# COM S 229, Spring 2015
## Programming Project 1.04
### User Interface with Ncurses

So last week we added some characters, made them move around and smite one another. You added some code to drive your `@`. You can rip that code out, now. We're going to add a user interface that you can use to drive your `@` manually. If you like, you can leave the auto-drive code in there and add a command to turn it on and off at runtime.

Still working in C, link in the ncurses library and use it for unbuffered I/O and to draw only a portion of your dungeon roughly centered on the PC. Handle recentering however you like—every turn, when the PC gets too close to the edge, etc.—but now you only draw an $80 \times 24$ portion of the dungeon that contains the PC.

Since we are only drawing a portion of the dungeon, we want to be able to look around, so we're adding a *look mode* in addition to PC control, which we'll call *control mode*.

We're going to add stairs now, too. When the PC goes up or down stairs, a new dungeon is generated and populated with the PC and new monsters. An upward staircase is represented with $<$; a downward staircase with $>$. The PC uses stairs by entering the appropriate stair command while standing on the staircase. NPCs cannot use stairs, not even the smart ones. Stairs provide an important means of escape for PCs.

All commands are activated immediately upon key-press. There is never a need to hit enter. Any command which is not explicitly defined for a mode is a no-op for that mode. Implement the following commands:

| Key(s) | Action |
|---|---|
| `7` or `y` | In control mode, attempt to move PC one cell to the upper left. |
| `8` or `k` | In control mode, attempt to move PC one cell up. In look mode, attempt to move the view pane up by no more than one window height. |
| `9` or `u` | In control mode, attempt to move PC one cell to the upper right. |
| `6` or `l` | In control mode, attempt to move PC one cell to the right. In look mode, attempt to move the view pane to the right by no more than one window width. |
| `3` or `n` | In control mode, attempt to move PC one cell to the lower right. |
| `2` or `j` | In control mode, attempt to move PC one cell down. In look mode, attempt to move the view pane down by no more than one window height. |
| `1` or `b` | In control mode, attempt to move PC one cell to the lower left. |
| `4` or `h` | In control mode, attempt to move PC one cell to the left. In look mode, attempt to move the view pane to the left by no more than one window width. |
| $>$ | In control mode, attempt to go down stairs. |
| $<$ | In control mode, attempt to go up stairs. |
| space | In control mode, rest for a turn. NPCs still move. |
| `L` | In control mode, enter look mode. |
| escape | In look mode, enter control mode. Return view to PC's position. |
| `S` | Save to disk and exit game. |

With these changes, we no longer need the delay; the game pauses automatically for input. And ncurses should handle the redrawing, so we're no longer spewing the entire dungeon to the terminal each turn. Things look much nicer.

Did you notice that save command above? Not very useful with the way dungeons are currently saved, given that we've now got more functionality, so we'll need to update our save file format and save semantics. From now on, the game automatically loads the save file if it exists. If the game exits because the player entered the save command, the save file is written. If the game exits because the PC dies, the game ensures that the save file is deleted before exiting. Note that the file version marker changes! Here's the new file format:

| Bytes | Values |
|---|---|
| 0–5 | A semantic file-type marker with the value `RLG229` |
| 6–9 | An unsigned 32-bit integer file version marker with the value 1 |
| 10–13 | An unsigned 32-bit integer size of the rest of the file (total size of the file minus 14) |
| 14–17 | An unsigned 32-bit integer offset of the user block, or of EOF if you are not using the user block. |
| 18–61457 | The row-major dungeon matrix from top to bottom, with four bytes for each cell. In each cell, the first byte is non-zero if the cell represents open space, zero otherwise; the second byte is non-zero if the cell is part of a room, zero otherwise; the third byte is non-zero if the cell is part of a corridor, zero otherwise; and the fourth byte is the unsigned, 8-bit integer hardness value of the cell, where open space has a hardness of 0 and immutable material has a hardness of 255. |
| 61458–61459 | An unsigned 16-bit integer containing the number of rooms in the dungeon |
| 61460–$x$ | The positions of all of the rooms in the dungeon, given with 4 unsigned 8-bit integers each. The first byte is the $x$ position of the upper left corner of the room; the second byte is the $y$ position of the upper left corner of the room; the third byte is the width of the room; and the fourth byte is the height of the room. |
| $x+1$–$x+2$ | The position of the PC. Each byte is an unsigned 8-bit integer. The first byte is the $x$ position of the PC. The second byte is the $y$ position of the PC. |
| $x+3$–$x+6$ | The four byte game turn; corresponds with the PC's next turn. |
| $x+7$–$x+8$ | Unsigned 16-bit integer containing the number of NPCs in the dungeon. |
| $x+9$–$y$ | A list of the NPCs in the dungeon. Each NPC contains a `char` containing its character graphic; two unsigned 8-bit integers for NPC position, $x$ followed by $y$; an 8-bit boolean value for the monsters intelligence (zero is not smart, non-zero is smart); an 8-bit boolean value for the monsters telepathic ability (zero is not telepathic, non-zero is telepathic); two bytes for the monster's last known position of the PC, defined as all other positions above (use 255,255 if the NPC has never detected the PC)); 17 bytes that you may use (or not) as you choose, for a total of 24 bytes per NPC. |
| $y$–end | The *user block*. You may use this however you like, including not using it. If you've implemented non-required extensions to the game, and you want to save them, write them here. |