Objects and Graphics

One Last Thought on Loops...

Looping Forever

- **`while`** loops iterate as long as their condition evaluates to **`true`**.
- A loop of the form **`while (true)`** will loop forever (unless something stops it).

```
while (true) {
    ...

}
```

- You can immediately exit a loop by using the **`break`** statement.

Looping Forever

- **`while`** loops iterate as long as their condition evaluates to **`true`**.
- A loop of the form **`while (true)`** will loop forever (unless something stops it).

```
while (true) {
    …
    if ( … ) break;
}
```

- You can immediately exit a loop by using the **`break`** statement.

The “Loop-and-a-Half” Idiom

- Often you will need to
  - read a value from the user,
  - decide whether to continue, and if so
  - process the value.
- Technique: The **loop-and-a-half idiom**:

```
while (true) {
    /* … get a value from the user … */
    if (condition)
        break;

    /* … process the value … */
}
```

Time-Out For Announcements!

Programming Assignment 2

- Second programming assignment is currently out and is due on Friday, January 31.
- Suggestion: Have `PythagoreanTheorem`, `HailstoneSequence`, and `FindRange` done by Friday.
- Assignment review hours this **Sunday, January 26** from 5PM – 6PM in Hewlett 200.
  - Will not be recorded, but we'll post notes online.

Keith's Office Hours

- New OH location: Gates 300.
- Same times as before:
  - Tuesday, 10:15AM – 12:15PM
  - Wednesday, 4:30PM – 6:30PM
- Everyone is welcome!

A Friendly Reminder: Honor Code

Your Emails

- As of last night, I've read 532 of your emails.
- I'll try to compile a list of some of the really wonderful stories from those emails for Friday.
- If you'd prefer that I not include your interesting story in the slides, please feel free to email me. I respect your privacy!

Casual CS Dinner

- Casual dinner for women studying computer science is **tonight** on the Gates fifth floor, starting at 6PM.
- I'll be shortening my normal office hours in Gates 300 tonight to 4:30PM – 6:00PM.
- Everyone is welcome!

Back to CS106A!

Object-Oriented Programming

An **object** is an entity
that has state and behavior.

Has a fur color.
Has an energy level.
Has a level of cuteness.
Can be your friend.
Can sit.
Can stay.
Can bark.

An **object** is an entity
that has state and behavior.

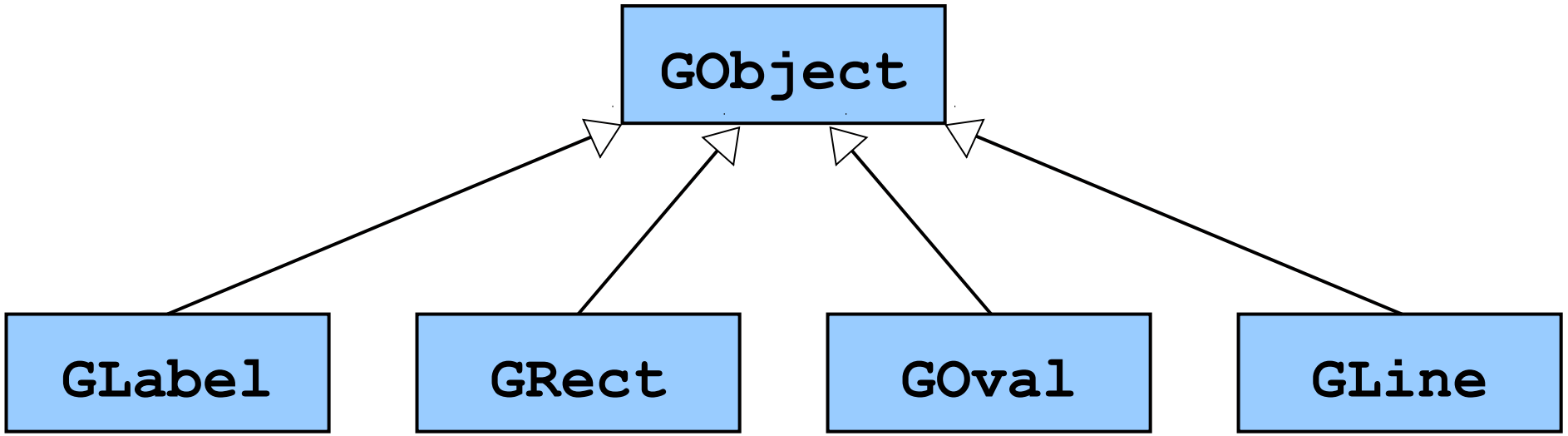
A **class** is a set of features and behavior
common to a group of objects.

An **instance of a class** is an
object that belongs to that class.

Programming with Graphics

The `GObject` Hierarchy

The classes that represent graphical objects form a hierarchy, part of which looks like this:



Graphic courtesy of Eric Roberts

Sending Messages to a **GLabel**

```
public class HelloProgram extends GraphicsProgram {
    public void run() {
        GLabel label = new GLabel("hello, world", 100, 75);
        label.setFont("SansSerif-36");
        label.setColor(Color.RED);
        add(label);
    }
}
```

label

hello, world

Graphic courtesy of Eric Roberts

Objects and Variables

- Variables can be declared to hold objects.
- The type of the variable is the name of the class:
  - `GLabel label;`
  - `GOval oval;`
- Instances of a class can be created using the **new** keyword:
  - `GLabel label = new GLabel("Y?", 0, 0);`

Sending Messages

- To call a method on an object stored in a variable, use the syntax

***object*.*method*(*parameters*)**

- For example:

```
label.setFont("Comic Sans-32");
label.setColor(Color.ORANGE);
```

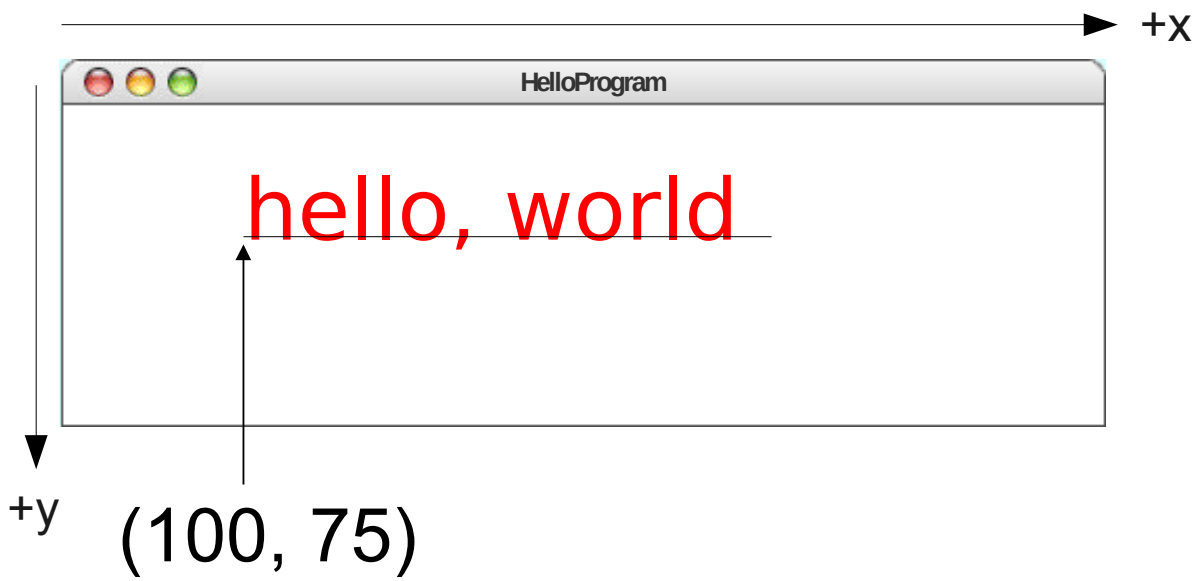
Graphics Coordinates

- Origin is upper left.
- *x* coordinates increase from left to right.
- *y* coordinates increase from top to bottom.
- Units are **pixels** (dots on the screen).
- **`GLabel`** coordinates are baseline of first character.

Graphic courtesy of Eric Roberts

Graphics Coordinates

- Origin is upper left.
- *x* coordinates increase from left to right.
- *y* coordinates increase from top to bottom.
- Units are **pixels** (dots on the screen).
- **`GLabel`** coordinates are baseline of first character.



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

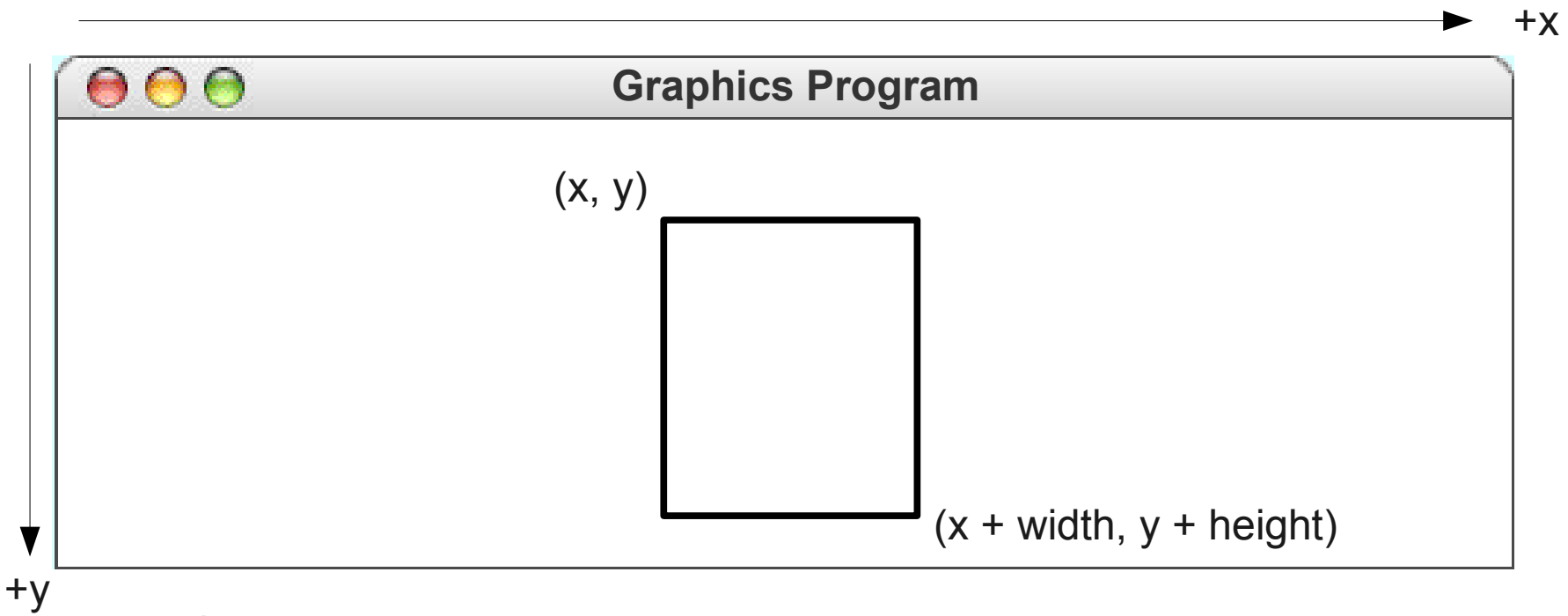
Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

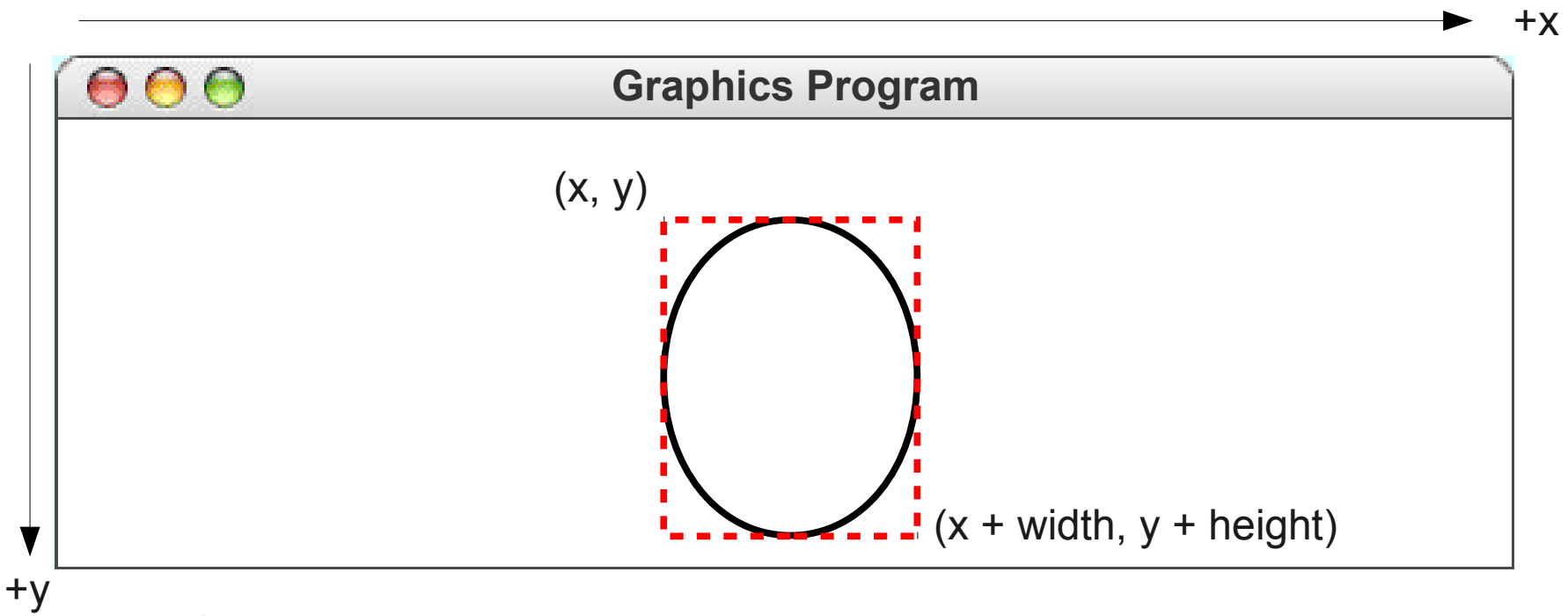
Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates an oval that fits inside the rectangle with the same dimensions.



Graphic courtesy of Eric Roberts

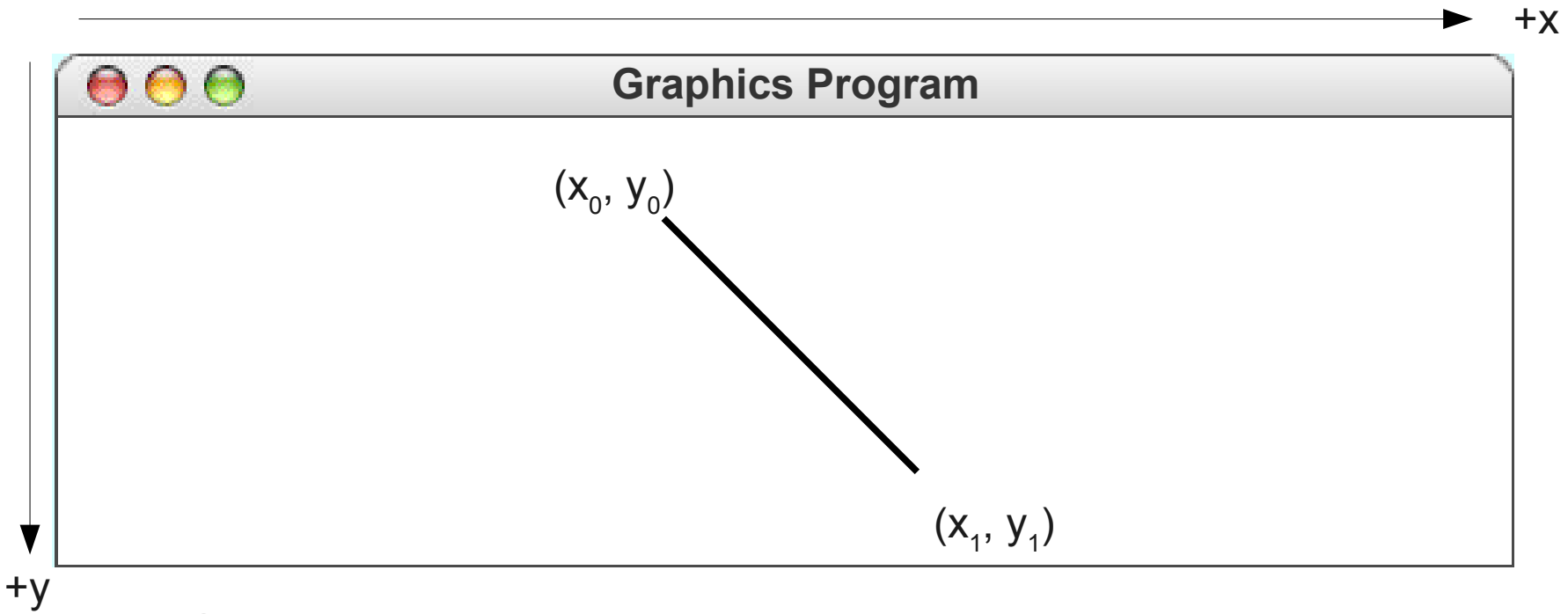
Drawing Geometrical Objects

Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`
Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*`,` *y*`,` *width*`,` *height*`)`
Creates an oval that fits inside the rectangle with the same dimensions.

`new GLine(` x_0`,` y_0`,` x_1`,` y_1`)`
Creates a line extending from (x_0, y_0) to (x_1, y_1).



Graphic courtesy of Eric Roberts

Drawing Geometrical Objects

Constructors

`new GRect(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates a rectangle whose upper left corner is at (*x*, *y*) of the specified size

`new GOval(` *x*`,` *y*`,` *width*`,` *height*`)`

Creates an oval that fits inside the rectangle with the same dimensions.

`new GLine(` x_0`,` y_0`,` x_1`,` y_1`)`

Creates a line extending from (x_0, y_0) to (x_1, y_1).

Methods shared by the **`GRect`** and **`GOval`** classes

object`.`**`setFilled(`***fill*`)`

If *fill* is **`true`**, fills in the interior of the object; if **`false`**, shows only the outline.

object`.`**`setFillColor(`***color*`)`

Sets the color used to fill the interior, which can be different from the border.

Graphic courtesy of Eric Roberts

The Collage Model

The Collage Model

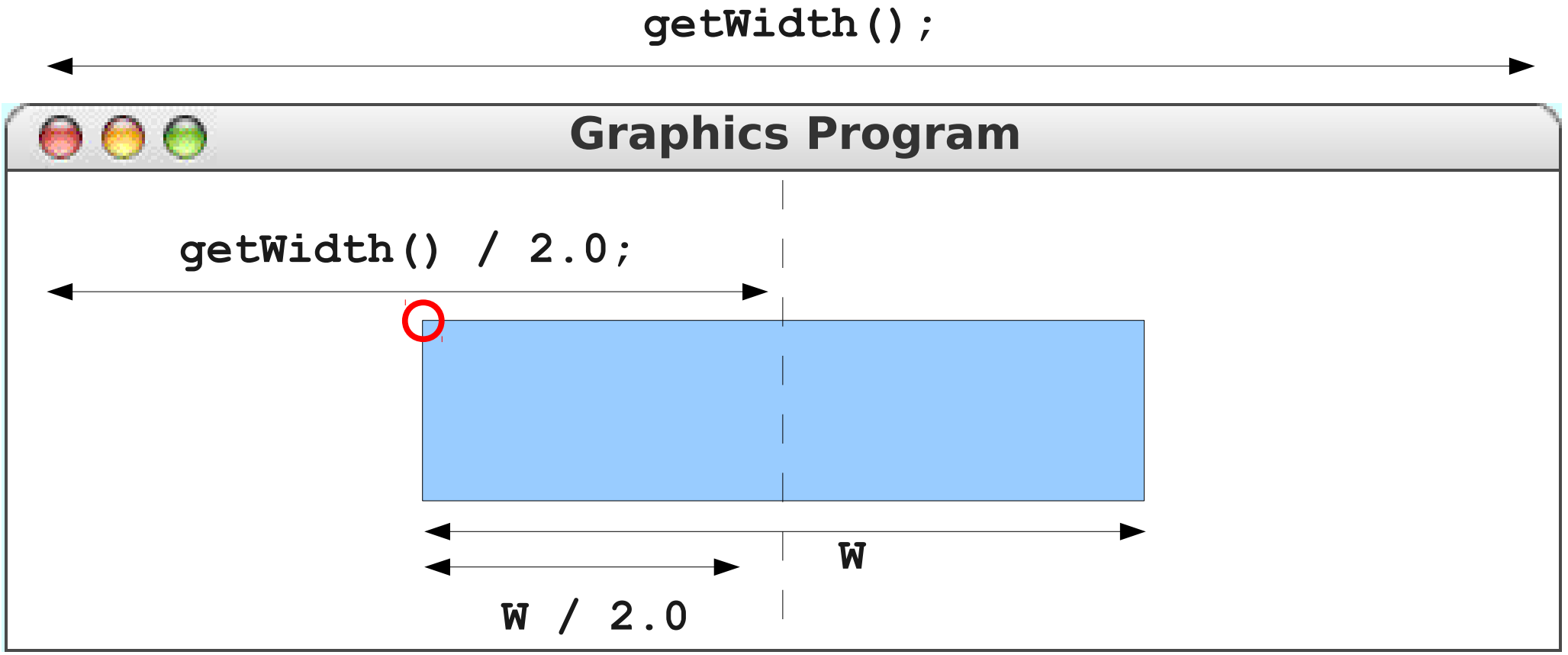
Size of the Graphics Window

Methods provided by **`GraphicsProgram`** class

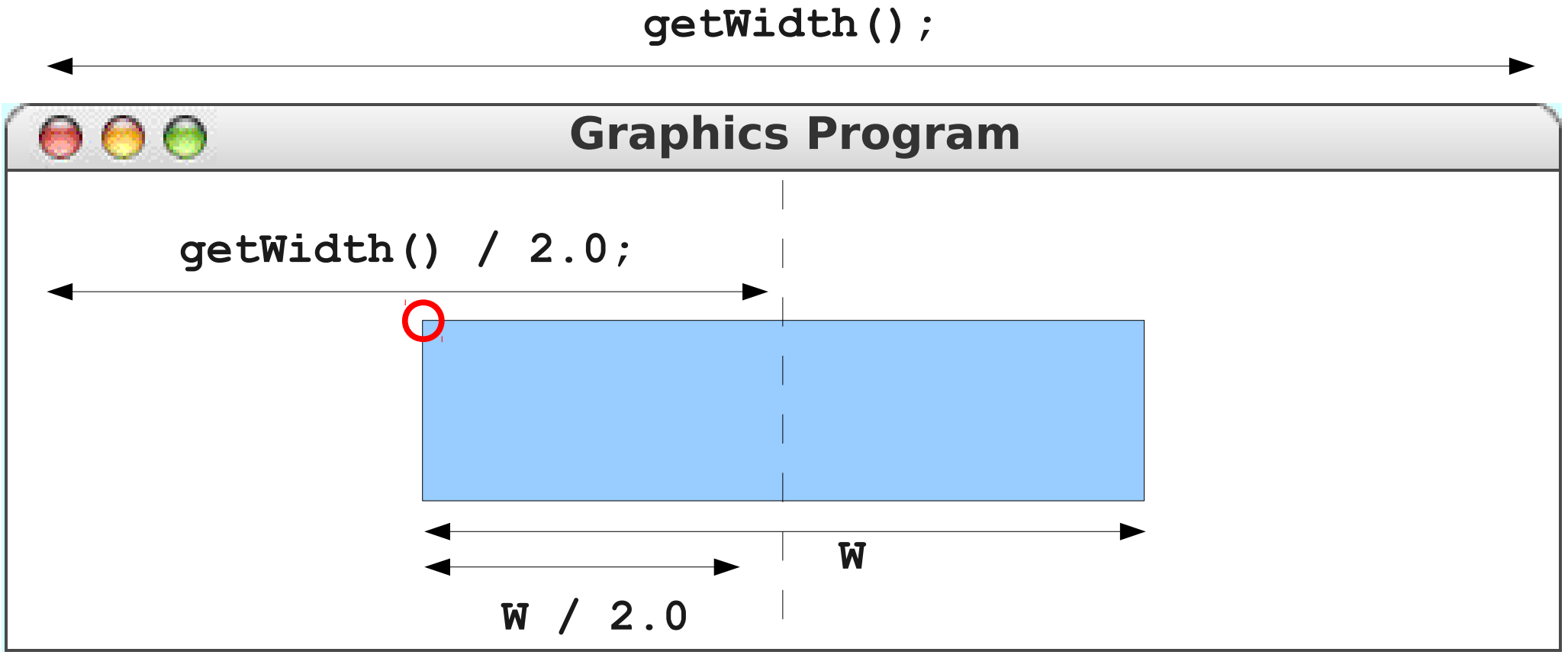
`getWidth()` Returns the width of the graphics window.
`getHeight()` Returns the height of the graphics window.

Note: receiver of these calls is the **`GraphicsProgram`** itself, so we don't need to specify a separate object as receiver.

Centering an Object



Centering an Object




```
x = (getWidth() / 2.0) - (W / 2.0);
x = (getWidth() - W) / 2.0;
```