**Graphics in RAPTOR** [Adapted from [**https://raptor.martincarlisle.com/**](https://raptor.martincarlisle.com/) by Dr. Wayne Brown]

**Process Abstraction**

An abstraction is an idea without details. To reinforce what abstraction is, consider the following examples. The "abstract" of a technical paper explains the overall content of a document but leaves all the details to the body of the paper. An "abstract art" painting is typically a "blob" of colors that, when looked at with some imagination, can be construed to be some recognizable, real-world object. Your imagination adds the required details needed to transform the blobs into something recognizable. Consider the two images below, both of which are images of a stealth fighter. One contains fine-detail and the other is abstract.

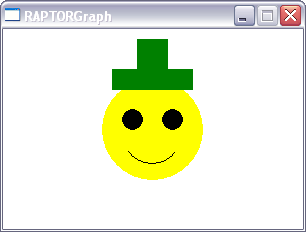
[[1]](#footnote-1) [[2]](#footnote-2)

In computer science, abstraction is a key element of problem solving. We know from human physiological studies that the average human brain can only actively think about approximately 7 things at one time. To solve complex problems, you must be able to think about the "big issues" of the problem without considering all the details related to each individual issue. In computer programming we abstract away details by grouping series of related instructions into separate and distinct units called ***procedures***.

As an introduction to the concept of procedures, the remaining discussion of this reading introduces a set of procedures that will allow you to draw simple graphic objects on a computer screen. Drawing graphical objects on a computer screen actually takes a considerable amount of work. But algorithms for doing this work have been developed in the past and there is no need for us to "reinvent the wheel." Therefore, if you want to draw a line on the screen, just call a procedure that contains all the detailed instructions required to draw the line. This frees up your thinking so that you can concentrate on a higher-level view of your problem. By "abstracting away" the details of drawing a line, we can think about our current problem-solving task at a "higher level of abstraction." Again, this is a key element in problem solving. As you develop simple computer programs that produce graphical output, hopefully you will gain an appreciation for "procedural abstraction."

### 

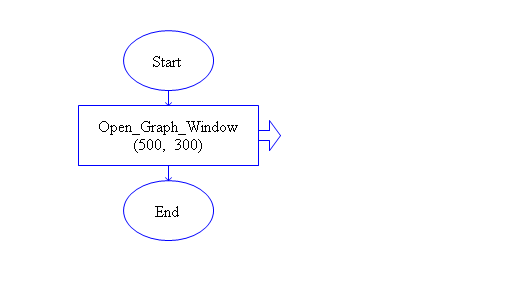
### Overview of RAPTORgraph

RAPTORgraph is a set of pre-defined procedures for drawing graphical objects on a computer screen. All RAPTORgraph commands are relative to a special graphics window, an example of which is depicted in the figure below. You can draw lines, rectangles, circles, arcs and ellipses   
of various sizes and colors into a graphics window. You can also display   
text in the graphics window.

You can also interact with a graphical program by determining the position of the mouse in the graphics window and determining if and where a mouse button or keyboard key was clicked. The following sections will teach you about

* the graphic window,
* the drawing commands, and
* user interactions.

### The Graphics Window

To use RAPTORgraph, you must open a graphics window. This graphics window must be created prior to calling any other RAPTORgraph procedures or functions.

**Open\_Graph\_Window (X\_Size, Y\_Size)**

If you used the above procedure call in your program, a graphics window 500 pixels wide by 300 pixels high would be created. Such a graphics window is depicted in the figure below.

+X axis

+Y axis



The graphics window always begins with a white background. The origin of the graphics window’s (X,Y) coordinate system is at the bottom left-hand corner of the window. The X-axis starts from 1 going left to right. The Y-axis starts from 1 going bottom to top.

### Drawing Commands

RAPTORgraph has a number of procedures that draw shapes in the graphics window; some examples are summarized in the table below. All graphic commands draw on top of any previously drawn graphics. Therefore, the order in which shapes are drawn is important. All of the graphic procedures require a set of arguments (or parameters) that specify where the shape is to be drawn, how big it is, its color, and, if it covers a region, whether it is an outlined or a solid shape.

|  |  |
| --- | --- |
| **Shape** | **Procedure Call with Description** |
| line | Draw\_Line(X1, Y1, X2, Y2, Color)  Draws a straight line between (X1,Y1) and (X2,Y2) of the specified color. |
| rectangle | Draw\_Box(X1, Y1, X2, Y2, Color, Filled/Unfilled)  Draws a rectangle by specifying any corner of the box, (X1,Y1) and the opposite corner, (X2,Y2). |
| circle | Draw\_Circle(X, Y, Radius, Color, Filled/Unfilled)  Draws a circle given its center (X,Y) and its radius. |
| draw text | Display\_Text(X, Y, Text, Color)  Draws the characters in the text string, where the (X,Y) location is the upper-left corner of the first drawn character. Text is always drawn from left to right, horizontally across the window. |

To specify the color of an object, use one of the following values:

|  |  |
| --- | --- |
| White  Black  Red  Blue  Green  Cyan  Magenta  Yellow  Brown | Light\_Gray  Dark\_Gray  Light\_Blue  Light\_Green  Light\_Cyan  Light\_Red  Light\_Magenta |

To “draw” an image in the drawing area of a graphics window, use the following procedure.

|  |  |
| --- | --- |
| **Action** | **Procedure Call with Description** |
| draw an image | Draw\_Bitmap(Bitmap, X, Y, Width, Height)  Draws an image (loaded by a call to Load\_Bitmap), where (X,Y) specifies the upper-left corner of the image and Width and Height specify the amount of the image to draw. |

The arguments (or parameters) to the drawing procedures must be specified in the order they are defined. In addition, an argument can be one of three things:

* a numerical or string constant,
* a variable that contains an appropriate value, or
* an equation that calculates an appropriate value.

An example of each case is given next

|  |  |  |
| --- | --- | --- |
| **Constant Arguments:** This example shows the use of numerical constants to specify the center point and radius of the 3 circles. Note that the order of the drawing (biggest to smallest circle) is very important. If you drew the circles in any other, you would not get the same final drawing. |  |  |
| **Variable Arguments:** This example shows the use of variable arguments to specify the center point of the 3 circles. The picture that is shown to the right was drawn after the user entered a value of 80 for xCenter and a value of 120 for yCenter. |  |  |
| **Equation Arguments:** This example shows the use of equations to specify the radius of 3 circles. The picture that is shown to the right was drawn after the user entered a value of 90 for Initial\_radius and a value of 0.8 for Radius\_ratio. |  |  |

**User Interactions**

To make a graphics program interactive (so that the user is able to direct what happens in the window) you will need to use one or more "input" functions or procedures. Some input commands will halt your program until the user input occurs. These are sometimes referred to as "blocking" functions because they block your program from executing until the user input occurs. Other input commands get current information about the mouse or keyboard but do not halt the execution of your program. The following is a brief description of two popular mouse input commands.

|  |  |
| --- | --- |
| **Input** | **Procedure Call with Description** |
| wait for the press of a mouse button | Wait\_For\_Mouse\_Button(Which\_Button)  A procedure that simply waits until the specified mouse button (either Left\_Button or Right\_Button) is pressed. |
| wait for the press of a mouse button | Get\_Mouse\_Button(Which\_Button, X, Y)  A procedure that takes a button (either Left\_Button or Right\_Button) and returns the coordinates of a click of that button. If no click is ready to be processed, it waits until the user presses the desired button. For example, Get\_Mouse\_Button(Right\_Button, My\_X, My\_Y) waits for a click of the right mouse button and then puts the location of the click into the variables My\_X and My\_Y. |

1. http://a51.janus.dk/ [↑](#footnote-ref-1)
2. revised from a drawing from http://www.cncden.com/fan\_art3/Stealth\_Fighter.jpg [↑](#footnote-ref-2)