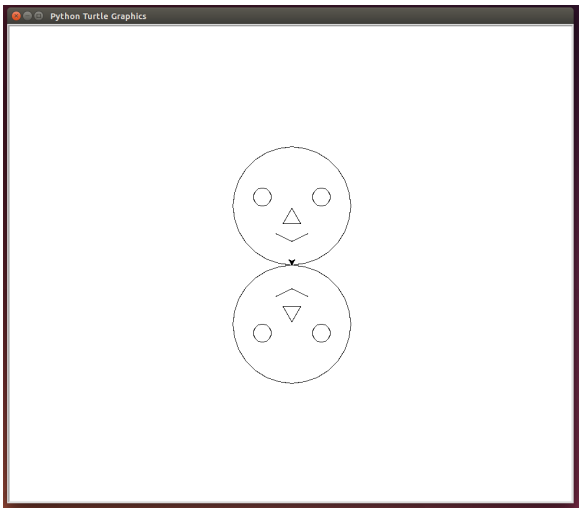
Computer Science I
Turtle Drawing

CSCI-141
Lecture

08/20/2015

1 Problem Statement



Using Python's turtle graphics module, we would like to design a program that draws a simple face, worthy of a stick figure. To emphasize "code reuse," the program will ultimately draw two faces.

2 Analysis and Solution Design

We will utilize commands commonly found in turtle drawing packages. Turtles are small imaginary objects that have pens connected to their undersides. Examples of what the turtle can be told to do include:

- move forward or backward a certain distance;
- move in a circle (Python always draws them counter-clockwise (CCW));
- turn a certain number of degrees left (CCW) or right (CW);
- put its pen up or down. If its pen is down, the turtle will draw a line as it moves, revealing its path.

Our design will involve these activities:

- Define a *home position* for the turtle.
- Describe, via *algorithms*, how to draw each feature of the face by leaving the home position, drawing the feature, and then returning to the home position.

Home Position: The center of the turtle's canvas, with the turtle facing upward (North) and its pen up is the desired home position for this problem.

By defining a home position, we make it easier to add, remove, or rearrange the order of the feature-drawing algorithms. This flexibility has a cost: the extra movement from and to the home position causes our design to be somewhat less efficient than a brute force design. Flexibility of design and performance/efficiency are software qualities that are often at odds.

2.1 Algorithms

An *algorithm* is a special set of instructions. In the words of David Berlinski[1],

```
An algorithm is

a finite procedure,

written in a fixed symbolic vocabulary,

governed by precise instructions,

moving in discrete steps, 1, 2, 3, ...,

whose execution requires no insight, cleverness,
intuition, intelligence, or perspicuity,

and that sooner or later comes to an end.
```

Below we define two algorithms. One will put the turtle in its home position, and the other will draw the entire face. Each algorithm is decomposed into a series of smaller steps. Executing the whole program consists of calling a procedure to execute the first algorithm, followed by calling a procedure to execute the second algorithm.

Initialization Algorithm

- move turtle to home position
- rotate turtle north
- lift pen up

Face-Drawing Algorithm

- draw the outline of the face
- draw the mouth
- draw the nose
- draw the eyes

2.2 Implementation

We'll see that for this program each step contained in the Initialization algorithm can be implemented using a built-in command from the turtle drawing package. The steps of the

1. **Advent of the Algorithm**. Harcourt. ISBN 0156013916 / 9780156013918 / 0-15-601391-6.

Face-Drawing algorithm must be expanded when the algorithm is fully written in Python code. You will see in the solution each 'draw' command translated into its own procedure.

It's important to have documentation on the language and the modules we need to use to solve this problem. There are a number of links to Python language documentation on the course's resources page, `http://www.cs.rit.edu/~csci141/resources.html` You may find the "Summary Sheet" a good place to start. Information on the functions and procedures of the turtle module is at `http://docs.python.org/py3k/library/turtle`

See the accompanying source file, `smiling_faces_py.txt`. *Note: The file has a* `.txt` *suffix because browsers may try to execute Python source files.*

Note: When the turtle software is loaded, the turtle is already at the location we desire for this program, and it is facing East. All that needs to be done is to orient the turtle correctly and pull its pen up.

3 Testing (Test Cases, Procedures, etc.)

Large software programs have a practically infinite number of different ways they might execute, depending on input supplied to them. It is quite an art to choose a small set of tests that cover all the functionality in the software. Since this program is very small, this is not hard to do. Here are the two test cases.

1. Does a single face get drawn correctly?
   Call the Face-Drawing procedure with the turtle at its starting position.
2. Does it draw a face correctly, no matter where the turtle starts, and no matter how many times the face-drawing procedure is executed?
   Change the turtle's position and orientation and call the Face-Drawing procedure again; the program should draw a new face at that new location and orientation.