PROGRAMMING

Lecture 09

Hanbat National University
Dept. of Computer Engineering
Changbeom Choi

OUTLINE

Scope of a variable

Modules

Graphics

Reading assignment:

Read tutorial on cs1graphics

(Chapter 3 of Object Oriented Programming in Python)

http://www.cs1graphics.org/

SCOPE OF A VARIABLE

```
def test():
    print (a)
    a = 77
    print (a)

a= 99
test()
```

What will happen with this program?

Unfortunately, it will be canceled !

Why ? Variable a is **local** but not **global.**

A **local variable** does not have a **value** when it is used in the first print statement !

What does it mean by local or global ?

Why local?

LOCAL VARIABLES

A **variable** is **local** if its **scope** is **restricted inside** a **function**: It **appears** when a function is **called**, **exists** during **execution** of the function, and **disappears** when the function is **terminated**.

Local variables include:

- **parameters** of a **function**
- **variables** on the **left hand side** of an **assignment** statement in a function

Evaluating ax^2 + bx + c

```
def quadratic( a, b, c, x ):
    quad_term = a * x ** 2
    lin_term = b * x
    return (quad_term + lin_term + c)

print (quaratic(2, 4, 5, 3))
```

a → 2

b → 4

c → 5

x → 3

Which variables are local?

The variables in blue color!

Why?

Global variables

Variables defined **outside** of all functions are called **global** variables. These are **referenced** by their names. Global variables can be used inside a function:

```
def turn_right():
    for i in range(3):
        hubo.turn_left()
hubo = Robot()
turn_right( )
```

Why global variables?

Well, ...convenient sometimes!

Consider a program that evaluates 2*x^2 + 5x + 4
while changing x.

What if a **global variable** is **changed by mistake** ?

```
def qudratic():
   quad_term =a * x ** 2
    lint_erm = b * x
    return  (quad_term + lin_term + c)
a = 2
b = 5
c   =4
```

```
x = 3
print (quadratic())
................
x = 5
print (quadratic())
a = “Joseph”        by mistake
print (a)
......
x= 2
print (quadratic())
```

Unpredictable side effects!

Modular programming

A software development method to **decompose** a **large problem** into **small problems**, **develop and test** the **program** for solving each **small problem, independently,** and **combine all these programs** to construct a **large program.** A small program itself is a function or consists of multiple functions.

In modular programming, the input(**parameters**) of every function and its output(**return values**) should be well-defined. Why ?

Function calls

By explicitly **providing** the **arguments** corresponding to the **parameters**, you do not need to worry about what is happening inside a function.

What if global variables are used instead of parameters?

You should have to **remember** all **global variables** used in a **function** so as to **avoid side effects** by modifying these by mistake.

In large programs, using global variables is dangerous, as they could be modified by mistake.

The program revisited

a is local.

```
def test():
    print (a)
    a = 77
    print (a)

a= 99
test()
print (a)
```

```
def test():
    global a
    print (a)
    a = 77
    print (a)

a= 99
test()
print (a)
```

a is global.

```
def left_turn():
    global hubo_direction
    hubo.turn_left()
    hubo_direction += 90
hubo = Robot()
hubo_direction = 0
left_turn()
print (hubo_direction)
```

```
def turn_right():
    global hubo_direction
    for i in range(3):
        hubo.turn_left()
    hubo.dirction -= 90
hubo = Robot()
hubo_direction = 0
turn_right()
print (hubo_direction)
```

We can change the value of a global variable inside a function by explicitly defining it as global in the function!

```
def f(a):
    print (“a = “, a)
def g():
    a = 7
    f(a+1)
    print (“a = “, a)
a = “Letter a”
print (“a = “, a)
f(3.14)
print (“a = “, a)
g()
print (“a = “, a)
```

Guess what will be printed.

```
a = Letter a
a = 3.14
a = Letter a
a= 8
a = 7
a = Letter a
```

```
def swap(a,b):
    a, b = b, a
x, y = 33, 555
swap(x, y)
print (x, y)
```

What will be printed ?
555 33 or 33 555 ? Why?

Can you fix the program so that it swap the two values ?

```
def swap(a,b):
    a, b = b, a
    

x, y = 33, 555


print (x, y)
```

return a,b

x, y = swap(x, y)

MODULES

A Python **module** is a **collection of functions** that are grouped together in a **file**. Python comes with a large number of **useful modules.**

- **math** for mathematical functions
- **random** for random numbers and shuffling
- **sys** and **os** for accessing the operating system
- **urllib** to download files from the web
- **cs1robots** for playing with robots such as hubo
- **cs1graphics** for graphics
- **cs1media** for processing photos

We can also create our own modules.

You can get information about a module using the help function:

```
>>> help("cs1media")
>>> help("cs1media.picture_tool")
```

Decomposition and Abstraction

With **clear interfaces** to classes or functions in a module, you can easily use these. For example, cs1robots is a module that contains a **class,** Robot. Robot can **easily** be **used** without understanding how it is implemented.

In Python, a **class** generates objects(instances) with **attributes** (data) and **methods**.

Object-oriented programming

How to use a module

```
import math
print (math.sin(math.pi / 4))
```

```
from math import *
print (sin(pi / 4))
print (math.pi)
```

Why an error?

```
from math import sin, pi
print (sin(pi / 4))
print  pi
print (cos(pi/4))
print (math.pi)
```

Why errors?

```
from cs1robots import *
create_world()
hubo = Robot()
hubo.move()
hubo.turn_left()
```

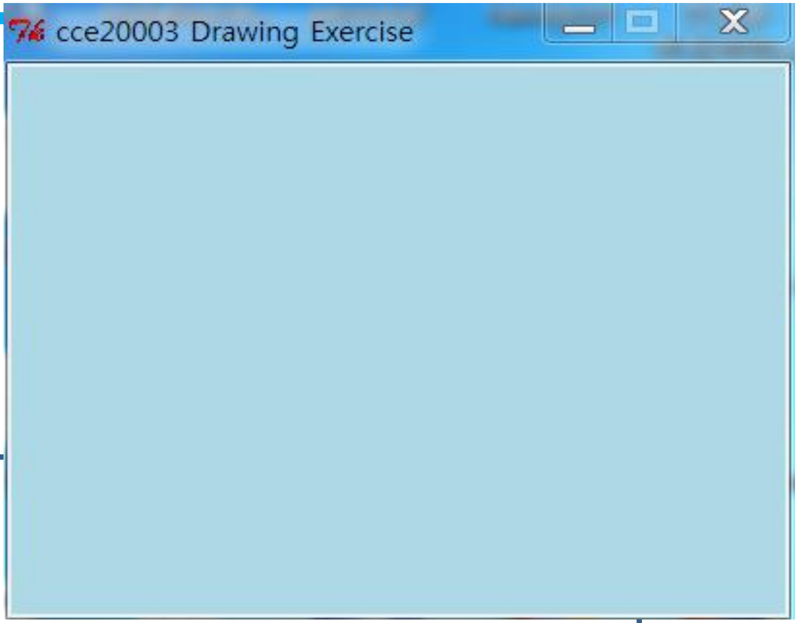
```
import cs1robots
create_world()
hubo = cs1robots.Robot()
hubo.move()
hubo.turn_left()
```

Which one do you prefer ?
Well, ……

The second approach is recommended. Why?

GRAPHICS: CS1GRAPHICS

Creating a canvas to draw on



cce20003 Drawing Exercise

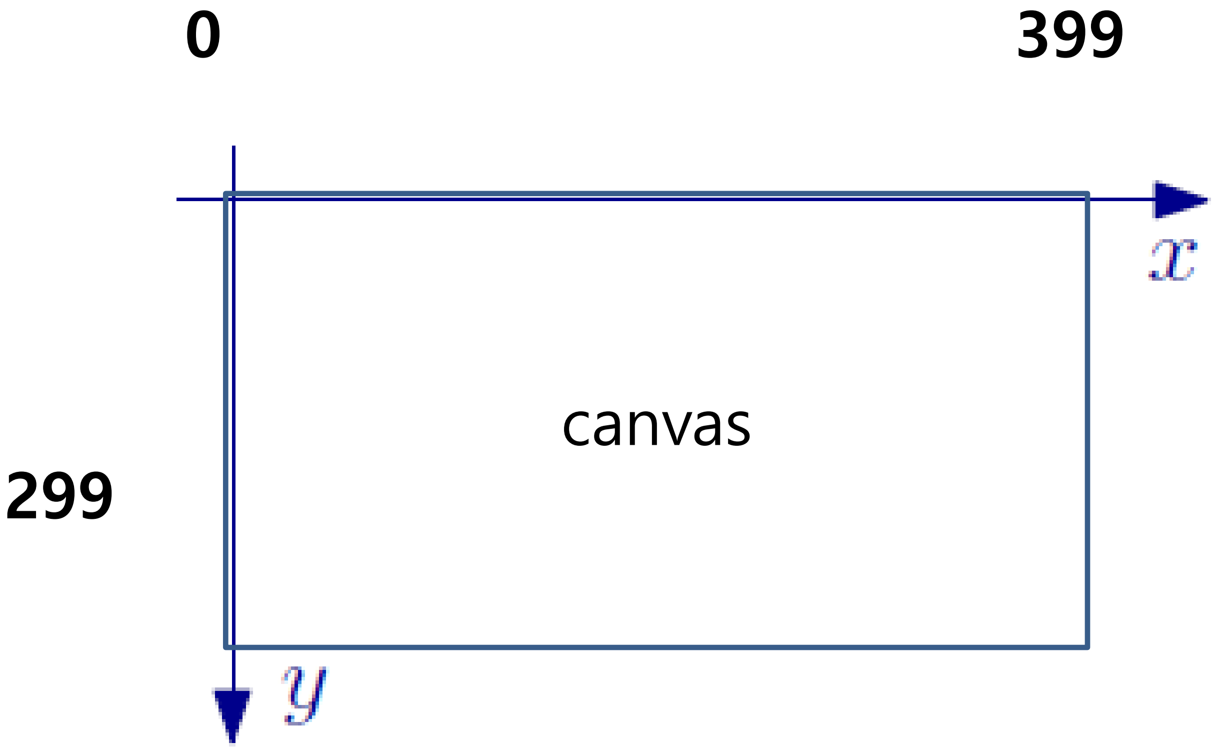
```
from cs1graphics import *
canvas = Canvas(400, 300)

canvas.setBackgroundColor(“light blue”)
canvas.setTitle(“SIT22001 Drawing Exercise”)
```

Reference: http://www.cs1graphics.org/

Read tutorial(Chapter 3)

canvas = Canvas(400,300)



0

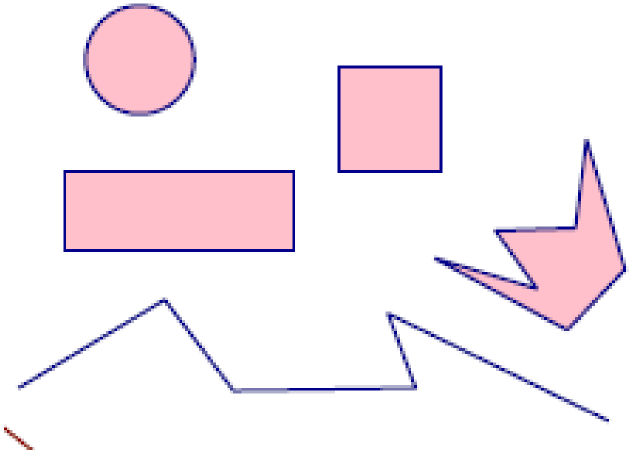
399

canvas

299

DRAWABLE OBJECTS

1. Circle(radius)
2. Square(side)
3. Rectangle(width, height)
4. Polygon(……….)
5. Path(……….)
6. Text(message, font_size)
7. Image(image_filename)



fillable objects

How to draw an object: Square

```
sq = Square(100)
canvas.add(sq)
sq.setFillColor(“blue”)
sq.setBorderColor(“red”)
sq.setBorderWidth(5)
sq.moveTo(200,200)

for i in range(100):
    sq.move(1,0)
```

Add a square to canvas.

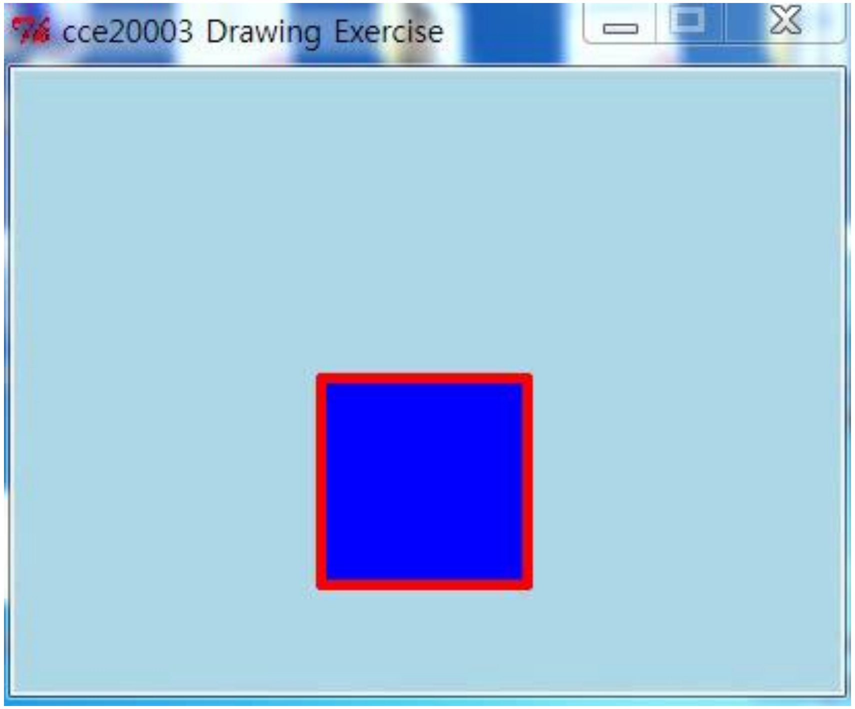
Move the reference point of the square to (200, 200).
(absolute move)

Relative move with respect to the current reference point.

The previous code to initialize the canvas.

```
sq = Square(100)
canvas.add(sq)
sq.setFillColor(“blue”)
sq.setBorderColor(“red”)
sq.setBorderWidth(5)
sq.moveTo(200,200)
```

```
for i in range(100):
    sq.move(1,0)
```

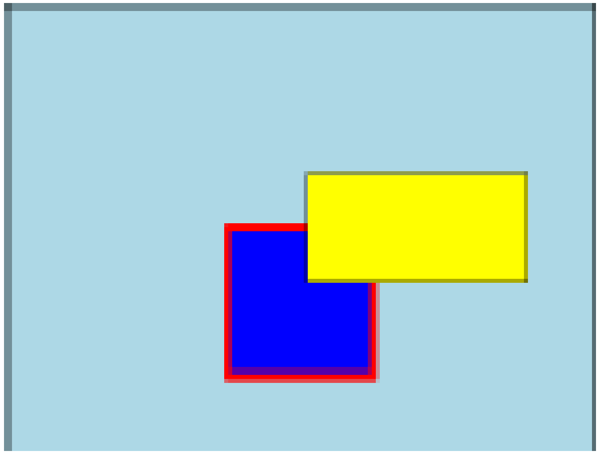


Animation

A **circle** and a **rectangle** can be created in a similar manner. For other drawables read the reference.

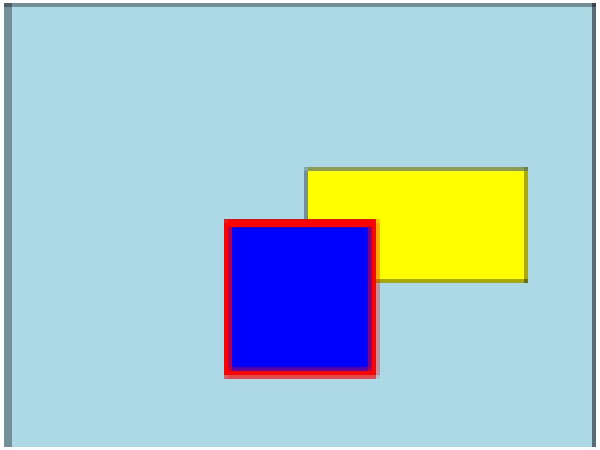
Depth

```
rect = Rectangle(150, 75)
canvas.add(rect)
r.setFillColor("yellow")
r.moveTo(280, 150)
```



Changing depths:

```
sq.setDepth(10)
rect.setDepth(20)
```



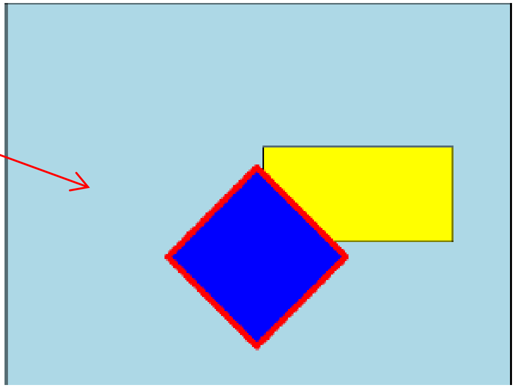
The default value is 50.

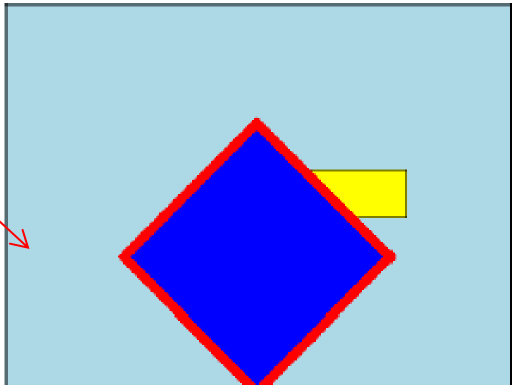
Rotation (wrt the reference point)

```
  sq.rotate(45)
```

Scaling (wrt the reference point)

```
  sq.scale(1.5)
  rect.scale(0.5)
```





Fade-out:

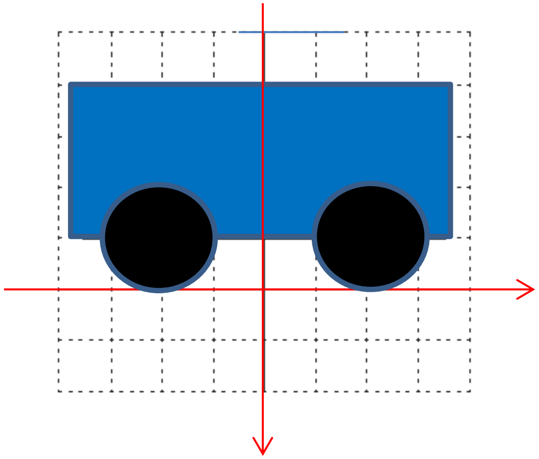
```
  for i in range(80):
     sq.scale(0.95)
  canvas.remove(sq)
```

Mirror flipping (around an axis).

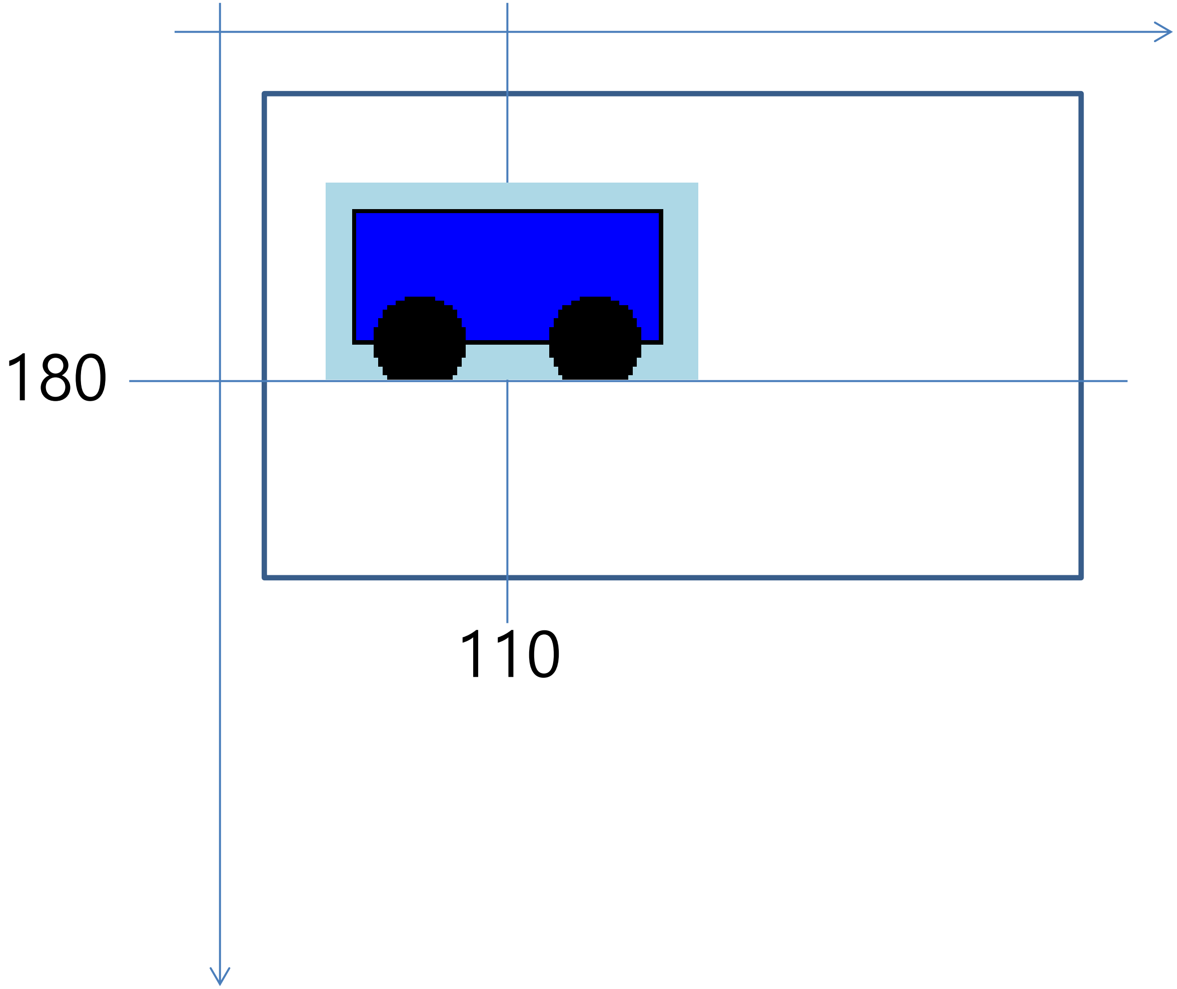
```
   rect.flip(5)
```

Layer: Grouping drawables together

```
car = Layer()
tire1 = Circle(10, Point(-20,-10))
tire1.setFillColor('black')
car.add(tire1)
tire2 = Circle(10, Point(20,-10))
tire2.setFillColor('black')
car.add(tire2)
body = Rectangle(70, 30, Point(0, -25))
body.setFillColor('blue')
body.setDepth(60)
car.add(body)
```

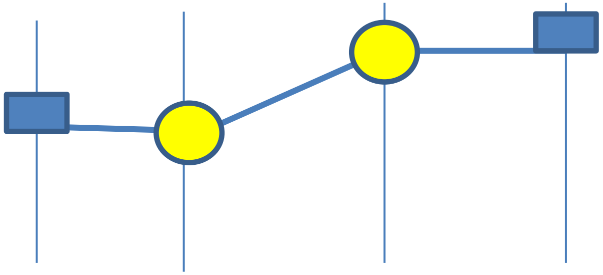


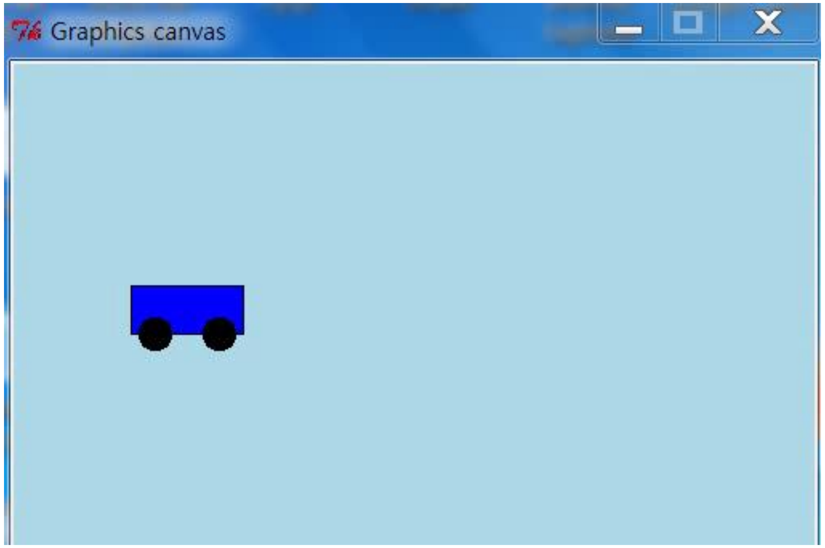
```
car.moveTo(110,180)
car.setDepth(20)
canvas.add(car)
```



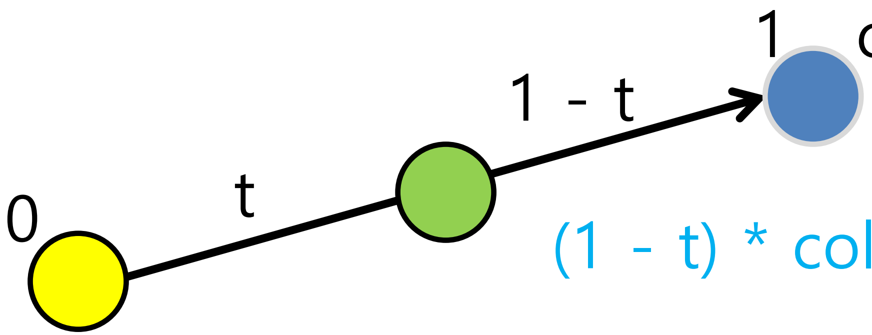
The **whole layer** can be **transformed** as a **single object** !

```
for i in range(50):
   car.move(2, 0)
for i in range(22):
   car.rotate(-1)
for i in range(50):
   car.move(2,-1)
for i in range(22):
   car.rotate(1)
for i in range(50):
   car.move(2,0)
for i in range(10):
   car.scale(1.05)
car.flip(90)
```





Color interpolation



color2 =(r2, g2, b2)

(1 - t) * color1 + t * color2

color1 = (r1, g1, b1)

```
def interpolate_colors(t, color1, color2):
    r1, g1, b1 = color1
    r2, g2, b2 = color2
    return (int((1-t) * r1 + t * r2), int((1-t) * g1 + t * g2),
             int((1-t) * b1 + t * b2))
```

Color conversion

From a color name to an (r,g,b) tuple

```
Color(color).getColorValue()
```

```
print (Color(“red”).getColorValue())
      (255, 0, 0)
```

How about the reverse conversion (from rgb to color name)?

Not available yet. Why?

You shall practice color interpolation and conversion in the next lecture!