Drawing library - ezgraphics

Introduction to Programming

Rae Harbird, r.harbird@ucl.ac.uk

Drawing simple graphics

- We will create simple drawings, using a graphics module, ezgraphics associated with the book Python for Everyone
- It's a simplified version of Python's more complex library module `tk`
- These slides are adapted from teaching content available with the book.
- Documentation for ezgraphics includes information about how to install and use it.

Installing ezgraphics

There are options. Install ezgraphics on your own computer or use Codio where it's already installed.

There is a separate video clip for installing ezgraphics on your computer.

Using the Graphics module

- Open a new file in your favourite editor, make sure to name the file with a `.py` extension
- Paste in the following code and run it:

```python
from ezgraphics import GraphicsWindow

# Create a graphics window (640 x 480 pixels):
win = GraphicsWindow(640, 480)

# Access the canvas contained in the graphics window:
canvas = win.canvas()
canvas.drawRect(15, 10, 20, 30)

# Wait for the user to close the window
win.wait()
```

The canvas

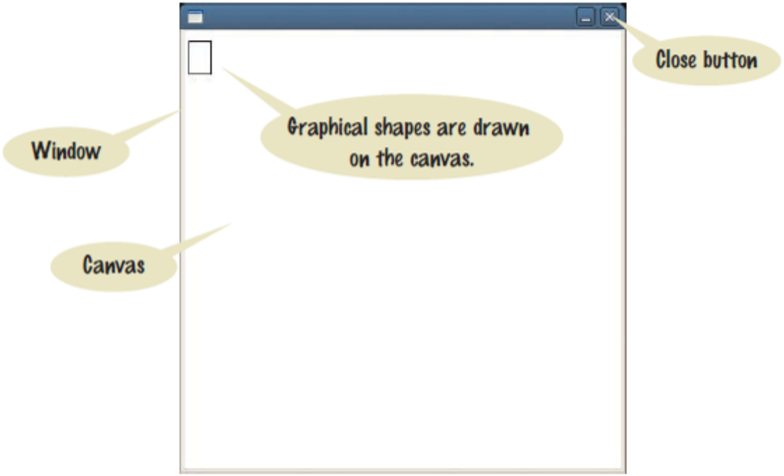


Figure 1: ezgraphics canvas

Canvas coordinates

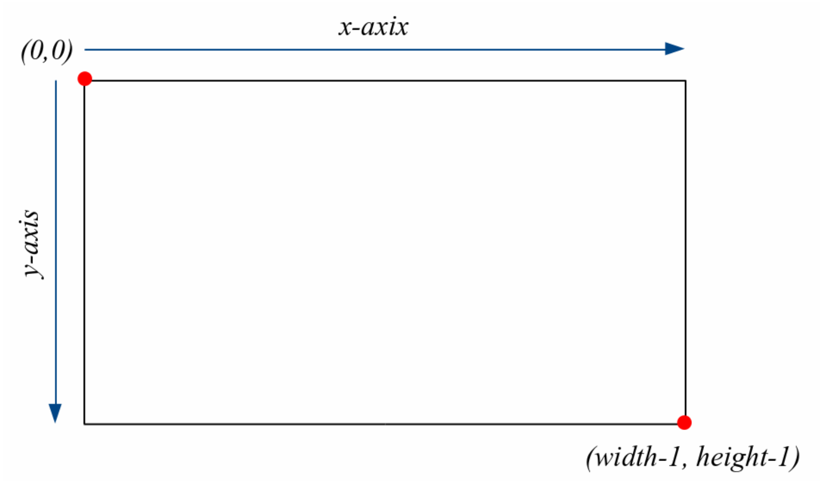


Figure 2: ezgraphics canvas

Drawing shapes

- Basic shapes have 4 properties: x coordinate, y coordinate, width and height

```
canvas.drawRect(15, 10, 20, 30)
```

- Draws a rectangle with the upper top left corner at point ($x = 15$, $y = 10$) in the window with a height of 20 and a width of 30
- Common shapes that can be drawn include: rectangles, squares, circles and ovals

Drawing lines

Lines require slightly different properties to shapes:

A line is two points:

- Point 1(x1 coordinate, y1 coordinate)
- Point 2(x2 coordinate, y2 coordinate)

```
canvas.drawLine(x1, y1, x2, y2)
```

Common drawing methods

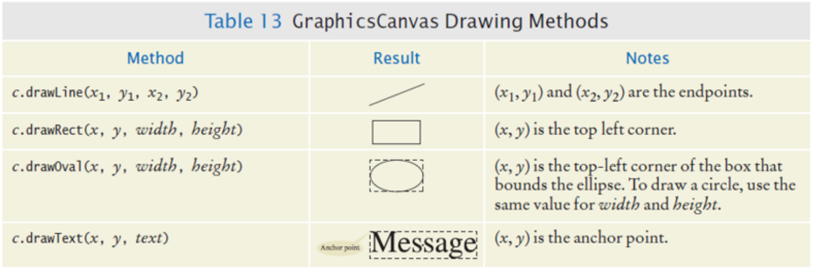


Table 13 GraphicsCanvas Drawing Methods

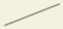
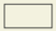
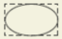
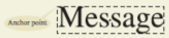
Method	Result	Notes
`c.drawLine(`x_1`, `y_1`, `x_2`, `y_2`)`		(x_1, y_1) and (x_2, y_2) are the endpoints.
`c.drawRect(`x`, `y`, `*width*`, `*height*`)`		(x, y) is the top left corner.
`c.drawOval(`x`, `y`, `*width*`, `*height*`)`		(x, y) is the top-left corner of the box that bounds the ellipse. To draw a circle, use the same value for *width* and *height*.
`c.drawText(`x`, `y`, `*text*`)`	 Message	(x, y) is the anchor point.

Figure 3: Common drawing methods

Methods

Yes, we are using methods … but what are these?

It is enough to know for now that we have created a `canvas` object and that the methods are **verbs** that we can do with a `canvas` object.

You can think of a method as a function but don't forget to use the object name

We'll be coming back to these concepts later in the course when we'll write our own classes.

Draw oval block

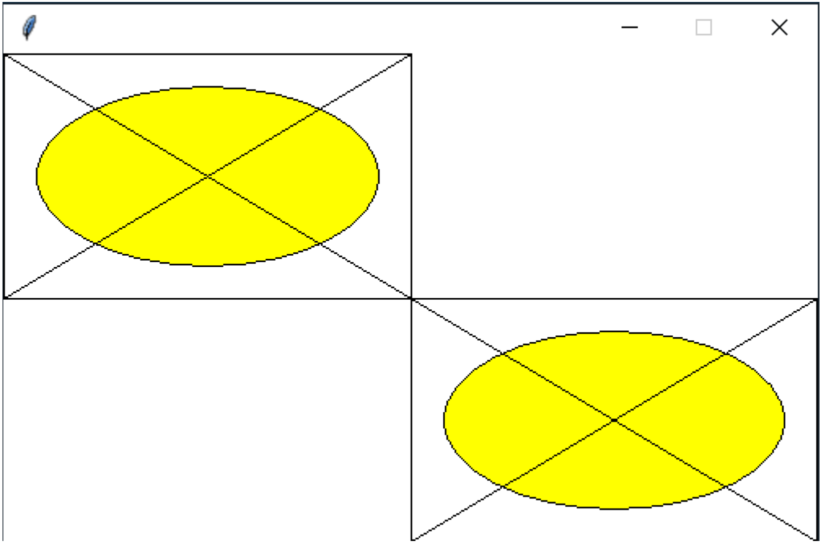


Figure 4: Oval block

Code for oval block

Goal: write a function to draw an oval block as shown above on a given position on the canvas. The oval has a margin of 20 pixels around it.

Function: `def draw_oval_block(canvas, x, y, width, height)`

- `canvas` - the canvas object
- `x` - x coordinate of top left of the shape
- `y` - y coordinate of the top left of the shape
- `width` - width in pixels of the shape
- `height` - depth in pixels of the shape

Assume that there is a constant for the WIDTH and HEIGHT of the canvas and the MARGIN around the oval.

Pixel references

Function: `def draw_oval_block(x, y, width, height)`

- leftmost pixel is x
- rightmost pixel is x + width
- topmost pixel is y
- bottommost pixel is y + height

Plan for oval block

Function: `def draw_oval_block(canvas, x, y, width, height)`

1. Draw rectangle
2. Draw oval
3. Draw cross

Look at the code `draw_oval.py`

Draw rays block

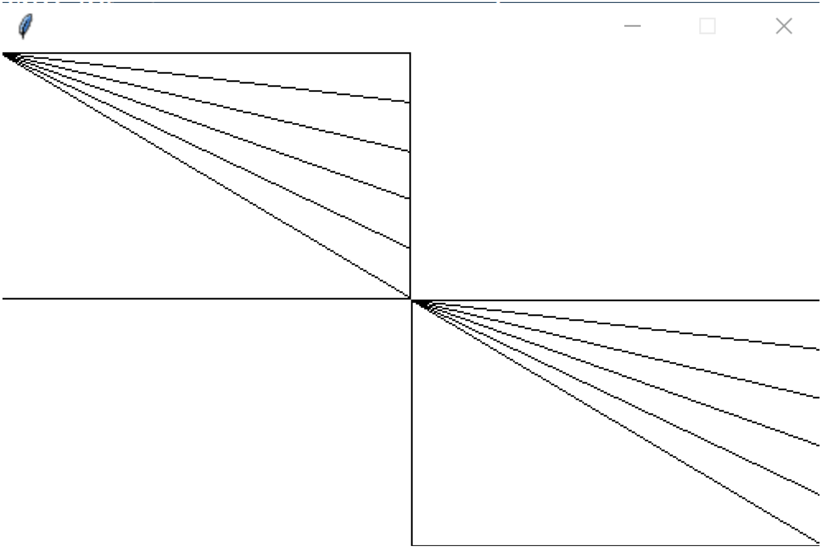


Figure 5: Rays block

Code for rays block

Goal: write a function to draw the rays block as shown above on a given position on the canvas.

Function:

```
def draw_rays_block(canvas, x, y, width, height, number_of_rays)
```

- `canvas` - the canvas object
- `x` - x coordinate of top left of the rectangle
- `y` - y coordinate of the top left of the rectangle
- `width` - width in pixels of the rectangle
- `height` - depth in pixels of the rectangle
- `number_of_rays` - number of rays

Assume that there is a constant for the WIDTH and HEIGHT of the canvas.

Look at the code in `draw_rays.py`