

CS 354

Project 1 - turtle graphics

Due Feb 1, 5:00 pm

Overview

For this assignment you will be implementing an interpreter for a very simple turtle graphics language. See http://en.wikipedia.org/wiki/Turtle_graphics for information on turtle graphics. Our language will have the following commands:

1. `up` (pen up)
2. `down` (pen down)
3. `forward dist`
4. `left degrees`
5. `right degrees`
6. `loop num`
7. `color color` (0=black, 1=red, 2=green, 3=blue)
8. `origin` (place turtle at canvas center pointing up)

See the files `example1.txt` and `example2.txt` for example usage. The default color is black and default position/orientation is (0, 0) and facing up. Your program will read a command file and render the graphics.

The purposes of this assignment are threefold:

1. Get familiar with basic OpenGL commands
2. Get familiar with C++
3. Review and solidify trigonometry and vector algebra concepts

Instructions

1. Compile the skeleton code by typing `make` at the command-line
2. Run the skeleton code by typing `./turtle example1.txt`. This will output commands as they are read from the command file.
3. Look for all TODO comments in `main.cpp` and follow the instructions.
4. Add OpenGL commands to `main.cpp` to render the commands from the command file.
5. `cpplint.py` runs with every `make`, so your code should remain conformant to the coding guidelines during the development process.
6. Your final product should produce renderings exactly equal to what is shown in figure 1.
7. Write your own turtle graphics program commands in a file called `design.txt` and render it with your program. You should make this design interesting and creative. Your 10% creativity points will come from this. You are not permitted to expand the language – you can use only commands listed in the overview section above.
8. Take a screenshot of your `design.txt` program using the following command (on Linux): `import -window "Turtle graphics" design.png`.

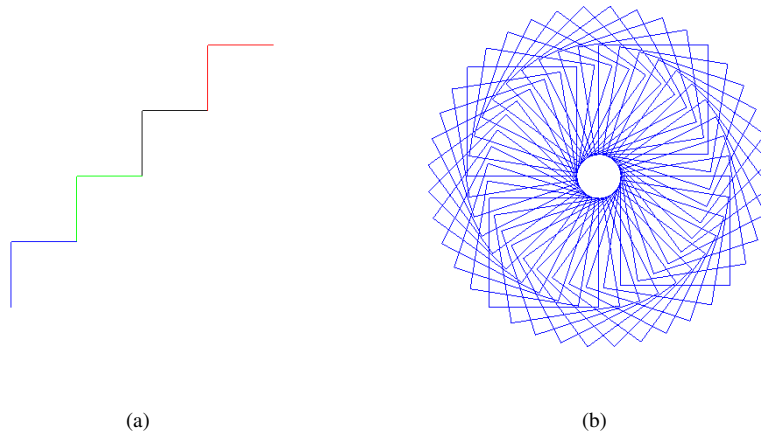


Figure 1: (a) Output from `./turtle example1.txt`. (b) Output from `./turtle example2.txt`.

9. Submit and verify your submission as follows:

```
turnin --submit vkeshari project1 .
mkdir test
cd test
turnin --verify vkeshari project1
make
./turtle example1.txt
```

Scoring

1. 80% - Correctly render example1.txt, example2.txt and other test files
2. 10% - Create an interesting and creative design.txt
3. 10% - Coding quality and style given by cpplint and visual inspection

Notes

1. You may use the standard C++ library (e.g. `vector`). You *may not* use any other third-party libraries without prior approval.
2. You can submit your work as frequently as you like – only the most recent submission will be retained. Suggestion: submit first thing to get familiar with how it works and submit occasionally during development. This way there won't be any surprises when you're up against the deadline.