Lecture 10

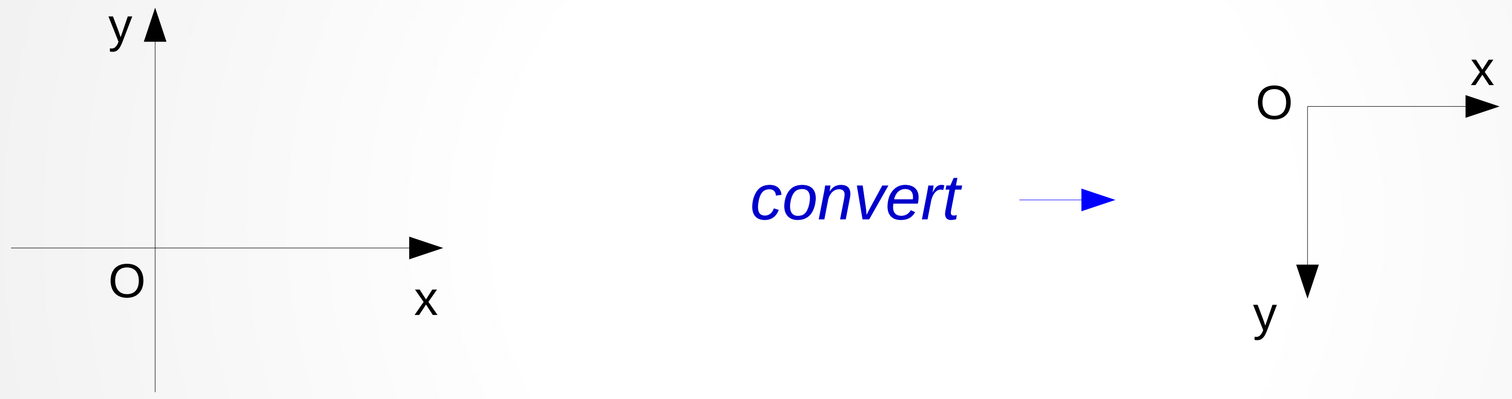
Topics: *Chapter 4. Objects and Graphics*

4.6 Choosing coordinates

4.7 Interactive Graphics

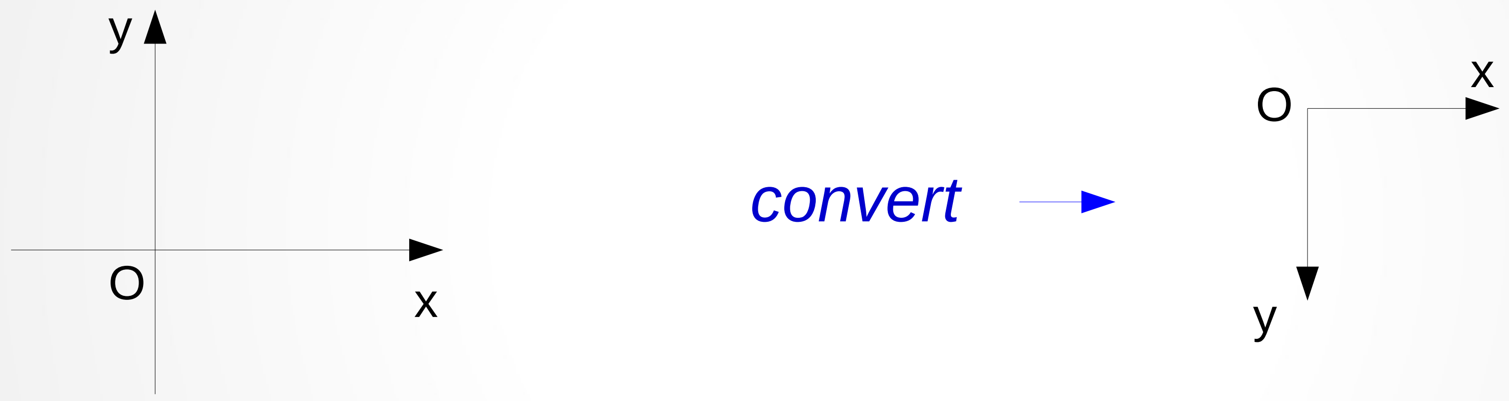
4.6 Choosing coordinates

We discussed that the graphical window coordinates are different from regular rectangular coordinates system. Hence, when we draw anything using graphics library, we have to keep in mind, that the origin is at the top left corner of the window:



4.6 Choosing coordinates

We discussed that the graphical window coordinates are different from regular rectangular coordinates system. Hence, when we draw anything using graphics library, we have to keep in mind, that the origin is at the top left corner of the window:



Another issue: Imagine, that we decided to write a program of tic-tac-toe. We'll start with drawing of 3 rows and 3 columns (i.e. two horizontal lines and two vertical lines). At this point we will have to decide what are the coordinates of the lines, depending on what are the dimensions of the window (the size).

4.6 Choosing coordinates

setCoords

To ease our life class `GraphWin` allows us to specify a coordinate system for the window using the `setCoords` method.

```
setCoords(xll,yll,xur,yur)
```

the lower left corner x- and y-coordinates

the upper right corner x- and y-coordinates

All subsequent drawings will be done with respect to the altered coordinate system (except for `plotPixel`)

see tic-tac-toe_begin.py

4.7 Interactive Graphics

More practice with mouse clicks

Recall `getMouse` method:
- when it is invoked the program pauses and waits the user to click the mouse somewhere in the graphics window, then the spot where the user clicks returned to the program as `Point`.

4.7 Interactive Graphics

More practice with mouse clicks

Recall `getMouse` method:
- when it is invoked the program pauses and waits the user to click the mouse somewhere in the graphics window, then the spot where the user clicks returned to the program as `Point`.

Quadrilateral:
 Let's write a program that asks the user to click a mouse 4 times then draws a quadrilateral with corners at the clicked points:
quadrilateral.py

4.7 Interactive Graphics

More practice with mouse clicks

Quadrilateral:
Let's write a program that asks the user to click a mouse 4 times then draws a quadrilateral with corners at the clicked points:
quadrilateral.py

Design/Algorithm:

```
p_1=win.getMouse()
p_2=win.getMouse()
p_3=win.getMouse()
p_4=win.getMouse()
define polygon by 4 points,
draw polygon
```

```
    See the program quadrilateral.py
```

4.7 Interactive Graphics

Events

Graphical interfaces can be used for input as well as output.

Usually in a GUI environment we can *click on buttons*, *choose items from menus*, *type information* into on-screen text boxes, and so on.

4.7 Interactive Graphics

Events

Graphical interfaces can be used for input as well as output.

Usually in a GUI environment we can *click on buttons*, *choose items from menus*, *type information* into on-screen text boxes, and so on.

When the user moves the mouse, clicks on a button or types a key on the keyboards, this generates an *event*. This *event object* is further sent to an appropriate part of the program to be processed. Then the appropriate action will be taken.

4.7 Interactive Graphics

Event-driven programming

Event-driven programming can be tricky for novice programmers, since it is hard to figure out «who's in charge» at any given moment.

Our graphics module hides the underlying event-handling mechanism and provides two simple ways of getting user input in a `GraphWin`:

- mouse clicks
- textual input

4.7 Interactive Graphics

Event-driven programming

Event-driven programming can be tricky for novice programmers, since it is hard to figure out «who's in charge» at any given moment.

Our graphics module hides the underlying event-handling mechanism and provides two simple ways of getting user input in a `GraphWin`:

- mouse clicks ← *we already saw*
- textual input

4.7 Interactive Graphics

Textual Input

Graphics library has a simple `Entry` object, that draws a box on the screen that can contain text.

```
Entry(centerPoint,width)
```

number of characters of text that can be displayed

It has `setText` and `getText` methods.
`setText(string)` – sets the text of the object to `string`
`getText()` - returns the current string

4.7 Interactive Graphics

Textual Input

Let's write a program that converts centimeters to inches:
The conversion is as follows: 1 in = 2.54 cm

4.7 Interactive Graphics

Textual Input

Let's write a program that converts centimeters to inches:
The conversion is as follows: 1 in = 2.54 cm

Design/algorithm:

```
Draw the input box for centimeters
Draw the button "Convert"
Wait for the mouse click, then
Get the value from the box for centimeters
Convert it cm to in using inches = cm / 2.54
Display the result of conversion
Notify the user on how to terminate the program
```

see conversion.py

Images

Our graphics library can display GIF and PPM images.

Example: let’s do a simple slide show of three pictures

see slideshow.py