

## Objectives

- Problem-solving with the Graphics' API

Oct 29, 2007

Sprenkle - CS111

1

## Object-Oriented Programming

- Objects **combine** data and methods together

Provides **interface** that users interact with

Hides internal data structures, implementation



Use an Application Programming Interface (**API**) to interact with a set of classes.

Oct 29, 2007

Sprenkle - CS111

2

## Using a Graphics Module/Library

- Allows us to handle graphical input and output
  - Example input: Mouse clicks
  - Example output: Pictures
- Not part of a standard Python distribution
- Made up of a collection of related **classes** of data and operations (**methods**) that manipulate the data

Oct 29, 2007

Sprenkle - CS111

3

## Using a Graphics Module/Library

- Handout lists the various classes
  - **Constructor** is in bold
    - Review: creates an object of that type
  - For each class, lists *some* of their methods and parameters
  - Drawn objects have some common methods
    - Listed at end of handout
  - Known as an **API**
- Today: Problem-solving using Graphics Module's API

Oct 29, 2007

Sprenkle - CS111

4

## Snippet of Code

- Using our knowledge of Python and the Graphics module's API, we knew what this program does

```
from graphics import *
def main():
    win = GraphWin("My Circle", 100, 100)
    c = Circle(Point(50,50), 10)
    c.draw(win)
    win.getMouse()
main()
```

GraphWin object → win = GraphWin("My Circle", 100, 100)

Also known as an **instance of the GraphWin class**

Constructor → GraphWin("My Circle", 100, 100)

Method called on GraphWin object → win.getMouse()

Oct 29, 2007

Sprenkle - CS111

5

## Using the Graphics Library

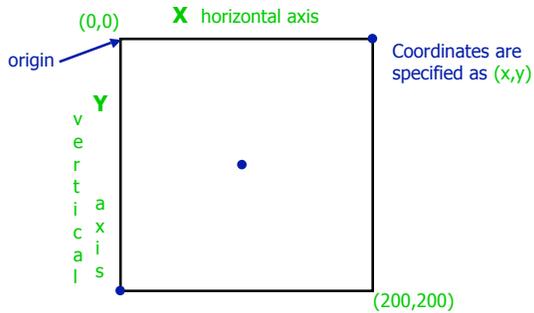
- In general, graphics are drawn on a canvas
  - A canvas is a 2-dimensional grid of pixels
- For our Graphics library, our canvas is a *window*
  - Specifically an **instance of the GraphWin class**
  - By default, a GraphWin object is 200x200 pixels

Oct 29, 2007

Sprenkle - CS111

6

## A GraphWin Object's Canvas



Oct 29, 2007

Sprenkle - CS111

7

## Snippet of Code

- What does this look like on the canvas?

```
from graphics import *  
  
def main():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse()  
  
main()
```

Oct 29, 2007

Sprenkle - CS111

8

## The GraphWin Class

- All parameters to the constructor are optional
- Could call constructor as
  - GraphWin()
    - Title, width, height to defaults ("Graphics Window", 200, 200)
  - GraphWin(<title>)
    - Width, height to defaults
  - GraphWin(<title>, <width>)
    - Height to default
  - GraphWin(<title>, <width>, <height>)

Oct 29, 2007

Sprenkle - CS111

9

## The GraphWin API

- **Accessor** methods for GraphWin
  - Return some information about the GraphWin
- Example methods:
  - <GraphWin>.getWidth()
  - <GraphWin>.getHeight()
    - (not on handout)

Oct 29, 2007

Sprenkle - CS111

10

## Problem: Draw a Full-Canvas Tic-Tac-Toe Board

- Function header: **def** drawTictactoe(canvas)
  - Given the GraphWin object *canvas*, draw a full-size tic-tac-toe board

Oct 29, 2007

Sprenkle - CS111 tictactoe.py

11

## The GraphWin API

- <GraphWin>.setBackground(<color>)
    - Colors are strings, such as "red" or "purple"
      - Can add numbers to end of string for darker colors, e.g., "red2", "red3", "red4"
- ```
win = GraphWin()  
win.setBackground("purple")
```
- Does not return anything to shell
  - Called for its change in **win's** state, i.e., this method is a **mutator**

Oct 29, 2007

Sprenkle - CS111

12

## Modification to Tic-Tac-Toe

- **clone** a vertical line and horizontal line and shift appropriately
- Why clone?
  - Maintain the same properties (color, line-width, length)
  - Simplifies code

Oct 29, 2007

Sprengle - CS111 [tictactoe2.py](#)

13

## Using the Graphics Library

- How do we create an instance of a Rectangle?
- Draw the rectangle?
- Shift the instance of the Rectangle class to the **right** 10 pixels
- What are the x- and y- coordinates of the upper-left corner of the Rectangle now?

Oct 29, 2007

Sprengle - CS111 [rectangle.py](#)

14

## Getting Input from the User

- `<GraphWin>.getMouse()`
  - Returns the user's mouse click as a **Point** object
    - We haven't saved the returned value in our examples yet
  - Example problem: Have the user tell you where to draw a line
    - What do you need from the user?
- Entry objects
  - Get text from user

Oct 29, 2007

Sprengle - CS111 [userDraw.py](#)

15

## Problem: Circle Shift

- Move a circle to the position clicked by the user
  - Repeat five times

Oct 29, 2007

Sprengle - CS111 [circleShift.py](#)

16

## Animation

- Use combinations of the method **move** and the function **sleep**
  - Need to **sleep** so that humans can see the graphics moving
  - Computer would process the **moves** too fast!
- **sleep** is part of the **time** module
  - takes a float representing *seconds* and pauses for that amount of time

Oct 29, 2007

Sprengle - CS111 [animate.py](#)

17

## Animate Moving to User Click

- In X steps, move from the circle's current location to the location clicked by user

Oct 29, 2007

[animate2.py](#)  
[fenway.py](#)

Sprengle - CS111

18

## Colors

- Strings, such as "blue4"
- Can also create colors using the function `color_rgb(<red>,<green>,<blue>)`
  - Parameters in the range [0,255]
  - Example color codes:
    - [http://en.wikipedia.org/wiki/List\\_of\\_colors](http://en.wikipedia.org/wiki/List_of_colors)

Oct 29, 2007

Sprengle - CS111

19

## This Week

- Lab
  - Dictionary, file practice
  - Draw and animate a scene or a face or "something significant" using the **graphics** library
    - Practice using an API
- Start defining **our own** classes and APIs
  - Abstracting away the implementation details for users
- Broader Issue: the Risks of Electronic Voting

Oct 29, 2007

Sprengle - CS111

20