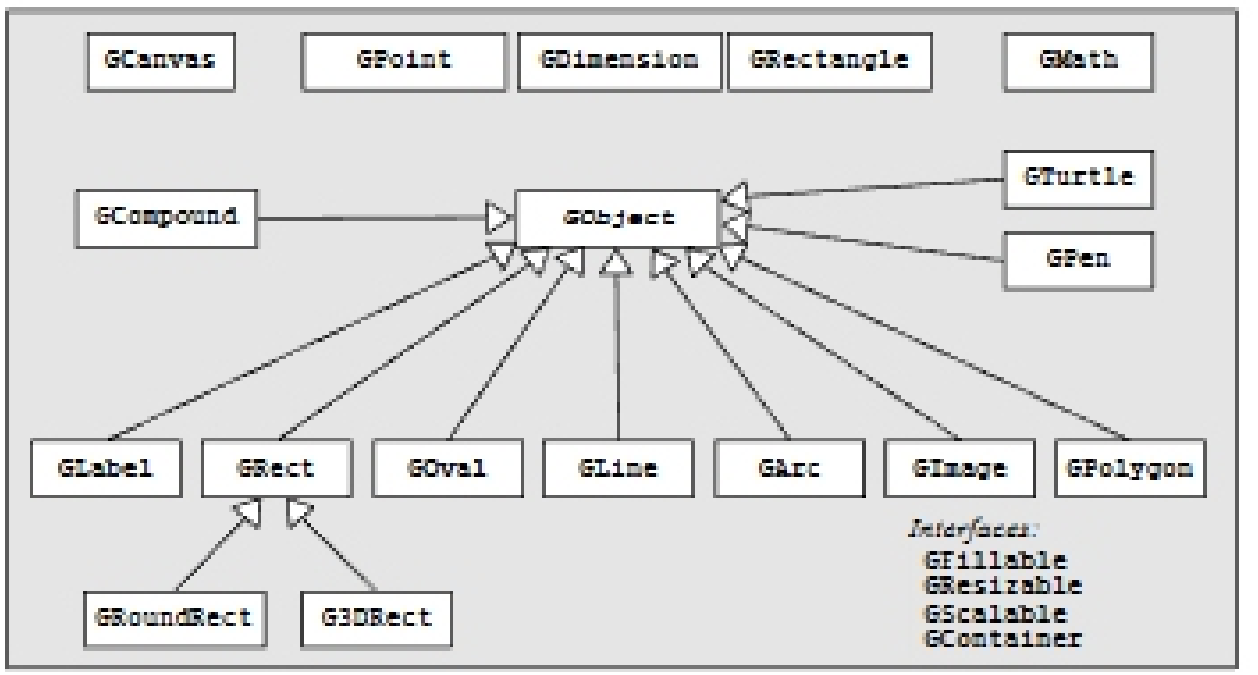
- The `acm.graphics` package uses a ***collage*** model in which you create an image by adding various objects to a canvas.
- A collage is similar to a child's felt board that serves as a backdrop for colored shapes that stick to the felt surface. As an example, the following diagram illustrates the process of adding a red rectangle and a green oval to a felt board:



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

Structure of the `acm.graphics` Package

The following diagram shows the classes in the `acm.graphics` package and their relationship in the Java class hierarchy:



The `GCanvas` Class

- The `GCanvas` class is used to represent the background canvas for the collage model and therefore serves as a virtual felt board. When you use the `acm.graphics` package, you create pictures by adding various `GObjects` to a `GCanvas`.
- For simple applications, you won't actually need to work with an explicit `GCanvas` object. Whenever you run a program that extends `GraphicsProgram`, the initialization process for the program automatically creates a `GCanvas` and resizes it so that it fills the program window.
- Most of the methods defined for the `GCanvas` class are also available in a `GraphicsProgram`, thanks to an important strategy called *forwarding*. If, for example, you call the `add` method in a `GraphicsProgram`, the program passes that message along to the underlying `GCanvas` object by calling its `add` method with the same arguments.

Methods in the `GCanvas` Class

The following methods are available in both the `GCanvas` and `GraphicsProgram` classes:

`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 following methods are available in `GraphicsProgram` only:

`pause(`*milliseconds*`)`	Pauses the program for the specified time in milliseconds
`waitForClick()`	Suspends the program until the user clicks the mouse

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. If, for example, you want to center a `GLabel`, you must first create it and then use its size to determine its location.