Lecture 7

Chapter 3 Getting Started with Graphics

3.1 The Canvas

3.2 Drawable Objects

Getting started

- go to http://www.cs1graphics.org/,
- download a current version of cs1graphics.py library from our web page, and
- save it in your documents folder, and
- Originally the library was created for Python 2, but later support for Python 3 was added.

Chapter 3 uses this library for graphics applications.

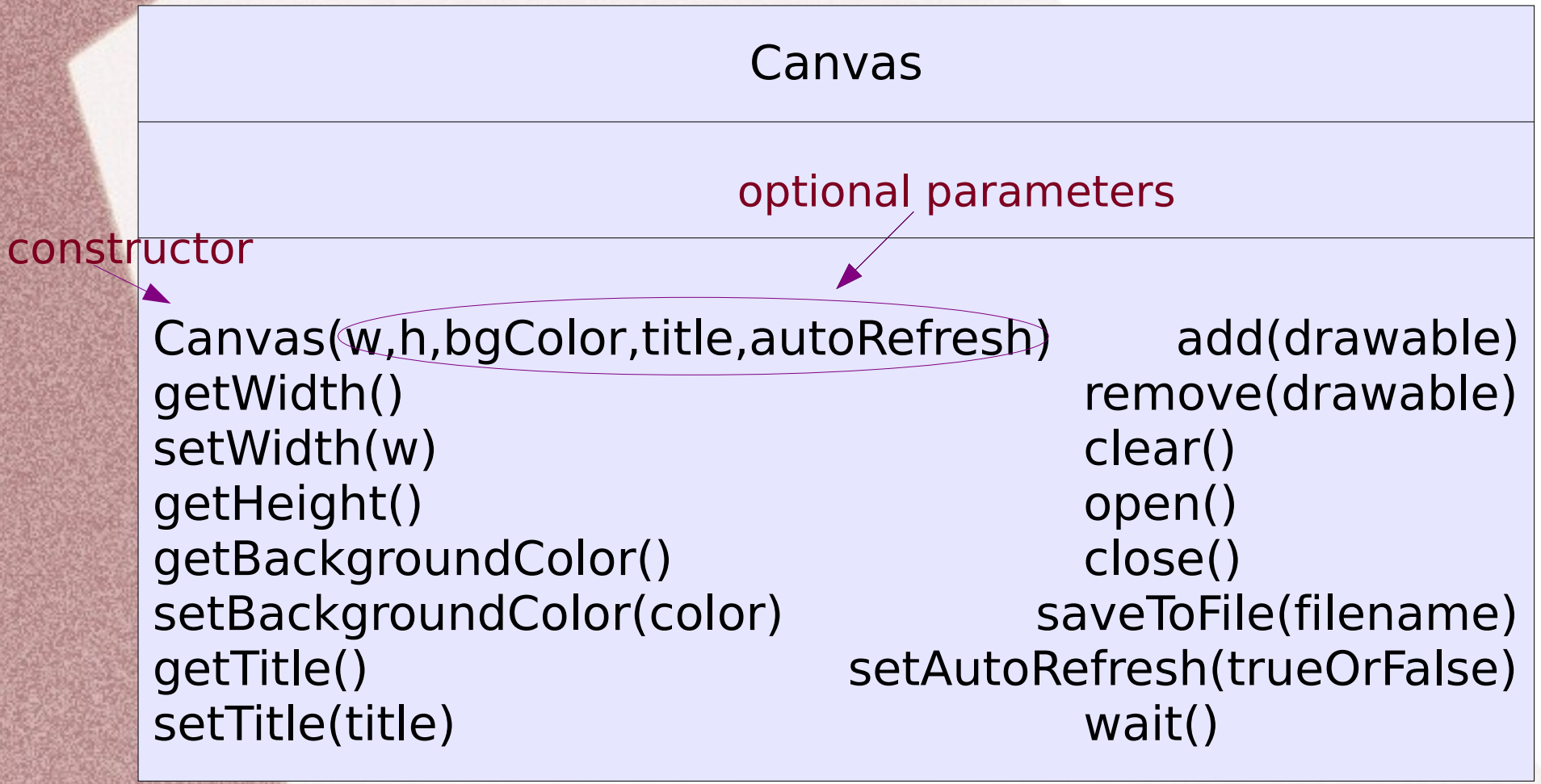
3.1 The Canvas

Class Canvas provides the basic windows for displaying graphics.

Canvas	
Canvas(w,h,bgColor,title,autoRefresh)	add(drawable)
getWidth()	remove(drawable)
setWidth(w)	clear()
getHeight()	open()
getBackgroundColor()	close()
setBackgroundColor(color)	saveToFile(filename)
getTitle()	setAutoRefresh(trueOrFalse)
setTitle(title)	wait()

3.1 The Canvas

Class Canvas provides the basic windows for displaying graphics.



3.1 The Canvas

Example:

```
def main():
    paper=Canvas() #an instance of class Canvas is created,
    # displayed on the screen and assigned to paper

    paper.setBackgroundColor('skyBlue')
    # or (note, that colors of backgrounds are different)
    paper.setBackgroundColor((218,240,248))
    # note, that the RGB color is given as a tuple

    paper.setWidth(600)   # sets the width of the window,
                          # measured in pixels
    paper.setHeight(700) # sets the height of the window,
                          # measured in pixels
    paper.setTitle("First Experiment")
                         # sets the title of the window
```

OR

```
    paper=Canvas(600,700,(218,240,248),"First Experiment")
```

3.1 The Canvas

By default, the `autoRefresh` variable is set to True.

Taken from documentation of cs1graphics.py:

autoRefresh:

When **True** (the default), every change to the canvas or to an object drawn upon the canvas will be immediately rendered to the screen.

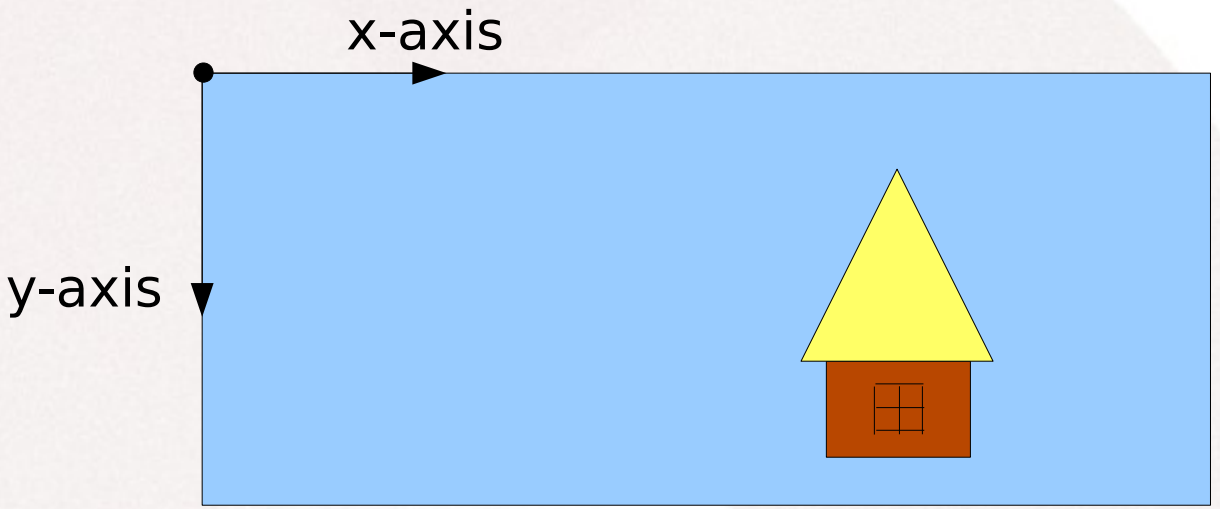
When **False**, all changes are recorded internally, yet not shown on the screen until the next subsequent call to the refresh() method of this canvas. This allows multiple changes to be buffered and rendered all at once.

! Canvas is mutable

3.1 The Canvas

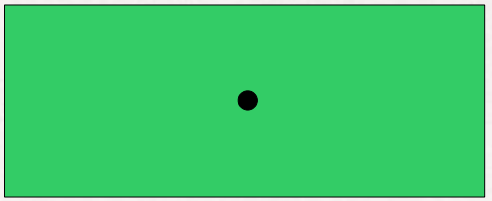
The coordinate system

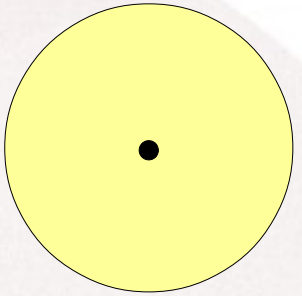
The origin is at the top left corner of the graphics window.



3.2 Drawable Objects

Each drawable object has a reference point. Usually it is the center of an object (for Square, Circle, Rectangle, or Text). For Polygon and Path the reference point is the location of the first point (initially), and for an Image is it the top left corver.





See page 94 for the overview of drawable objects – use it as a reference manual.

Also look through pages 93-99 for review of Circle, square, Rectangle, Polygon, and Path.

The process of drawing of an object, briefly consists of the following steps:

build(define) an object → display it → [do anything to it] → erase (remove) it

3.2 Drawable Objects

Depths

In order to show which objects should be in the front, which ones in the back, and so forth we can use `setDepth(depth)` method.
By default, all objects are assigned a depth value of **50**.

The greater the value the more behind (deeper) is the object.

Example:

```
r=Rectangle(200,100,Point(300,350))

c=Circle(300,Point(300,350))

r.depth(20), r.setFillColor('Red')

c.depth(40), c.setFillColor('Blue')

paper.add(r)

paper.add(c)
```

3.2 Drawable Objects

Depths

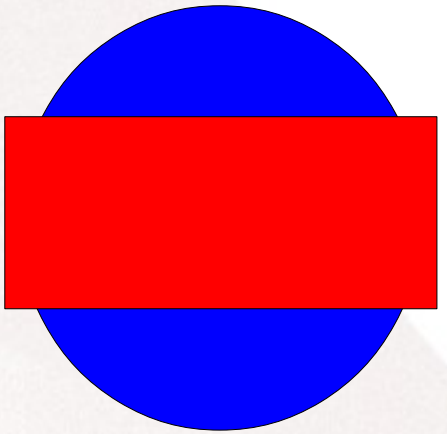
In order to show which objects should be in the front, which ones in the back, and so forth we can use `setDepth(depth)` method.
By default, all objects are assigned a depth value of **50**.

The greater the value the more behind (deeper) is the object.

Example:

```
r=Rectangle(200,100,Point(300,350))

c=Circle(300,Point(300,350))

r.depth(20), r.setFillColor('Red')

c.depth(40), c.setFillColor('Blue')

paper.add(r)

paper.add(c)
```

result:



3.2 Drawable Objects

`Text` and `Image` classes

`Text` class is used for rendering character strings within the drawing area of the canvas.

constructor: `Text(message,fontsize)`

By default, the font size is 12

! The reference point for a text object is aligned with the center of the displayed message and by default is set to Point(0,0). Therefore after the object it defined it is worth moving it or changing the coordinates of the reference point.

Example:

```
m=Text("Hello! How are you?")
m.move(140,50)
paper.add(m)
```

3.2 Drawable Objects

`Text` and `Image` classes

`Image` class provides support for using an image loaded from a file.

constructor: `Image(filename)`

! The reference point is the top left corner and initially set to Point(0,0).

Example:

```
pic=Image("tree.png")
Pic.moveTo(200,200)
paper.add(pic)
```

See example1.py, example1_2, example2.py, example3.py and example4.py

In-class assignment

1. Run Python and check that you can use graphics library (that Python Interpreter sees it)

2. Write a program that creates a graphics window with the title “Abstract Things”

3. In this window draw **one triangle** filled with **any color** (your choice)

4. Draw a **line** and a **rectangle** (filled with any color and not overlapping with the triangle)

5. Then draw a rectangle that overlaps (at least a little bit) both triangle and rectangle and is “above” them, i.e. closer to us than triangle and rectangle.

6. Remove everything except rectangle from the graphics window, and move this rectangle to the right top corner.

Homework Assignment (not for grade)

Write a program that draws a composite object (i.e. not a square, rectangle, or circle) in the middle of the graphics window, then moves this object to the right side of the graphics window, and then to the left side of the graphics window. Movement should be smooth (without jumps). During the movement object should change it's color at least 12 times.

Afterwards in order to notify the user that the program won't do anything else display a text, saying something like "That's it. No more tricks!".

Don't forget to close the graphics window.

Use sleep() method !