SimpleGraphics Commands

Window functions

`setSize (width, height)`
Sets the size of the display window. Can be called at any time, although it is called once before any drawing commands are used. Default size of the display is 800 by 600.

`getWidth ()`
Returns the width of the display.

`getHeight ()`
Returns the height of the display.

Fill/outline functions

`setFill(colorstring)` or `setFill (r, g, b)`
Sets the color used to fill shapes. Calling this function has no impact on anything already drawn on the display. Defaults to white.

`setOutline(colorstring)` or `setFill (r, g, b)`
Sets the color used to draw a point, line, or the outline of any shape. Calling this function has no impact on anything already drawn on the display. Defaults to black.

`setBackground(colorstring)` or `setBackground (r, g, b)`
Clears the display and fills the screen with the specified color. Defaults set to light gray.

`setLineWidth(width)`
Sets the width of a point, line, or the outline of any shape. Calling this function has no impact on anything already drawn on the display. Defaults to 1.

Drawing functions

All drawing functions will draw an outline using the color set using `setOutline` and will fill the shape (other than point) using the color set using `setFill`.

`point(x, y)`
Draws a single point at the specified coordinates.

`line (x1, y1, x2, y2)`
Draw a line from the starting coordinates to the ending coordinates.

`triangle (x1, y1, x2, y2, x3, x3)`
Draws a triangle using the three pairs of coordinates as the endpoints.

`rect (x, y, w, h)`
Draws a rectangle using `x`, `y` as the upper left corner with a width of `w` and a height of `h`.

`circle (x, y, r)`
Draws a circle using `x`, `y` as the center of circle and a radius of `r`.

`ellipse (x, y, halfWidth, halfHeight)`
Draws an ellipse (oval) using `x`, `y` as the center of the ellipse, with `halfWidth` being half the ellipses width and `halfHeight` being half the ellipses height.

`blob (x1, y1, x2, y2, x3, x3, ..., `x_n`, `y_n`)`
Draws a smoothed polygon, connecting the first pair of coordinates to the second pair, the second pair to the third pair, and so. The last pair of coordinates will be connected to the first pair or coordinates. Requires at least three pairs of coordinates.

Functions for animations

`setFrameRate (updatesPerSecond)`
Sets the maximum number of times the screen should be redrawn each second (`updatesPerSecond`). Generally used only when creating animations and when auto update is set to `False`.

`setAutoUpdate(autoUpdate)`
Determines if Python updates the screen after each graphics call. Use `update()` to update the screen if auto update is set to false. Defaults to `True`. Auto update is generally set to `False` when creating animations, otherwise is set to `True`.

`update()` – updates the display, showing all requested graphics commands since the last call to update. Should only be used when auto update is set to False. When creating animations, it is generally the last function call inside the loop.

`clear ()`
Clears the display and fills the screen with the specified background color. When creating animations, it is generally the first function call inside the loop.

`isClosed ()`
Returns `True` when the graphics display has been closed, otherwise returns `False`. Generally used only when creating animations. Used to tell the Python code the graphics

Mouse/Keyboard functions

Mouse and keyboard functions will generally only be used when creating animations.

`mouseX()`
Returns the x location of the mouse.

`mouseY()`
Returns the y location of the mouse.

`leftButtonPressed()`
Returns `True` if the left mouse button has been pressed, otherwise returns `False`.

`rightButtonPressed()`
Returns `True` if the right mouse button has been pressed, otherwise returns `False`.

`getKey()`
Returns *and removes* the most recently pressed key. If no key has been pressed since the previous call to `getKey()` `None` is returned.

To perform multiple test on a keystroke (it is the left arrow or the right arrow) store the returned value in a variable and test the variable. Regular keys, keys which have a character (such as `"a"-"z"`, `"A"-"Z"`, `"0"-"9"`, or punctuation characters) are returned as is. Special keys are returned as a string indicating the pressed key.

These strings include: `"F1"`, `"F2"`, …, `"F11"`, `"F12"`, `"Insert"`, `"Home"`, `"Prior"` (PgUp), `"Delete"`, `"End"`, `"Next"` (PgDn), `"Left"`, `"Right"`, `"Return"` (Enter key).

An example using getKey():

```
keyPressed = getKey()
if keyPressed != None:
   if keyPressed == 'a':
      print (“Lowercase a pressed”)
   elif keyPressed == 'A':
      print (“Uppercase a pressed”)
   elif keyPressed == 'Down':
      print (“Down arrow pressed”)
```