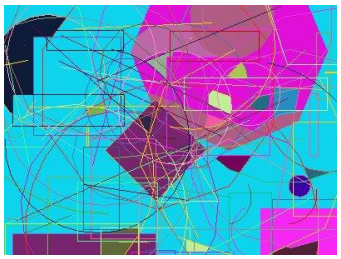
CENG 140 – C Programming

Spring 2018 - Homework 3

Image Description Compiler



Selim Temizer

Feedback : Between May 28th and June 1st, 2018
Due date : Tuesday, June 5th, 2018 (Submission through COW by 23:55)

In this homework assignment, you will be implementing a ***compiler/translator*** that reads textual image description commands from a text file in ***Temizer Image Description*** (***TID***) format, and produces another human-readable text file which is the described image in ***Plain PPM*** (portable pixel map) format.

Your compiler will support the following 10 textual image description commands:

- ***Image*** : Sets the size (width and height) and the PPM file name of the image
- ***SetColor*** : Sets the RGB color to be used by all subsequent commands
- ***Clear*** : Clears the canvas (with the last set RGB color)
- ***Point***, ***Line***, ***Arc***, ***Rectangle***, ***Polygon*** : Draws the named geometric entity on the canvas
- ***Fill*** : (Flood) fills with the last set RGB color
- ***Print*** : Prints a rasterized string on the canvas

A detailed sample file (***Sample.tid***) along with its output (***Sample.ppm***) in the homework bundle illustrate the TID format, the PPM format, and the syntax of all commands that you should support. You are provided with stub code which handles some of the requirements for you. You need to implement the parts marked with the comment **TODO** in ***ImageDescriptionCompiler.c*** file. The Windows (32-bit) and Linux (64-bit) executables of my sample implementation are also provided for you to try out various inputs and see the correct outputs. Note that for this homework assignment, you may assume that the input will not have any errors in it (for example the TID file will contain an ***Image*** command before any other commands, each command will have the correct number and type of arguments, etc.). There is also a Java application which generates random TID files (for fun or for testing), two of which describe the images shown at the top of this page (***some nice abstract art***). Make sure that the Java application and the TID compiler executable are in the same directory. To run the Java application, you may use the following command:

Prompt> `java –jar ImageGenerator.jar`

Bonus - Mona Lisa Challenge (10 points)

Hand build (or even better, employ artificial intelligence to automatically create) a TID file, which resembles the famous painting Mona Lisa as much as possible. The grading for this part will be completely subjective, and you will be competing with your classmates. The shortest TID file with the highest resemblance will get the highest grade.

What to submit? (Use ***only ASCII characters*** when naming all of your files and folders)

We are only asking for the following files:

- The first file is the implemented **ImageDescriptionCompiler.c** file.
- The second file is optional. If you would like to take The Mona Lisa Challenge, then prepare your **MonaLisa.tid** file by using the template in the homework bundle.

Zip the file(s) above (tar also works, but I prefer Windows zip format if possible), name the compressed file as <ID>_<FullNameSurname> (with the correct extension of .zip or .tar) and submit it through COW. For example:

e1234567_SelimTemizer.zip

There are a number of design decisions and opportunities for enhancements and creative extensions that are deliberately left open-ended in this homework specification. We have enough time until the deadline to discuss your suggestions and make further clarifications as necessary. There will be bonuses awarded for all types of extra effort. Late submissions will NOT be accepted, therefore, try to have at least a working baseline system by the deadline. Good luck.