

# Project 1

CS 135: Computer Science I  
Spring 2008

## Objectives

1. Demonstrate that you can integrate all that you have learned by developing a complete application: Speed-Sketch.

## 1 Speed-Sketch

Design, implement, and test a complete C/C++ application program that plays a game based on **etch-a-sketch**. The objective of the game is to trace a predefined pattern on the screen using etch-a-sketch type controls in the minimum number of cursor moves. The lower the number, the higher your score. You will use the `curses` library to implement your program.

Suppose your screen contains a square as shown in Figure 1, your job as player is to use the cursor controls to retrace the square. Hitting the up arrow key will start your cursor moving up,



Figure 1: Starting screen showing the pattern to be traced in blue and your cursor position in green.

covering the blue '-' dashes with your trace character '#'. Here's a screen shot of my playing the game and making it around the first corner without mishap (Figure 2). Note that the score is being displayed on the top left. Finally, Figure 4 shows a screen shot of my finishing tracing the pattern. My score is now 200.0 which is the highest score possible.

Now if I don't trace the pattern well and miss, I lose points. Figure ?? shows one such case, where I hit the wrong key and started off in the wrong direction, but I did it make it back on the right path a bit later. Notice my low score.

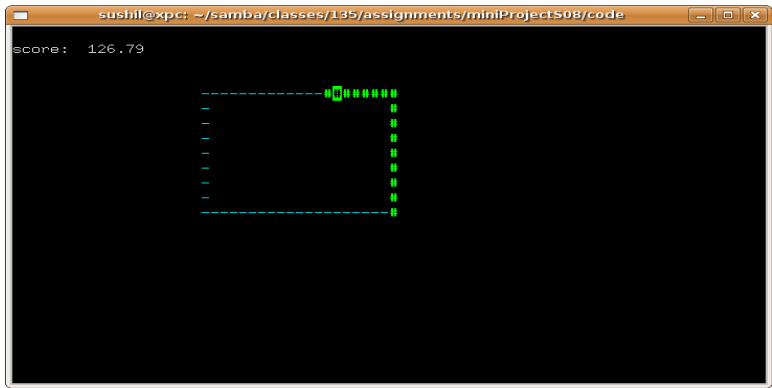


Figure 2: Making progress in tracing the pattern

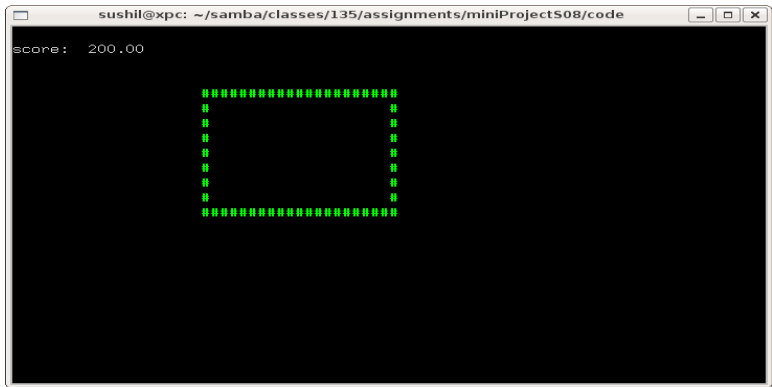


Figure 3: Finished, with a perfect game

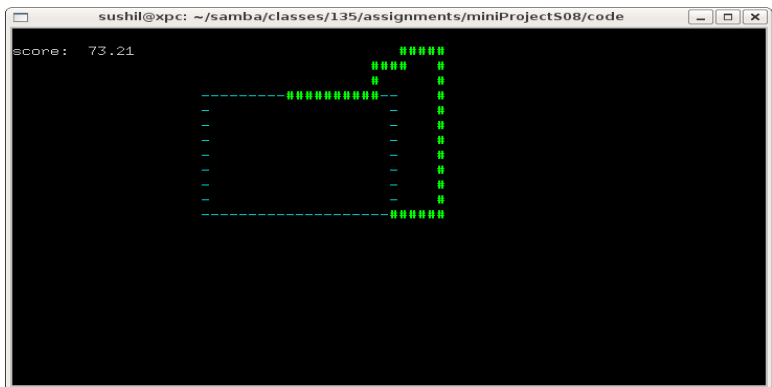


Figure 4: Missed at the start but got back after a bit.

## 1.1 Creating game maps

Let's call the square pattern that we used above, the square **map**. Your game playing application should have two modes.

1. Creating and storing a pattern (map making mode).
2. Playing the game by tracing a stored pattern (map tracing or playing mode).

So far, we have only described how to play the game assuming someone has already created a pattern to be traced. Now how do you create this pattern to be traced?

Here's the answer: Your game has two modes. When your game runs in map making mode, we can sketch out a pattern to be used for playing and store this pattern in a file. Later, when your program runs in map tracing mode, it will read this pattern from the file, display it on the screen, and ask you to play by tracing the program. Figure 5 shows the screen after I have started making a new pattern when running in "map making mode."



Figure 5: Creating a map

The complete map, comprising the x, y coordinates of all screen locations that have a '#' is stored in a file called `trace.txt`. You can get a copy of `trace.txt` from the class webpage for the map in Figure 6. When you play the game (map tracing mode), your program reads this file, displays it on the screen using blue '-' characters, and you play by tracing this map. Figure 6 shows what the game looks like during play on this newly created map.

Now how do you switch between the two modes?

You ask the user when you start up the game. Here's a screen shot (Figure 7) of the splash screen that comes up when you run the executable.

Finally, when you finish playing the game, the program should check whether you made the top ten list. The current top ten list is stored in a file. If your score is in the top ten, then you replace the name and score of the person with the lowest score with your name and score. At some point, you will have to ask the player for their three letter handle.

Here are the requirements for your program.

1. You must have a splash screen
2. The splash screen must allow choosing between map making and map playing modes.

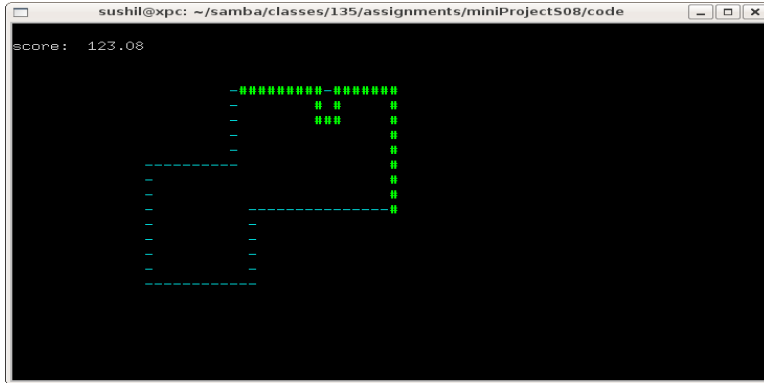


Figure 6: Playing on the new map

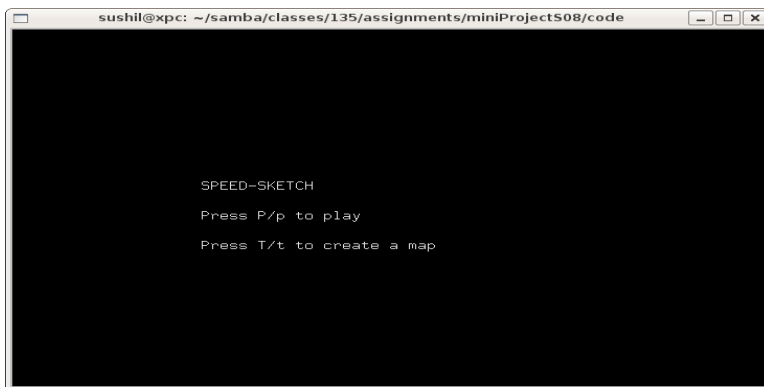


Figure 7: A very simple splash screen

3. You must use the UP, DOWN, LEFT, and RIGHT arrow keys for map making and map playing.
4. Hitting one of the arrow keys above causes the cursor to begin moving and tracing its motion with a **velocity** of one unit every  $\frac{1}{5}$  of a second. You can do this with a call to `halfdelay(2)` during curses initialization. This is you default game speed.
5. Pressing the space bar causes your cursor to stop moving.
6. In map tracing mode, you must read the pattern to be traced from the file `trace.txt`. Check for I/O errors.
7. In map making mode, you must record and then store the pattern you are making to a file named `trace.txt`. Check for I/O errors.
8. In map tracing mode, you must show the changing score on the screen.
9. You must use multiple colors to show the pattern to be traced and to show the current map being traced by the player.
10. Your score should be computed as follows:

$$\text{score} = \frac{t + h - m}{t} \times 100$$

where  $t$  is the number of positions in the pattern to be traced,  $h$  is the number of positions the player has traced in the pattern, and  $m$  is the number of positions that the player has moved the cursor to and that do not belong to the pattern.

11. The 'Q' and 'q' characters quit the game.
12. You must have a splash screen to start the game with the game's name and **game author names** on the screen.
13. You must have a file with the names and scores of the top ten players. You may assume that all names are three character names (handles).
14. If a player breaks into the top ten, replace the name and score of the worst of the top ten players with the new player's name and score.
15. Your cursor must never leave the boundary defined by your terminal (command) window.

## 2 Handing it in

There are two deliverables for this assignment/project.

1. **Due the week of March 30:** A design document for this project. A sample design document is on the class web page. Turn this typeset document in to your TA in lab in a binder/folder.
2. **Due the week of April 6:** An updated design document and a folder/binder with
  - (a) Cover sheet with

- i. Project title
  - ii. Section Number
  - iii. Your names (both names if you are in a group)
  - iv. Your email (both email addresses if you are in a group)
  - v. Your TA's name
- (b) Source code (.cpp) file(s) for the project on a CD/USB stick with your name and section number written on your CD/USB stick.
  - (c) Executable for the project on the CD/USB stick.
  - (d) Screen shots for all error-free test cases
    - Identify and annotate each screen shot. In other words, describe what each screen capture is showing.
  - (e) `trace.txt` and `score.txt` for all error free test cases.
  - (f) Testing on at least one case where you catch a file I/O error
  - (g) Hardcopy of your source code (printout)

Ask an instructor or TA if you have questions. **Start now!**