Practicum 12 – Curses, Again

Assignment Details

Assigned: April 21st, 2014. Due: April 25th, 2014 at midnight.

A complete submission will include the modified code from the practicum.

Practicum

This practicum exists strictly to go over Programming Assignment 4 functionality.

We'll be enhancing (slightly) the code from Practicum 11 to ensure everyone has access to some code that meets the PA4 requirements.

PA4 Requirement One

Firstly, we need to display the dungeon level in the same spot every turn without adding any additional scrolling.

As it turns out… that's done for us in Practicum 11, which we're basing this code off of, so we're going to go with that. As instead of managing monster state, Practicum 11 just generates a new random level each time, it is going to look a bit silly, but hopefully still demonstrative.

PA4 Requirements Two and Three

We're going to need to get input directly from the keyboard, and then display some additional information beyond the dungeon level.

To make this a little more useful, we're going to use the ncurses setting for supporting the keypad. Right after the `initscr()` call in main, add:

```
keypad(stdscr, TRUE);
```

This tells ncurses to set the keypad option for the main screen we're working with to true, which turns on keypad parsing – which means we can read in the arrow keys a bit more easily.

Now, to demonstrate that we're going to do something with this keyboard input, we're going to need a few more variables to work with. We'll use a `stringstream` to make things a little easier – remember to `#include <sstream>` at the top of your file – keep track of the last command entered, and add some coördinates to draw a player character. Add a `stringstream` variable, an `int` variable to hold the key retrieved from the keyboard, a `string` variable to hold the command, and two `int`s to hold the coördinates. While we're at it, add another `int` to store the game turn as well. Remember that the `int`s have to be initialized to something useful; 1 will be fine for the turn, and some place on the dungeon works for the coördinates.

So, after we've read in the keyboard input – via the `getch()` function[1] – add a chain of if statements that test for the ncurses-defined keypad macros – `KEY_DOWN`, `KEY_UP`, `KEY_LEFT`, and `KEY_RIGHT`. For each one, set your `string` storing the command to an appropriate value (e.g., `"Down"` for `KEY_DOWN`), and then make the appropriate modification to a coordinate value.

In the final else block, you can set the command `string` to the character read in from the console:

```
sCommand = static_cast<char>(iCommand);
```

This sets it to a string of length 1 with the first (and only) character being the value passed.

Note that you still have to check for `'q'`![2]

Now that we've processed some input, we need to add the information display. Before we call `refresh()` in the main loop, let's:

- Clear out the `stringstream` variable, by setting it to a blank string; calling `str("")` on it does this.
- To the `stringstream`, insert the turn count and the last command entered.
- Use ncurses's `move()` and `addstr()` functions to move the cursor to a reasonable position for output (say, 20,0 – remember that ncurses likes the y coördinate to go first), and then write out the contents of the stringstream (calling `str().c_str()` on it will work; the `str()` method returns its value as a `string`, and `string`'s `c_str()` method returns that value as a C-style string for ncurses to work with).
- Don't forget to increment the turn counter!

Now that we've hit the main requirements for PA4, let's do one more thing:

After we're done, let's output something to the console. Note that using ncurses – and calling `initscr()` and `endwin()` – clears the terminal screen and then restores it to what it was before you started. `cout` doesn't do anything while we're between `initscr()` and `endwin()`, so we'll need to put any output we want the user to see *after* we call `endwin()`. To finish up this practicum, output the turn that the "game" ended on to the console after `endwin()` gets called, but before `main()` ends.

---

[1] Note that this function returns an `int`; with the keypad option turned out, the values for `KEY_LEFT` *et al* are greater than 255…

[2] However, for printing characters, the `int` returned will be equal to the appropriate `char`.