Tutorial 1: Drawing with the `SimpleGraphics` Library

1 Introduction

The `SimpleGraphics` library is a custom Python library designed to make graphics in Python as simple as possible while providing the flexibility necessary to create a variety of different applications. With this library it is possible to generate graphical output using as little as two lines of code.

The remainder of this document explores the use of the `SimpleGraphics` library for basic graphical output. Later tutorials will describe how the `SimpleGraphics` library can be used to manipulate images and read keyboard and mouse input.

2 Getting Started

You must download the `SimpleGraphics` library before you can use it in your programs. Save `SimpleGraphics.py` in the same folder as the program you are creating.

Your program must import the `SimpleGraphics` library before using it. Adding the following line of code to the top of your file will import all of the functions stored in the library, allowing you to call them directly.

```
from SimpleGraphics import *
```

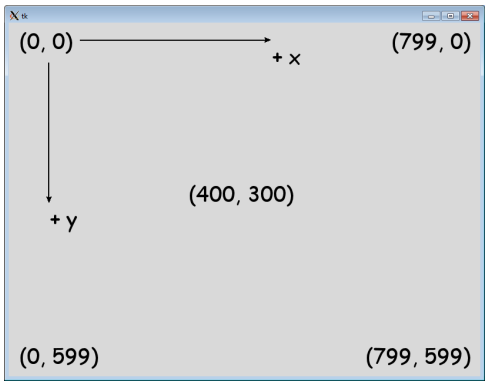
Running a program containing only this line of code will open a blank window. Additional lines of code are necessary to add lines, shapes and text to it. Click on the close button or press the `Esc` key to terminate your program.

2.1 Graphics Basics

The window opened when the `SimpleGraphics` library is imported is 800 pixels wide and 600 pixels tall. Both an x value and a y value are needed to

identify a specific pixel within the window. Some shapes only require one position while others require multiple coordinates. In addition, some shapes like rectangles and ellipses require a width and a height in addition to a location.

In the computer graphics coordinate system $(0, 0)$ is located in the upper left corner of the window. Values on the x-axis increase from left to right while values on the y-axis increase from top to bottom. You may initially find this coordinate system counterintuitive because the y-axis is flipped compared to the Cartesian coordinate system that is widely used in mathematics.



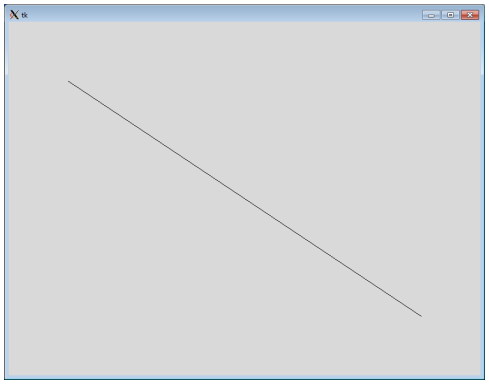
2.2 Drawing Your First Primitive

The `SimpleGraphics` library allows you to draw a variety of different shapes which are referred to as *graphics primitives*, or simply *primitives*. These include lines, rectangles, ellipses, polygons and curves, among others. A complete list of the supported primitives can be found later in this tutorial.

Primitives are drawn by calling functions. For example, a line segment can be drawn by calling the `line` function and providing the two end points of the segment. The first two parameters are the x and y positions of one end of the line while the last two parameters are the x and y positions of the other end of the line. A diagonal line can be drawn extending your program so that it consists of the following two lines of code:

```
from SimpleGraphics import *
line(100, 100, 700, 500)
```

Running this program will generate the following output:



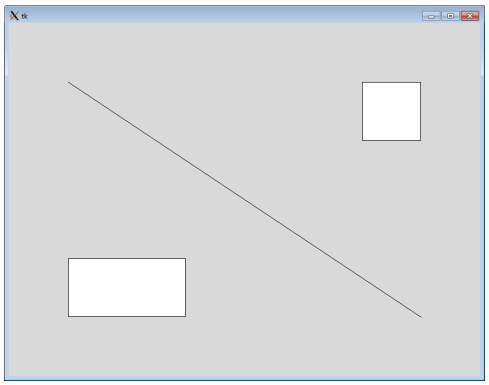
2.3 Adding Rectangles

A rectangle can be added to your application by calling the `rect` function. It requires four parameters. The first two parameters specify the x and y coordinates of its upper left corner. The remaining two parameters are the rectangle's width and height respectively. Adding the following line of code to your program will draw a rectangle below the diagonal line.

```
rect(100, 400, 200, 100)
```

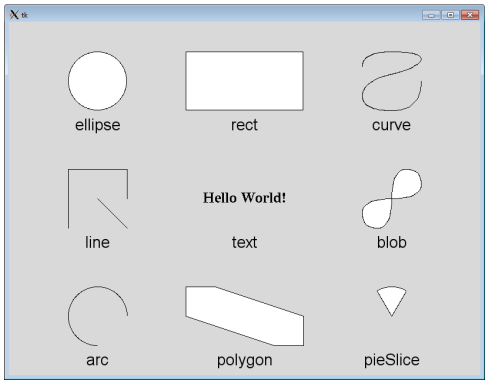
A square can be drawn above the diagonal line by adding the following line of code:

```
rect(600, 100, 100, 100)
```



2.4 Additional Primitives

The `SimpleGraphics` library supports 9 primitives which are shown in the following image.



The code necessary to generate each of these primitives is described in the following subsections.

2.4.1 Line

The `line` function requires a minimum of four parameters. The first two parameters represent the x and y coordinates of one end point of the line segment while the last two parameters represent the coordinates of the other end of the line segment. Multiple line segments can be drawn with a single function call by including addition parameters when calling the `line` function. The number of parameters must always be even, with each pair of parameters representing the x and y coordinate of a point in the window. Line segments are drawn to connecting the first point to the second point, the second point to the third point, and so on.

The following lines of code draw the line segments in the image shown previously:

```
line(150, 300, 200, 350)
line(100, 350, 100, 250, 200, 250, 200, 300)
```

2.4.2 Curve

The `curve` function is closely related to the `line` function described in the previous section. It takes several points as its parameters and then draws a curve that connects the first point to the last point. The curve is pulled toward the other points but does not typically pass directly through them, allowing a smooth shape to be maintained. The code used to draw the curve shown in the previous image follows:

```
curve(600, 75, 600, 50, 700, 50, 700, 75, \
     600, 100, 600, 150, 700, 150, 700, 100)
```

2.4.3 Rectangle

Rectangles are drawn by calling the `rect` function. It requires 4 parameters. The first two parameters represent the upper left corner of the rectangle while the remaining two parameters are its width and height respectively. If the width and height are the same then a square is drawn. The following line of code draws the rectangle shown in previous image:

```
rect(300, 50, 200, 100)
```

2.4.4 Ellipse

Like the `rect` function, the `ellipse` function also takes 4 parameters. Its first two parameters specify the upper left corner of the bounding box for the ellipse while the last two parameters are the width and height of the bounding box respectively. If the width and height of the bounding box are the same then a circle is drawn. The following line of code draws the ellipse shown in the previous image:

```
ellipse(100, 50, 100, 100)
```

2.4.5 Polygon

The `polygon` function can be used to draw irregularly shaped objects. Its parameters represent a collection of points that describe the perimiter of the polygon in the order x_1, y_1, x_2, y_2, ... x_n, y_n. Each point is connected to its two neighbours by a line segment, and the enclosed area is filled. The number of parameters provided to the polygon function must always be even. The polygon shown in the previous image can be drawn using the following line of code:

```
polygon(300, 450, 350, 450, 500, 500, 500, 550, 450, 550, 300, 500)
```

2.4.6 Blob

The `blob` function is used to construct an arbitrary shape with curved edges. Like a polygon, it's parameters represent a collection of points that describe the perimeter of the shape. However, the blob may not actually touch the points in question. Instead, the points provided are used to influence the blob's shape while maintaining a curved edge. Repeating a point in the blob's point list will ensure that the edge of the blob passes through that point. However, this typically results in a sharp point on the edge of the

blob. The following line of code was used to draw the blob in the previous image:

```
blob(600, 350, 600, 300, 700, 300, 700, 250, 650, 250, 650, 350)
```

2.4.7 Arc

The `arc` function requires six parameters. The first four parameters describe the bounding box for an ellipse, a portion of which will be drawn to form an arc, the same way the bounding box is specified for the `ellipse` function. The final two parameters specify the starting angle of the arc and its extent in degrees. The starting angle is specified with 0 degrees representing the 3 o'clock position on a clock face. The extent of the arc specifies the number of degrees counter clockwise over which the arc will be drawn. For example, the three quarters circle show in the previous figure can be drawn with the following command:

```
arc(100, 450, 100, 100, 0, 270)
```

2.4.8 Pie Slice

A pie slice can be drawn by calling the `pieSlice` function. It takes the same parameters as the `arc` function described in the previous section. When the pie slice is drawn, two line segments are drawn in addition to the arc. These lines segments connect the ends of the arc to the center of the circle and the enclosed region is filled. The pie slice shown in the previous figure can be drawn with the following function call:

```
pieSlice(600, 450, 100, 100, 60, 60)
```

2.4.9 Text

Text can be added to your image by calling the `text` function. The text function takes a minimum of three parameters which are the x and y coordiantes of the text, followed by the message that will be displayed.

A variety of fonts can be used for the displayed text. To use a font other than the default, call the `setFont` function. It's first parameter is the name of the font that you would like to use such as `Times` or `Arial`. The second parameter, which is optional, is the size of the font. If no size is specified then the font size defaults to 10. The final parameter is a string describing any modifications that should be performed to the font. Available modifications include `"bold"`, `"italic"` and `"bold italic"`.

The "Hello World!" message shown in the previous figure was drawn using the following lines of code:

```
setFont("Times", "24", "bold")
text(400, 300, "Hello World!")
```

By default, the text is centered on the specified location. However, this behaviour can be undesirable when you are trying to line up several lines of text of different lengths. As a result, a optional fourth parameter can be included which specifies where the coordinate is within the text. The supported values are the abbreviations `"n"`, `"e"`, `"s"` and `"w"` for north, east, south and west. For example, if the fourth parameter to text is `"w"` then the anchor point is on the west (left) edge of the text, and all of the text will appear to the right of the x coordinate that was provided as the first parameter. The anchor point can be placed in the corner by using two directions including `"ne"`, `"se"`, `"nw"` and `"sw"`.

2.5 Working with Color

All of the primitives that we have drawn so far can be drawn using a wide variety of colors. Filled shapes such as polygons, rectangles and blobs make use of both an outline color and a fill color. Unfilled shapes such as lines, curves and text make use of only the outline color.

alice blue	blue3	coral1	dark orchid	gold3	ivory4
antique white	blue4	coral2	dark red	gold4	khaki
aquamarine	brown	coral3	dark salmon	goldenrod	khaki1
aquamarine1	brown1	coral4	dark sea green	goldenrod1	khaki2
aquamarine2	brown2	cornflower blue	dark slate blue	goldenrod2	khaki3
aquamarine3	brown3	cornsilk	dark slate gray	goldenrod3	khaki4
aquamarine4	brown4	cornsilk1	dark turquoise	goldenrod4	lavender
azure	burlywood	cornsilk2	dark violet	green	lavender blush
azure1	burlywood1	cornsilk3	deep pink	green yellow	lawn green
azure2	burlywood2	cornsilk4	deep sky blue	green1	lemon chiffon
azure3	burlywood3	cyan	dim gray	green2	light blue
azure4	burlywood4	cyan1	dodger blue	green3	light coral
beige	cadet blue	cyan2	firebrick	green4	light cyan
bisque	chartreuse	cyan3	firebrick1	honeydew	light goldenrod
bisque1	chartreuse1	cyan4	firebrick2	honeydew1	light goldenrod yellow
bisque2	chartreuse2	dark blue	firebrick3	honeydew2	light gray
bisque3	chartreuse3	dark cyan	firebrick4	honeydew3	light green
bisque4	chartreuse4	dark goldenrod	floral white	honeydew4	light pink
black	chocolate	dark gray	forest green	hot pink	light salmon
blanched almond	chocolate1	dark green	gainsboro	indian red	light sea green
blue	chocolate2	dark khaki	ghost white	ivory	light sky blue
blue violet	chocolate3	dark magenta	gold	ivory1	light slate blue
blue1	chocolate4	dark olive green	gold1	ivory2	light slate gray
blue2	coral	dark orange	gold2	ivory3	light steel blue

Colors can either be specified by their name or by a combination of primary colors. The simple graphics library understands the names of several hundred colors, some of which are listed in this document.

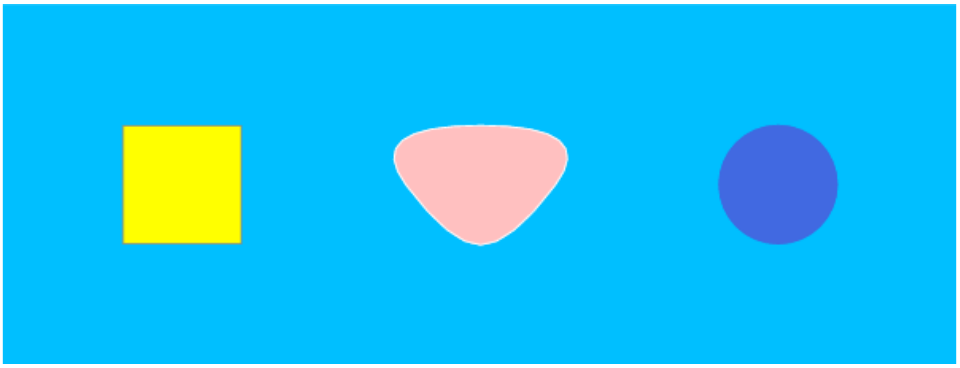
While the listed colors cover a wide variety of possibilities, they may not include the exact color that you need. As a result, it is also possible to form a color by mixing various amounts of red, green and blue. The amount of each color is specified on a scale from 0 (no contribution) to 255 (maximum contribution). All of the color functions provided by the `SimpleGraphics` library support both methods for entering a color.

Call the `setOutline` function to change the outline color used when drawing shapes. If only one parameter is specified then the parameter should be a string containing the name of a color. If three parameters are provided then each parameter should be a numeric value between 0 and 255. In the 3 parameter form the first parameter is the amount of red, the second parameter is the amount of green and the third parameter is the amount of blue.

light yellow	misty rose	peru	saddle brown	snow3	violet
lime green	moccasin	pink	salmon	snow4	violet red
linen	navajo white	pink1	salmon1	spring green	wheat
magenta	navy	pink2	salmon2	steel blue	wheat1
magenta1	navy blue	pink3	salmon3	tan	wheat2
magenta2	old lace	pink4	salmon4	tan1	wheat3
magenta3	olive drab	plum	sandy brown	tan2	wheat4
magenta4	orange	plum1	sea green	tan3	white
maroon	orange red	plum2	seashell	tan4	white smoke
maroon1	orange1	plum3	seashell1	thistle	yellow
maroon2	orange2	plum4	seashell2	thistle1	yellow green
maroon3	orange3	powder blue	seashell3	thistle2	yellow1
maroon4	orange4	purple	seashell4	thistle3	yellow2
medium aquamarine	orchid	purple1	sienna	thistle4	yellow3
medium blue	orchid1	purple2	sienna1	tomato	yellow4
medium orchid	orchid2	purple3	sienna2	tomato1	
medium purple	orchid3	purple4	sienna3	tomato2	
medium sea green	orchid4	red	sienna4	tomato3	
medium slate blue	pale goldenrod	red1	sky blue	tomato4	
medium spring green	pale green	red2	slate blue	turquoise	
medium turquoise	pale turquoise	red3	slate gray	turquoise1	
medium violet red	pale violet red	red4	snow	turquoise2	
midnight blue	papaya whip	rosy brown	snow1	turquoise3	
mint cream	peach puff	royal blue	snow2	turquoise4	

The fill color can be changed by calling the `setFill` function. It accepts parameters in the same form as the `setOutline` function. If you want to change both the fill color and the outline color to the same color then you can call the `setColor` function. The background color for the window can be changed by calling the `background` function. It also takes a color as its

parameter or parameters. The following program demonstrates the use of color by changing the background color of the window and drawing three primitives.



```
from SimpleGraphics import *

# Change the background to sky blue
background("deep sky blue")

# Draw a square using named colors
setOutline("seashell4")
setFill("yellow")
rect(100, 100, 100, 100)

# Draw a blob using colors specified by their red, green and blue values
setOutline(255, 255, 255)
setFill(255, 192, 192)
blob(300, 100, 500, 100, 425, 200, 375, 200)

# Draw an ellipse using the same color for the outline and fill
setColor("royal blue")
ellipse(600, 100, 100, 100)
```