Prep C
Speak activity

Dr. Madhav Rao

July 17, 2017

Introduction to speak activity

In this activity, you are to write a program, which will speak by linking your program to other *speak.c* module.

Task 1: Speak

Create a directory named *speak* that hangs off your *home* directory. Save the following code in *speak.c* file.

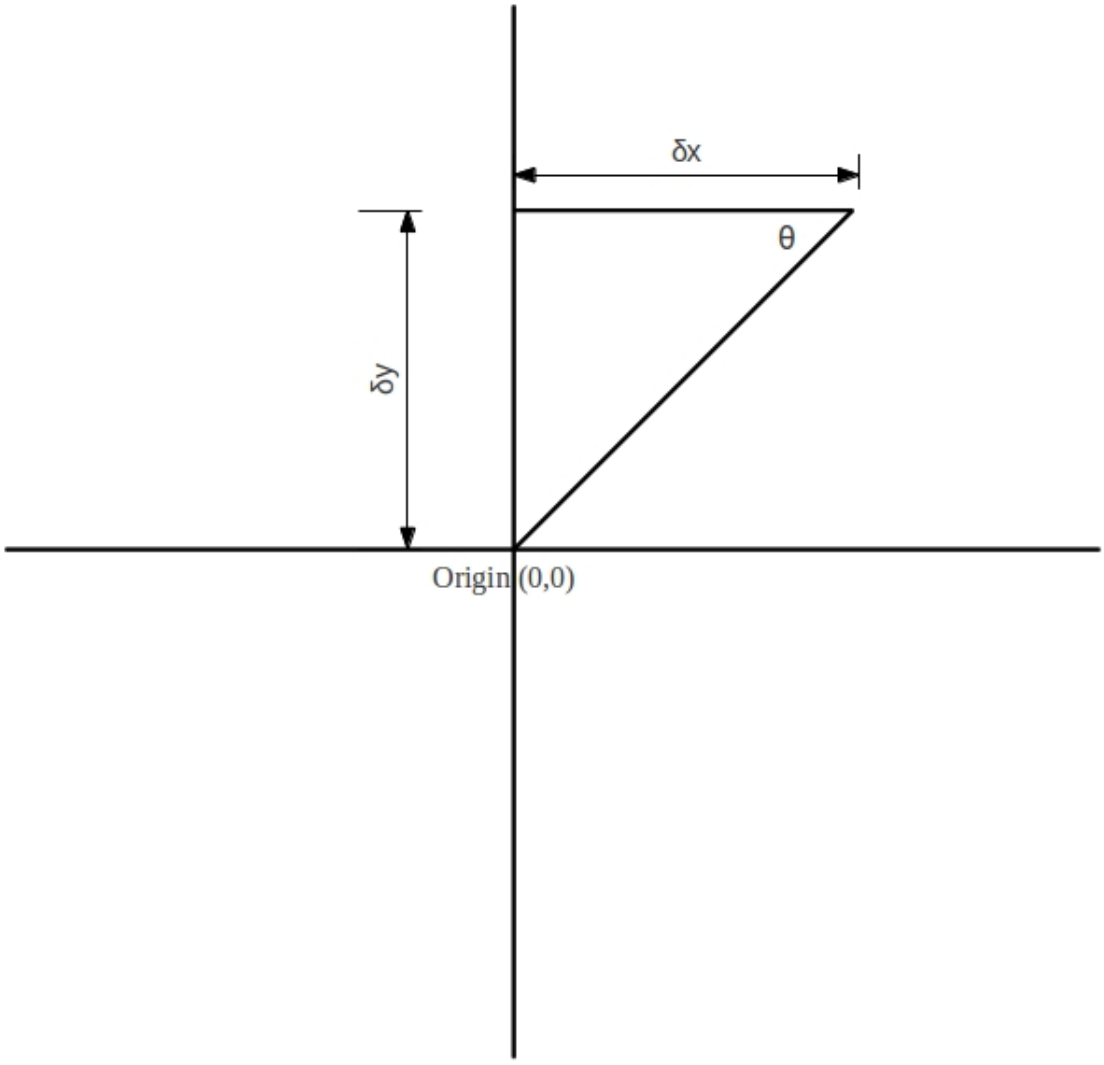
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

float angle;
int speak(char* path)
    {
    char path1[100]="festival --tts<./";
    strcat(path1,path);
    system(path1);
    return 1;
    }
```

Write a program named *move.c* that reads a series of two pixel values δ x and δ y from a data file *image.dat*. Assume that these pixel values are coming from a processed code of an image. Using these pixel values, you will have to compute the angle in degrees as shown in the diagram below. To compute the angle you can use a math library function *atan2*. Check the man pages of *atan2* for further usage in your program. Remember to include *math.h* file in the header and also link the math library using the linker options *-lm*. While you are linking two source files, use the link option of *-lm* to include the math library function *atan2* as well.

On computing the angle in degrees, your code should speak whether the next movement is *right*, *left* or *straight* using the following condition. The speak module is provided by speak function in your *speak.c* file.

```
-180 < angle < -90       Go right
0 > angle > -90          Go left
0 < angle < 90           Go right
90 < angle < 180         Go left
angle = 90               Go straight
```

Note that you have to create three text files: *left.txt*, *right.txt*, and *straight.txt* in the present directory *speak*. You need to put text in these *.txt* files so that the speak module reads out whatever you have written in *.txt* files. You need to call the function speak from your *move.c* as shown below based on the angular conditions.


δx
θ
δy
Origin (0,0)

```
speak("left.txt");
speak("right.txt");
speak("straight.txt");
```

Remember that your code should speak for series of data stored in *image.dat*. The *image.dat* will have information of pixels in the following format.

```
<x-value> <y-Value>
<x-value> <y-Value>
<x-value> <y-Value>
<x-value> <y-Value>
.................
```

The image.dat file can be downloaded from dropbox or LMS. You will have to use the file operations to open, read, and close the files. Ofcourse you will have to use conversion from string to float, if you are reading the contents as string in your program. The speak.c module contains *angle* global variable which is defined as float datatype.

Task 2: Series of data

Write a program named *rmove.c* satisfying the objectives given below. Assuming that we do not have *image.dat* file and we assume that our robot moves 10 pixel in x direction and 10 pixel in y direction for an instance. Finally our robot should move close to origin (0,0). In this case, the starting position is parsed from command line arguments as follows:

```
./rmove <startX> <startY>
```

In this case, print out different pixel values while tracing to origin (0, 0). Also use the speak module and indicate the robot movement before it moves.

Task3: Graph

Plot the series of tracing points in a graph. To plot this, your outputs should be redirected to *move.dat*. Use the following command:

```
./rmove <startX> <startY> > move.dat
```

We will restrict our *startX* and *startY* value from -100 to 100 for this graph. If you want to check giving higher value, you might want to change the *move.gnuplot* file. Download the *move.gnuplot* file from LMS or dropbox and run the following command.

```
gnuplot move.gnuplot
```

Your robot traces are visualized in a graph. If you are not able to get the graph, then your *rmove* program throws improper output.

Demonstration

Make sure your are in the *speak* directory. Make sure your code (*rmove.c* and *move.c*) are properly indented, neccesary comments, good selection of variable names, revision history and brief description of your code, before demonstrating it to your instructor.