CIS*2500 Intermediate Programming (Winter 2015)

Assignment Three

CIS*2500: Assignment 3

Due March 13, 11:55 PM

The Big Picture

You will write a program that draws a single level for a "Rogue-like" computer game. The program will parse a line of input text from an input file (room.txt), use the parsed text to determine the shape of the room and its contents and then draw the room. The input text file will have 6 rooms and your program should draw all six rooms on a single map. Your program should also allow the user to move the hero around the rooms and pick up the treasure.

Starting

- Play Rogue for a half hour so that you get the idea of what you are building. The source code is in the useful files folder. It should compile on your pi and compiled fine on a mac. It has not been tested on windows compilers.
- The string tokenizer (strtok) will be extremely useful for this assignment. Google has lots of good info about how to use it.
- Set up your assignment folder (A3). Make sure you have all of the required folders (bin, src, include, assets, docs). The input file will go in the assets folder.
- Write a program to read a line from a file and parse the line into the dimensions, doors and items for a room.

Musts: do all of these to get a mark greater than zero

- Your solution must take the name of an input file as a command line parameter (using argv/ argc) and use that file to draw the level.
- Your solution must use the character set from Rogue 3.6.3
- Your solution must compile without errors or warnings
- Your solution must run and draw the outline of the first room in the test file as a minimum requirement.
- Your solution must have a README file and a makefile

The Basics: 80%

Expectations for your game

- Correctly draw all six rooms and all of the contents and doors (just draw the rooms all at once, don't progressively reveal them)
- Allow the hero to move around the room using the w,a,s,d keys.
- Permit the hero to pick up items by walking over them. Print the list of things the hero has picked up when the program exits.
- Prevent the hero from walking through walls or over monsters. (no need to implement any sort of combat with monsters)
- Allow the hero to walk onto doors and stairs. Doors should not be picked up and neither should stairs.
- When the hero walks onto a door, move the hero to the room closest to that door (unless you have created a hallway drawing algorithm). You can define what closest means and document it in your readme.

Expectations for your code

- Use the ncurses library
- Use structs effectively (for example room is a good struct, hero could be a struct, level could even be a struct)
- Use dynamically allocated memory and pointers in preference to statically allocated memory
- Use dynamically allocated arrays in preference to statically allocated arrays
- Use argv/argc to get the name of the file to read as input
- Your code must conform to the coding guidelines for CIS*2500
- Your code must compile with no error or warning messages using the -std=c99 and -Wall flags in gcc
- Have separate source files for groups of tasks that repeat, such as dealing with operations for the room, parsing, drawing, and reading the file. Have a separate file for for main

- Have a text document called testing.txt in your docs folder that describes how you tested your solution to ensure it was working properly.

Details about Input File

- The input file will contain six rooms. You should draw the rooms 3 across and 2 down. No room will be larger than 20 rows X 25 columns.
- Each line of the file will consist of a space-delimited line of text, no longer than 150 characters.
- Lines will not have a space at the beginning but may have trailing spaces and may have more than one space between elements.
- Lines will always have a newline at the end.
- The first element of the input string will ALWAYS be the room dimensions give as RowsXColumns (i.e. 10X12). The dimensions given are the dimensions of the **floor** of the room. The walls must surround the floor.
- The other possible elements can be in any order
- Door elements will begin with a lowercase d followed by a letter representing the wall that the door is in, followed by a position measured from north or west. e.g. de3 is a door in the east wall in position three from the north edge (where the corner is position 0).
- All other elements will begin with a lowercase letter followed by coordinates. i.e. the hero is represented by h6,8 which means that the hero is at position 6, 8 where the north west corner of the room is position 0,0 (rows,columns).
- letters that can occur in room descriptions will be limited to s(stairs), g (gold),m(magic), h(hero), p(potion), w(weapon) and M(monster). The hero will always start in the first room and will have a starting coordinate. One of the six rooms will have a stairwell.
- there will never be more than one door in a wall
- You need do only minimal error checking. The input file used for grading will not contain intentional errors. You should check that the items, monsters, and the hero dont start in the wall and that doors are actually in the wall. adjust items by the minimum number of spaces necessary if there are errors.
- 10X16 de4 dw9 ds8 g8,7 M3,4 h6,5 p2,2 is one possible example of a line in the text file

Enhancements: 20%

Your mark for the enhancements section will depend on the depth/breadth you choose. You can do several enhancements partially, or you can do one or two challenging ones really thoroughly and get a good mark either way. Document what you choose to do in your README. Your enhancements should show off your programming and problem solving.

- write an algorithm to connect rooms with hallways

- implement combat with monsters
- allow monsters to move randomly within the level
- write an algorithm to generate a new level (or levels) randomly. When the hero walks over the stairs, go to your new level.
- make an inventory tracking system that can be accessed during game play
- allow the player to use the items that have been picked up
- allow the player to save progress and reload from the save point
- implement different types of monsters that behave differently from one another. If you choose this enhancement you may use any upper case letter to denote the different monsters.

Deliverables: Things you must hand in

1. your A3 folder with all required subfolders, c files and header files, README, testing doc and Makefile
2. your test file with the data you used to test your program (in the assets folder)
3. tag your final submission with the tag a3final

Notes

1. Assignments that do not compile will be given a grade of zero
2. Assignments that compile with warnings will lose 15% of the mark
3. Program organization and coding style are not worth marks, but marks will be deducted from your overall score if you do not comply with the guidelines for this course.
4. All work in this course is to be done independently. Submissions will be examined electronically for similarity. Submissions will also be compared to submissions from previous offerings of this assignment.

Submission status

Submission status	This assignment does not require you to submit anything online
Grading status	Graded
Due date	Friday, 13 March 2015, 11:55 PM
Time remaining	The due date for this assignment has now passed
Grading criteria	

This list represents the minimum set of things a program must do before it can be graded.

- Compiles without errors or warnings using –Wall –std=c99
- Takes the name of the input file from command line
- When started does not crash
- Draws the outline of the first room (minimum)
- Uses correct character set
- README and Makefile are present

Any programs that do not meet the intent of the assignment (i.e. a rogue-like game using ncurses) cannot be graded by a TA and must be referred to Judi.

Code submitted must adhere to the guidelines for the class and must pass the cpplint check without any errors other than the naming convention used for include guards.

Marks cannot be earned for the following, but will be deducted:

- cpplint errors .5 mark for each error
- submission organization- 1-5 mark deduction depending on how badly organized your submission is.
- makefile -3 marks if missing -1 marks if doesn't work but is present
- readme -3 marks if missing
- hard coded path names -2 marks

Game Play <ul> <li>Hero moves with wasd keys. <li>Hero picks up items. <li> Hero cannot go through monsters or walls. <li> Inventory is displayed when game is exited . <li>Can move to the next room from the door <li> Stairs do not disappear when hero walks over them <li> Hero cannot go free-exploring in the dungeon (is confined to halls and rooms) </ul> **Maximum mark** 5
Layout

	Each room is drawn correctly and items, monsters, hero, stairs and doors are placed correctly. **Maximum mark** 5
	Coding Constructs Functions are used extensively and correctly. Data is structured appropriately using structs and arrays Dynamic memory is used frequently and correctly and is freed. Pointers are used properly wherever appropriate. There are no global variables. **Maximum mark** 8
	Testing There is a testing document that describes how the game was tested. The document should include samples of the test file used (or the entire sample) and any other information about how the program was tested as you wrote it. **Maximum mark** 2
	Enhancements Students: be prepared to show off what you've done. We'll be pretty flexible about what we call an enhancement, but you truly have to have learned something or done something extra. **Maximum mark** 5
Last modified	Monday, 16 February 2015, 11:20 AM
Submission comments	▸ Comments (0)

Feedback

Grade

This list represents the minimum set of things a program must do before it can be graded.

- Compiles without errors or warnings using –Wall –std=c99
- Takes the name of the input file from command line

- When started does not crash
- Draws the outline of the first room (minimum)
- Uses correct character set
- README and Makefile are present

Any programs that do not meet the intent of the assignment (i.e. a rogue-like game using ncurses) cannot be graded by a TA and must be referred to Judi.

Code submitted must adhere to the guidelines for the class and must pass the cpplint check without any errors other than the naming convention used for include guards.

Marks cannot be earned for the following, but will be deducted:

- cpplint errors .5 mark for each error
- submission organization- 1-5 mark deduction depending on how badly organized your submission is.
- makefile -3 marks if missing -1 marks if doesn't work but is present
- readme -3 marks if missing
- hard coded path names -2 marks

Game Play <ul> <li>Hero moves with wasd keys. <li>Hero picks up items. <li> Hero cannot go through monsters or walls. <li> Inventory is displayed when game is exited . <li>Can move to the next room from the door <li> Stairs do not disappear when hero walks over them <li> Hero cannot go free-exploring in the dungeon (is confined to halls and rooms) </ul> **Maximum mark** 5	- Hero can be teleported into wall	4.5 / 5
Layout Each room is drawn correctly and items, monsters, hero, stairs and doors are placed correctly. **Maximum mark** 5	- Dungeon not laid out correctly (should be 3x2) - Mixed up rows/columns - Not using	3 / 5

	correct character set	
Coding Constructs Functions are used extensively and correctly. Data is structured appropriately using structs and arrays Dynamic memory is used frequently and correctly and is freed. Pointers are used properly wherever appropriate. There are no global variables. **Maximum mark** 8	- Ineffective use of dynamic memory - Duplicated structure in header files	5 / 8
Testing There is a testing document that describes how the game was tested. The document should include samples of the test file used (or the entire sample) and any other information about how the program was tested as you wrote it. **Maximum mark** 2	- No testing document	0 / 2
Enhancements Students: be prepared to show off what you've done. We'll be pretty flexible about what we call an enhancement, but you truly have to have learned something or done something extra. **Maximum mark** 5		0 / 5

13 / 25

Graded on	Wednesday, 18 March 2015, 3:18 PM
Graded by	Mitchell Reynolds