Python Programming

Nesneler ve Grafikler

Graphical User Interface (GUI)

- Aşina olduğunuz çoğu uygulamada pencereler, simgeler, düğmeler, menüler ve çizimler sağlayan Grafik Kullanıcı Arayüzleri (GUI: Graphical User Interface) bulunur.
- GUI'lerin Xerox tarafından icadından önce, uygulamalar metin tabanlıydı ve bilgisayar terminalleri kullanılıyordu.
- Apple ve Microsoft, Macintosh ve PC'lerde GUI'leri entegre etti.
- Artık GUI'ler yaygın olarak mevcuttur.

Object-Oriented Languages
Nesne Yönelimli Diller

- Modern bilgisayar dilleri “Nesne Yönelimlidir”
- Dünyayı temsil ederken, dünyayı temel alarak modellemek daha kolaydır.
- Nesneler: (Fiziksel nesneler)
  - Öğretmen
  - Öğrenci
  - Elma
- Yöntemler: (Nesnelerle yapabilecekleriniz)
  - öğretmen.soru(soru)
  - öğrenci.çalışma(kitap)
  - elma.yemek()
- Nesne yönelimli bilgisayar dillerine örnek olarak şunlar verilebilir: Java, C#, C++, Python
- Nesne yönelimli olmayan diller: C, Basic, Fortran, Pascal.

graphics.py

- There’s a graphics library (graphics.py) written specifically to go with the textbook. It’s based on Tkinter.
- You can download it from:

  http://mcsp.wartburg.edu/zelle/python/graphics.py
- Save it in the same directory where your graphics programs are located.
- Alternatively you can put it in Python’s Lib directory with other libraries

Simple Graphics Programming

- Since this is a library, we need to import the graphics commands
  `import graphics`
- A *graphics window* is a place on the screen where the graphics will appear.
  `win = graphics.GraphWin()`
- This command creates a new window titled “Graphics Window.”

Simple Graphics Programming

- *GraphWin* is an object assigned to the variable *win*. We can manipulate the window object through this variable, similar to manipulating files through file variables.
- Windows can be closed/destroyed by issuing the command
  `win.close()`
- If you don’t close the window you have to kill the program.

Simple Graphics Programming

- It's tedious to use the `graphics.` notation to access the graphics library routines.
- `from graphics import *`
  The "from" statement allows you to load specific functions from a library module. "*" will load all the functions, or you can list specific ones.

Simple Graphics Programming

- Doing the import this way eliminates the need to preface graphics commands with `graphics.`

```
from graphics import *
win = GraphWin()
```

Simple Graphics Programming

- A graphics window is a collection of points called *pixels* (picture elements).
- The default GraphWin is 200 pixels tall by 200 pixels wide (40,000 pixels total).
- One way to get pictures into the window is one pixel at a time, which would be tedious. The graphics routine has a number of predefined routines to draw geometric shapes.

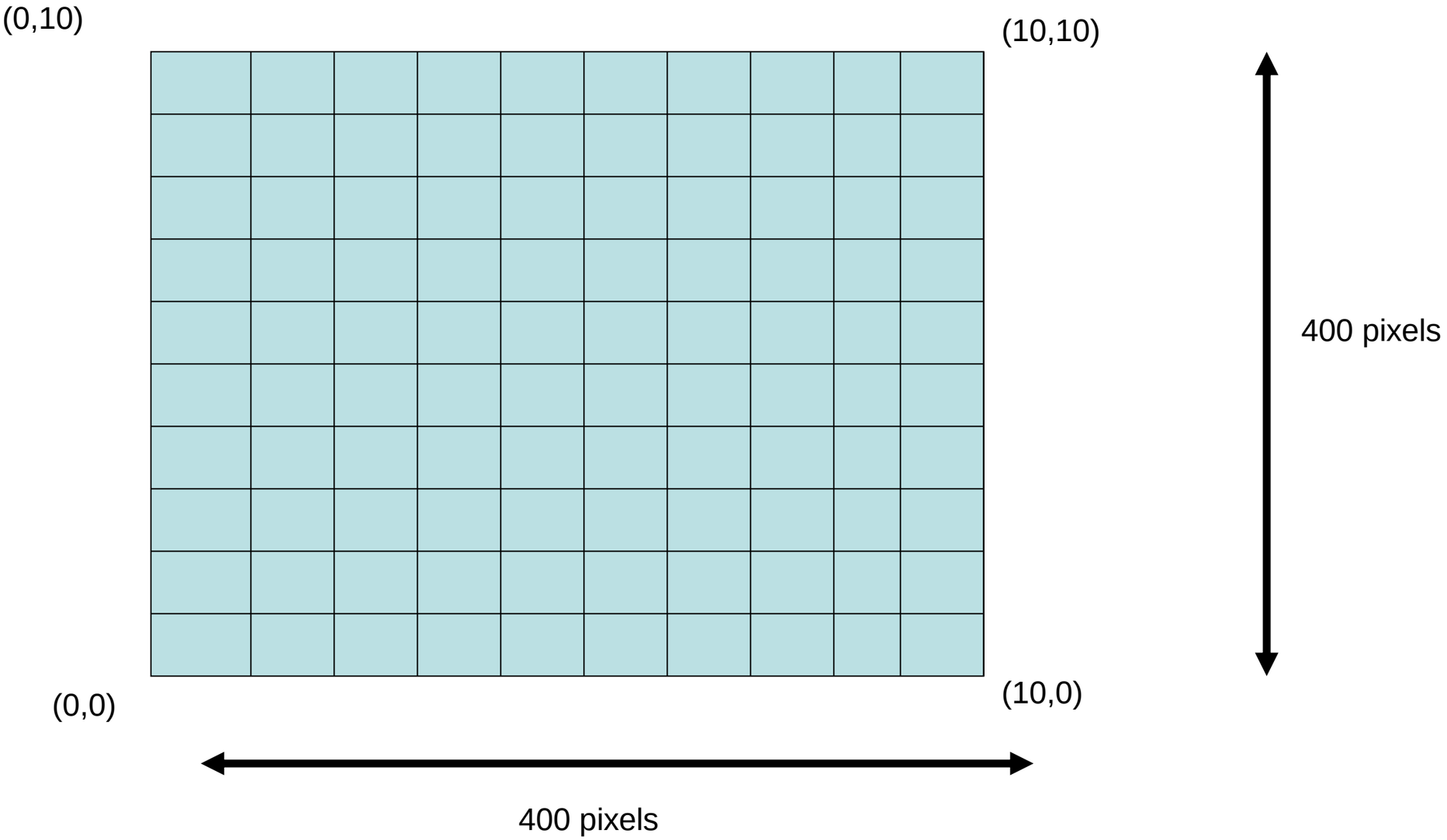
Scaling the Window

- To scale the window we do

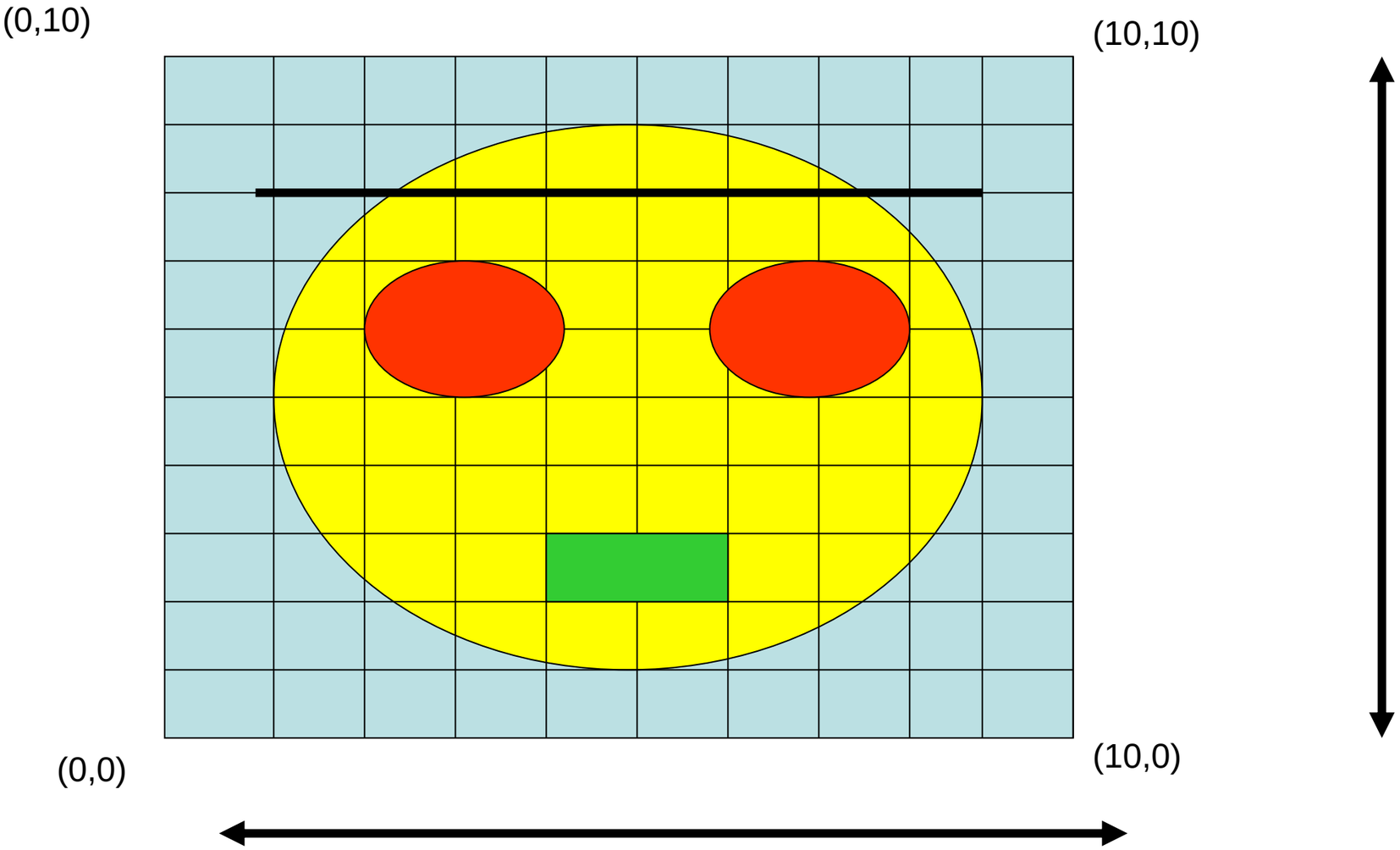
  win = GraphWin('Shapes', 400, 400)

  win.setCoords(0.0, 0.0, 10.0, 10.0)
- This will make the lower left corner to be (0,0) and the upper right corner to be (10,10)
- The size of the window is 400x400 pixels
- This will make drawing on the screen easier.

The screen



Simple Drawing



Simple Graphics Program

```
#
# graphics1.py - Simple graphics program.
#

from graphics import *

def Main():

    #Create a window 400x400 pixels
    win = GraphWin('Shapes', 400, 400)

    # Make the window scaled
    # bottom leftmost corner is (0,0)
    # top rightmost corner is (10,10)
    win.setCoords(0.0, 0.0, 10.0, 10.0)
```

Simple Graphics Program

```
#Draw a circle centered at 5,5
center = Point(5, 5)
circ = Circle(center, 4)
circ.setFill('yellow')
circ.draw(win)

# Draw left eye
eye1 = Circle(Point(3,6), 1)
eye1.setFill("red")
eye1.draw(win)

# Draw right eye
eye2 = Circle(Point(7,6), 1)
eye2.setFill("red")
eye2.draw(win)
```

Simple Graphics Program

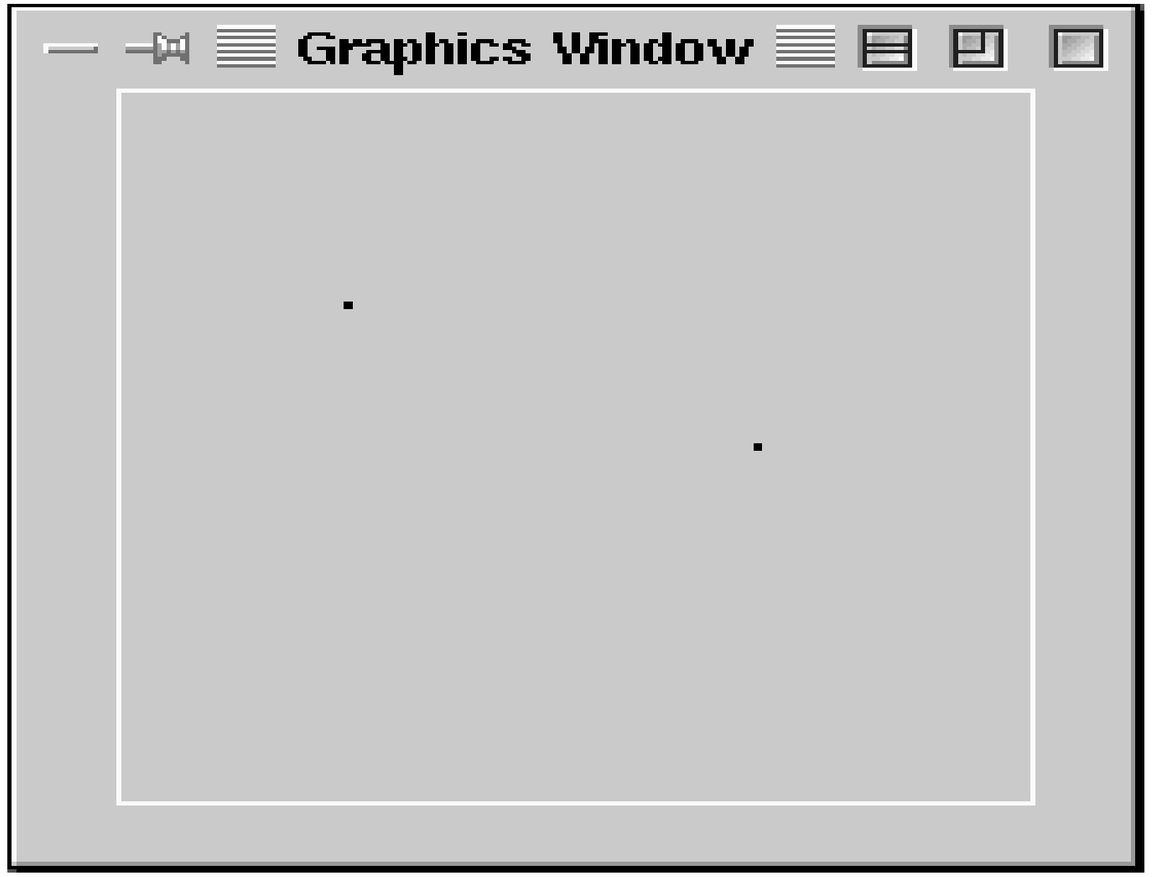
```
    # Draw mouth
    rect = Rectangle(Point(4, 2), Point(6, 3))
    rect.setFill("blue");
    rect.draw(win)

    # Draw line
    line = Line(Point(1, 8), Point(9, 8))
    line.draw(win)

    # Draw message
    message = Text(Point(5, 0.5), "Click anywhere to quit")
    message.draw(win)

    # Wait until we click mouse in the window
    win.getMouse()

    win.close()

Main()
```

Simple Graphics Programming

- The simplest object is the `Point`. Like points in geometry, point locations are represented with a coordinate system (*x*, *y*), where *x* is the horizontal location of the point and *y* is the vertical location.
- The origin (0,0) in a graphics window is the upper left corner.
- X values increase from right to left, y values from top to bottom.
- Lower right corner is (199, 199)

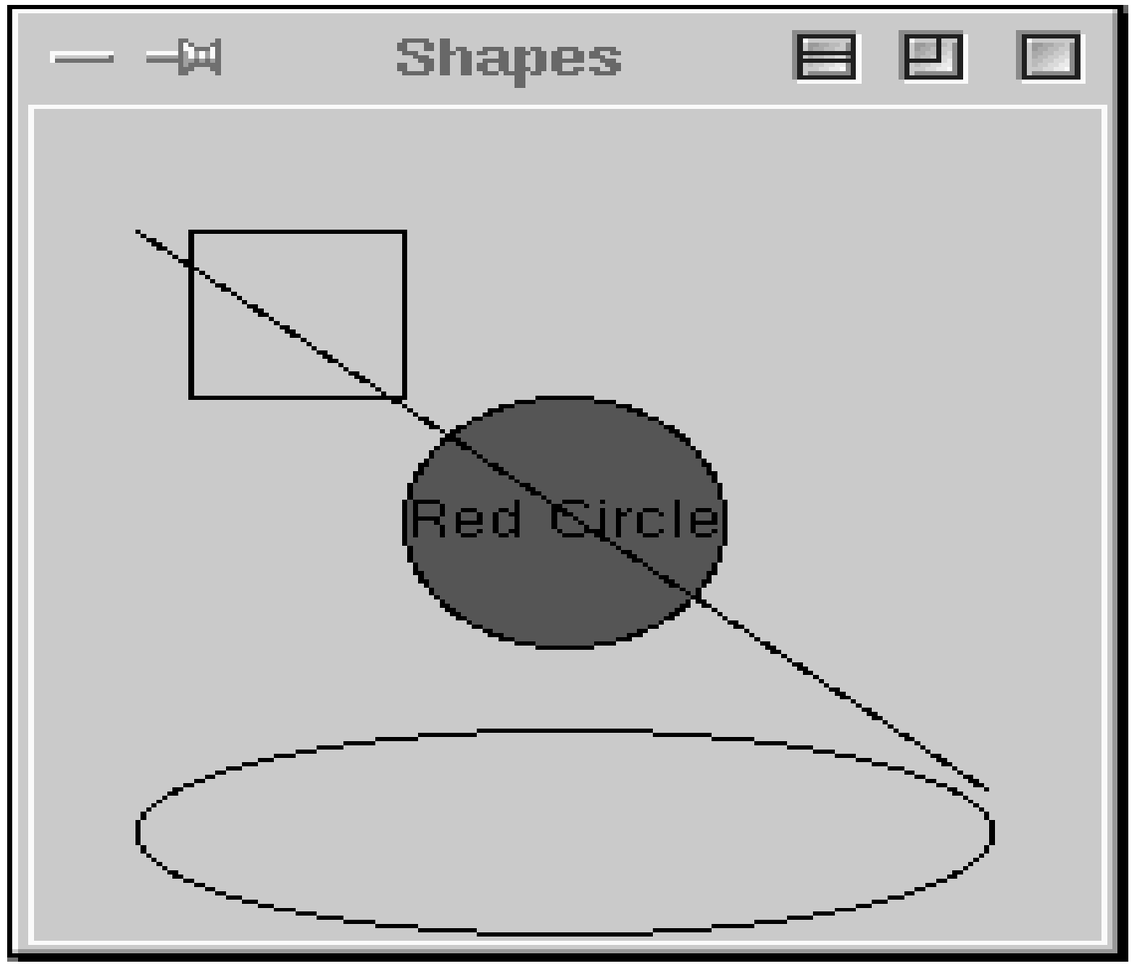
Simple Graphics Programming

```
from graphics import *
def Main():
  win = GraphWin('Shapes', 400, 400)
  p = Point(50, 50)
    p.draw(win)

    # draw the other point
    p = Point(350, 350)
    p.draw(win)

    # Wait for a clink on the window
    win.getMouse()

    # Close window
    win.close()

Main()
```



Simple Graphics Programming

```
>>> ### Open a graphics window
>>> win = GraphWin('Shapes')
>>> ### Draw a red circle centered at point (100, 100)
     with radius 30
>>> center = Point(100, 100)
>>> circ = Circle(center, 30)
>>> circ.setFill('red')
>>> circ.draw(win)
>>> ### Put a textual label in the center of the circle
>>> label = Text(center, "Red Circle")
>>> label.draw(win)
>>> ### Draw a square using a Rectangle object
>>> rect = Rectangle(Point(30, 30), Point(70, 70))
>>> rect.draw(win)
>>> ### Draw a line segment using a Line object
>>> line = Line(Point(20, 30), Point(180, 165))
>>> line.draw(win)
>>> ### Draw an oval using the Oval object
>>> oval = Oval(Point(20, 150), Point(180, 199))
>>> oval.draw(win)
```



Using Graphical Objects

- Computation is preformed by asking an object to carry out one of its operations.
- In the previous example we manipulated GraphWin, Point, Circle, Oval, Line, Text and Rectangle. These are examples of *classes*.

Getting Mouse Clicks

- The following code reports the coordinates of a mouse click:

```
from graphics import *
win = GraphWin("Click Me!")
p = win.getMouse()
print("You clicked", p.getX(), p.getY())
```

- We can use the accessors like `getX` and `getY` or other methods on the point returned.

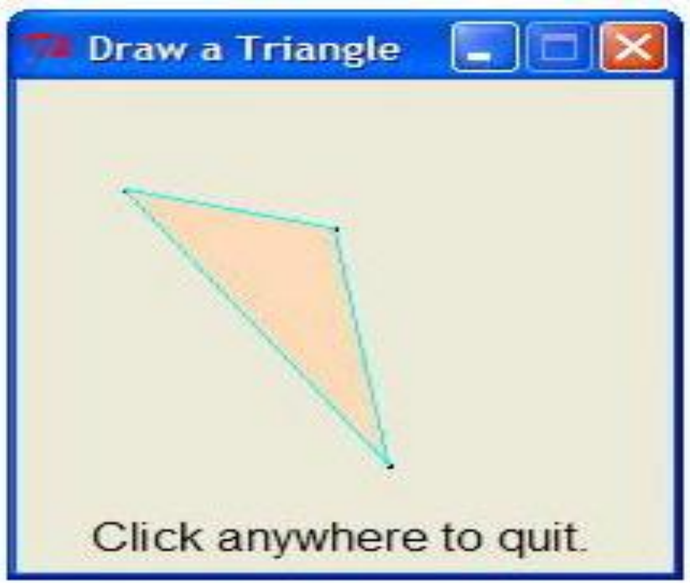
Getting Mouse Clicks

```
# triangle.pyw
# Interactive graphics program to draw a triangle

from graphics import *

def main():
    win = GraphWin("Draw a Triangle")
    win.setCoords(0.0, 0.0, 10.0, 10.0)
    message = Text(Point(5, 0.5), "Click on three points")
    message.draw(win)

    # Get and draw three vertices of triangle
    p1 = win.getMouse()
    p1.draw(win)
    p2 = win.getMouse()
    p2.draw(win)
    p3 = win.getMouse()
    p3.draw(win)
```

Getting Mouse Clicks

```
# Use Polygon object to draw the triangle
    triangle = Polygon(p1,p2,p3)
    triangle.setFill("peachpuff")
    triangle.setOutline("cyan")
    triangle.draw(win)

    # Wait for another click to exit
    message.setText("Click anywhere to quit.")
    win.getMouse()

main()
```

Getting Mouse Clicks



Getting Mouse Clicks

- Notes:
  - If you are programming in a windows environment, using the .pyw extension on your file will cause the Python shell window to not display when you double-click the program icon.
  - There is no triangle class. Rather, we use the general polygon class, which takes any number of points and connects them into a closed shape.

Getting Mouse Clicks

- Once you have three points, creating a triangle polygon is easy:

```
triangle = Polygon(p1, p2, p3)
```

- A single text object is created and drawn near the beginning of the program.

```
message = Text(Point(5,0.5), "Click on three points")
message.draw(win)
```

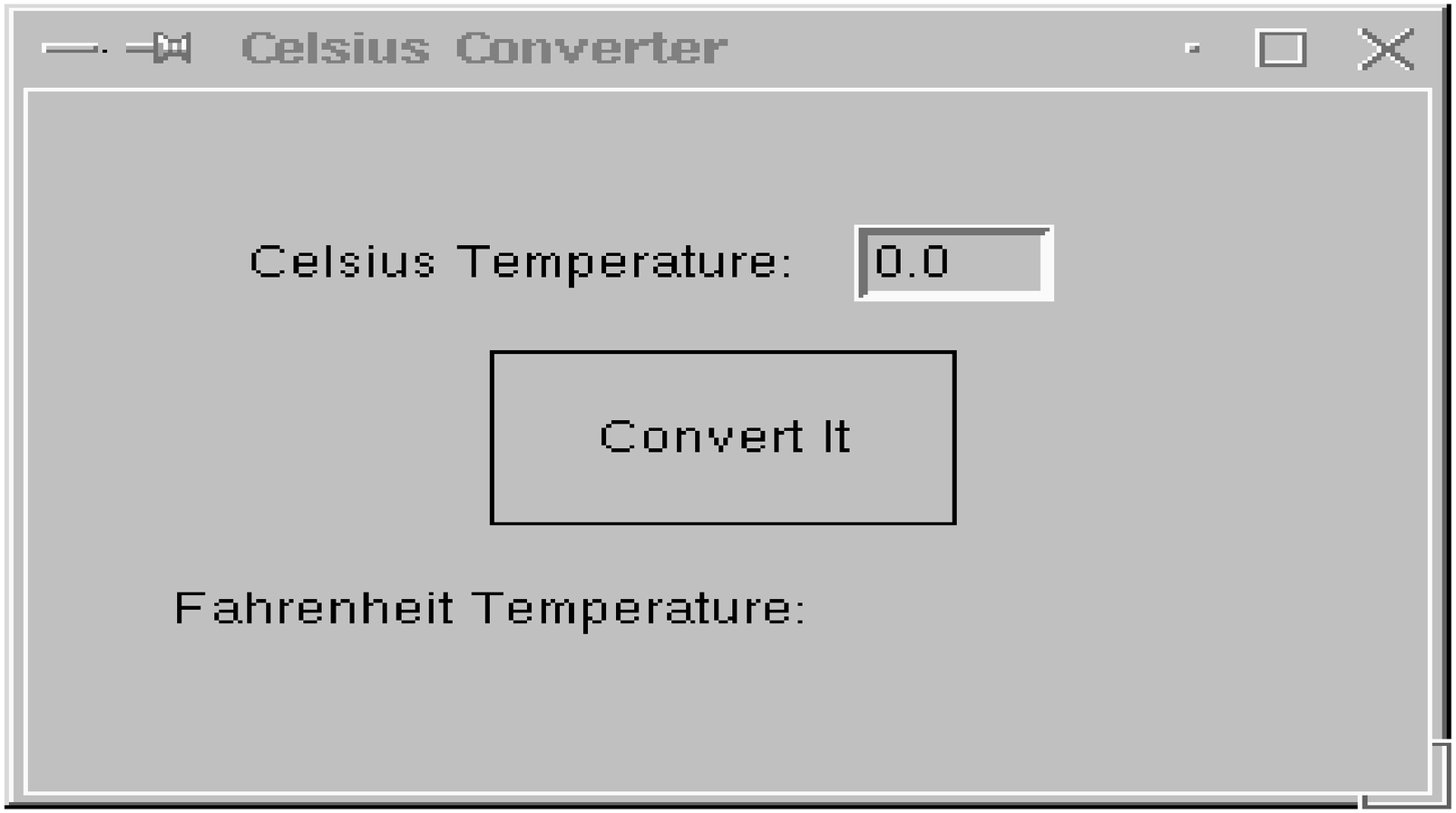
- To change the prompt, just change the text to be displayed.

```
message.setText("Click anywhere to quit.")
```

Handling Textual Input

- The triangle program's input was done completely through mouse clicks. There's also an `Entry` object that can get keyboard input.
- The `Entry` object draws a box on the screen that can contain text. It understands `setText` and `getText`, with one difference that the input can be edited.

Handling Textual Input



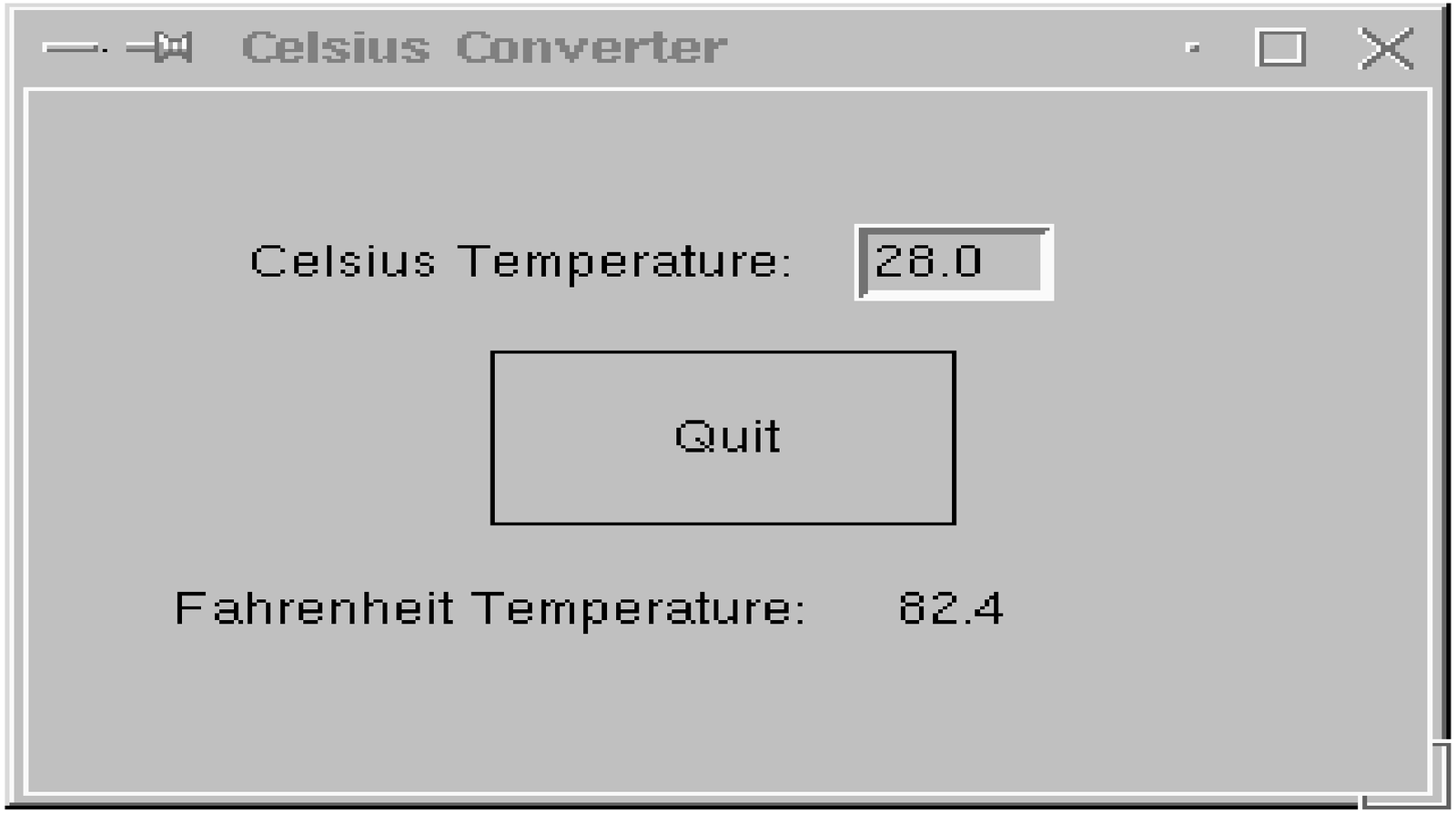
Handling Textual Input

```
# convert_gui.pyw
# Program to convert Celsius to Fahrenheit using a simple
#   graphical interface.

from graphics import *

def main():
    win = GraphWin("Celsius Converter", 300, 200)
    win.setCoords(0.0, 0.0, 3.0, 4.0)

    # Draw the interface
    Text(Point(1,3), "   Celsius Temperature:").draw(win)
    Text(Point(1,1), "Fahrenheit Temperature:").draw(win)
    input = Entry(Point(2,3), 5)
    input.setText("0.0")
    input.draw(win)
    output = Text(Point(2,1),"")
    output.draw(win)
    button = Text(Point(1.5,2.0),"Convert It")
    button.draw(win)
    Rectangle(Point(1,1.5), Point(2,2.5)).draw(win)
```

Handling Textual Input

```
    # wait for a mouse click
    win.getMouse()

    # convert input
    celsius = eval(input.getText())
    fahrenheit = 9.0/5.0 * celsius + 32

    # display output and change button
    output.setText(fahrenheit)
    button.setText("Quit")

    # wait for click and then quit
    win.getMouse()
    win.close()

main()
```

Handling Textual Input



Handling Textual Input

- When run, this program produces a window with an entry box for typing in the Celsius temperature and a button to “do” the conversion.
  - The button is for show only! We are just waiting for a mouse click anywhere in the window.

Handling Textual Input

- Initially, the input entry box is set to contain “0.0”.
- The user can delete this value and type in another value.
- The program pauses until the user clicks the mouse – we don’t care where so we don’t store the point!

Handling Textual Input

- The input is processed in three steps:
  - The value entered is converted into a number with `eval`.
  - This number is converted to degrees Fahrenheit.
  - This number is then converted to a string and formatted for display in the `output` text area.