G: Graphics

Gareth McCaughan Revision 1.8, May 14, 2001

Credits

Introduction

This sheet describes briefly all the graphics things (i.e., drawing pictures) you can easily do with Python. (You can actually do lots of other complicated things, but they're, errrm, complicated. We've made a bunch of useful things easy, and these are what this sheet describes.)

For any of this stuff to work, you need to have done

```
        from livewires import *
```

at some point. If you're writing a program, that should be the first line. If you're just typing commands one by one into the "Python Shell" window, you should type the `from livewires ...` thing once; after that, you don't need to do it again unless you quit Python.

There's an introduction to graphics in Sheet 3, so this sheet is more a reference than an introduction. Some of it is *very* terse; ask a leader if it doesn't make any sense. (If you *are* a leader and it still doesn't make any sense, ask Gareth.)

Coordinates

(0,0) is at the bottom left of the window. The window is 640 pixels by 480, by default. (You can make it a different size if you want to.) Coordinates are given in units of one pixel.

Almost all functions that take coordinates as arguments can take them either as two arguments, or as a tuple (x,y). So these two lines are equivalent:

```
        circle(300,200,10)
        circle((300,200),10)
```

Starting and finishing

Before you do anything else, you should say `begin_graphics()`. This will make a new window; the graphics commands will make things happen in that window.

After you're finished, you should say `end_graphics()`. The computer will then wait for the window to be closed.

You can change various things about the window:

`begin_graphics()`	*640 by 480, white background*
`begin_graphics(width=800,height=600)`	*A larger window*
`begin_graphics(background=Colour.black)`	*Black background*
`begin_graphics(title='Walrus')`	*Different window title*

Once the window is there, though, you can't change it.

Drawing things

There's always a "current point". It starts out at (0,0). Many drawing commands start drawing at the current point; most also move the current point.

`clear_screen()`	Empty the graphics window.
`move(x,y)`	Change the current point. Don't draw anything.
`draw(x,y)`	Draw a line from the current point to (`x`,`y`). Change the current point to (`x`,`y`).
`plot(x,y)`	Plot a single pixel at (`x`,`y`). Update current point.
`line(x,y,x,y)`	Draw a line between two endpoints.
`box(x,y,x,y)`	Draw a box given two opposite corners.
`circle(x,y,r)`	Draw a circle given centre and radius.
`text(t)`	Write some text.

Most of these can take an optional "keyword argument" specifying the colour to use: `draw(x,y,colour=red)` .

Some (`box` and `circle`) can take a keyword argument saying whether to fill them in or not (default is not to fill): `circle(300,200,100,filled=1)` .

`circle` can draw circular arcs too. Keyword parameter `endpoints`; should be a list or tuple of length 2, each element being either a point on the screen or an angle. If the first endpoint given is a point, you don't need to give the radius of the circle.

There's also `polygon()`. Argument is a list of coordinates. Keyword arguments: `colour`, `closed`, `filled`.

`text` takes some keyword arguments too. `size`, `serifs`. You can change the default values of these with `set_textsize(sz)` and `set_textserifs(x)`.

Colours

Just as there's a "current point", there's a "current colour". By default, it starts off being black. Colours look like this: `Colour(r,g,b)` where `r`,`g`,`b` are numbers between 0 and 1 indicating how much red, how much green and how much blue there are in the colour you want. So, for instance, `Colour(0,0,0)` is black; `Colour(1,1,1)` is white; `Colour(1,1,0)` is bright yellow.

You can assign colours to variables (i.e., give them names): `yellow = Colour(1,1,0)`, for instance.

To change the current colour, do `set_colour(c)` where `c` is a colour. This function call returns the "current colour" that was in force before you changed it, so you can do things like

```
def draw_red_line():
  old_colour = set_colour(Colour.red)       Colour.red is pre-defined
  move(100,100)
  draw(200,200)
  set_colour(old_colour)                    Leave current colour as it was before we started
```

As that program fragment indicates, there are some colours defined for you: `Colour.red`, `Colour.green`, `Colour.blue`, `Colour.black` and `Colour.white` . In fact, there are some more: `Colour.`*foo* where *foo* is one of: `dark_grey`, `grey`, `light_grey`, `dark_red`, `dark_green`, `dark_blue`, `yellow`, `brown` .

The mouse

We can detect mouse movement and clicks, but doing so may slow things down. If you want to do it, say `mouse_begin()` before trying, and `mouse_end()` when you're done.

`mouse_position()`	Return current mouse position as a 2-tuple.
`mouse_buttons()`	Return a dictionary mapping `'left'` etc to 0 or 1.
`mouse_wait(how)`	Wait for something to happen according to `how`.

Values of `how`: `'down'`, `'up'`, `'change'` wait for suitable value of button state. `'click'` waits for down then up. `'move'` waits for the mouse to move. `'any'` waits for a change in either position or button state.

The keyboard

You can also tell what keys are pressed; this might be useful for games. `keys_pressed()` returns a list containing all the keys currently pressed.

Movable objects

If you say `allow_moveables()` then from then on, all calls to `line`, `box`, `circle`, `text` and `polygon` will return values. These values can be passed to functions that move the corresponding objects around:

`remove_from_screen(object)` removes the object completely

`move_by(object,dx,dy)` moves the object `dx` pixels to the right and `dy` pixels upwards. (`dx` and `dy` can be negative if you like.)

`move_to(object,x,y)` moves the object to the position `(x,y)`.

The "position" of a circle is the position of its centre. The position of a box or line is the *first* pair of coordinates used to create it. The position of a polygon is the first pair of coordinates in its specification.

If you want to go back to immovable objects, say `forbid_moveables()`. Incidentally, you can spell "movable" either with or without the 'e' in the middle. Both spellings are correct.