CS 246 Winter 2016 Project - ChamberCrawler3000

Peter Buhr and Rob Schluntz

Due Date: Monday, April 4, 2016 at 23:55

This project is intended to be doable by two people in two weeks. Because the breadth of students' in this course is quite wide, exactly what constitutes two weeks' worth of work for two students is difficult to nail down. Some groups will finish quickly; others won't finish at all. We will attempt to grade this assignment in a way that addresses both ends of the spectrum. You should be able to pass the assignment with only a modest portion of your program working. Then, if you want a higher mark, it will take more than a proportionally higher effort to achieve, the higher you go. A perfect score will require a complete implementation.

Above all, make sure the basics work well! Grading for the project involves automated marking. The automated marking will test many small features by reading in a map and doing a few simple things. This is necessary for the bulk of your correctness grade. The rest of the grade is random generation and how your program behaves in longer playthroughs. Remember that you can test CC3k against expected output using the harness script from assignment 1!

1 The Game of ChamberCrawler3000

In this project, you will work to produce the video game ChamberCrawler3000 (CC3k), which is a simplified rogue-like (a genre of video game based upon the game Rogue[1] - `http://en.wikipedia.org/wiki/Rogue_(video_game)`). Prior experience with these types of games is not required, and should not discourage you from choosing this project.

A game of CC3k takes place on a board 79 columns wide and 25 rows high. The goal of the game is to descend 5 floors. Along the way, a character controlled by the player slays enemies and picks up treasure. A dungeon consists of those 5 floors with exactly the same layout, but different contents. While stair, item, player, and enemy location will vary, each floor will always consist of the same chamber layout exactly the way provided in this document.

CC3k differs from other rogue-likes in a significant way: it does not update the terminal/window in real-time but rather redraws all elements every turn (i.e. after every command). Many early rogue-likes were programmed in a similar fashion to CC3k.

Note that these specifications cannot hope to list every detail, especially in terms of displayed output, and the reference implementation should be carefully consulted to see how the game should behave in various contexts.

1.1 Some Definitions

It is understandable that this type of game may be new to readers of this document[2]. Accordingly, some definitions are provided here to aid in the reading of this document.

Definition 1: A **character** is a person/animal/thing in the game of CC3k. This can be either the player character (PC), who is controlled by the player of the game, or an enemy.

[1] Which is itself based on Dungeons and Dragons.
[2] In the internet vernacular, y'all be n00bz.

Definition 2: An **item** is something the player character can pick up or use. (Additionally, goblins may use potions.) In CC3k, this is either `gold` or `potions`. Potions offer potentially positive and negative effects to the ~~player character~~user, which may be a player or goblin.

Definition 3: A **chamber** is an individual room in the game of CC3k. Chambers are connected by **passages**.

Definition 4: A **floor** in CC3k is a predefined configuration of 5 chambers with connecting passageways. Figure 1 depicts an empty floor. Note that the configuration is the same for every floor in a game of CC3k.

Definition 5: **Health Points (HP)** is the representation of a character's health (both enemies and the player character). When a character's HP reaches 0, they are slain. For an enemy this means that they are removed from the floor and a tidy sum of gold is given to the player character. When the player character has 0 HP then the current game ends.

Definition 6: **Attack (Atk)** is the representation of a character's strength. This is how hard enemies can hit the player, and vice versa.

Definition 7: **Defense (Def)** is a percentage value deducted from the Attack value. Defence can never be above 100.

Definition 8: A **cell** is either a wall, floor tile, doorway, or passage.

Definition 9: Something is **spawned** means that the particular something (an enemy, gold, etc) should be generated and placed on the board.

Definition 10: A **1 block radius** denotes the 8 adjacent cells to the character or item.

2 System Components

The major components of the system are as follows:

2.1 Player Character

The player is denoted by the '@' symbol and is a valiant human adventurer of one of the following classes:

- The Knight (100 HP, 50 Atk, 50 Def), an all-around versatile fighter who lacks in special abilities and instead relies solely on her stats to defeat enemies.
- The Wizard (60 HP, 25 Atk, 0 Def), who is weak but has a powerful long-range attack, able to hit the first unit in a straight line in front of him. For example, see the following diagram, where o is the square attacked and x denotes any square eligible to be hit:

```
@oxxxxxxxg...D|

@...
.o..
..x.
...g
```

  In the first example, the Wizard hits the goblin to the east but not the dragon behind it. In the second example, he hits the goblin to the southeast.

- The Samurai (80 HP, 50 Atk, 15 Def), whose grace allows him to go unnoticed by the denizens of the dungeon until he attacks an enemy. All enemies behave similarly to the merchant: once the first enemy of a particular type is attacked, all enemies of that type become hostile.

The player character gains 5HP per turn. The player character can never have more HP than their starting HP at any time.

2.2 Enemies

Enemies are the mortal foes of our illustrious player character. In a traditional rogue-like, the enemy character would have some degree of artificial intelligence. However, for simplicity in CC3k, most enemies move randomly within the chamber that they spawned in. Dragons are stationary and always guard a treasure hoard. The following enemies can spawn:

Enemy	Display	HP	Atk	Def	Item Dropped	Special Abilities
Grid Bug	X	50	20	10	Random Potion	Cannot move or attack diagonally
Goblin	g	75	30	20	Gold Pile	Can drink potions
Merchant	M	100	75	5	Gold Pile	Ignores player by default
Orc	O	120	30	30	Gold Pile	None
Dragon	D	150	50	10	None	Guards a Dragon Hoard

Goblins do not actively seek out potions, but will preferentially drink potions they are beside in the same way that enemies preferentially attack the player instead of moving. A Goblin still prefers attacking to drinking potions.

By default, merchants are neutral to all parties[3]. However, merchants can be attacked and slain by the player character. Attacking or slaying a Merchant will cause every Merchant from that point forward to become hostile to the player character and from then on behave like other enemies. For the Samurai, all enemies begin neutral in a similar manner. [4].

Dragons always spawn in a one block radius of its dragon hoard (see Treasure). That is, if a dragon hoard is spawned then a dragon is spawned. Dragons carry no additional treasure. When an enemy dies, they drop the item they are carrying (if any) on the tile they were standing on.

Non-dragon enemies move to a random unoccupied adjacent square (see Section 3 for floor tile description). An enemy can never leave the room it spawned in. To guarantee consistent behaviour, we require that enemies move in a left-to-right, top-to-bottom fashion. In particular, ensure that each enemy is moved in the order they appear at the beginning of the turn (i.e. no enemy should move twice). For consistent random generation, number the unoccupied tiles surrounding an enemy clockwise starting from 0 at the top left. For example, see the following diagram, where x is the enemy trying to move, ? is an occupied tile, and the unoccupied tiles are numbered according to the described scheme.

```
012
3x4
567

?0?
1x2
3?4
```

In the first example, the enemy is surrounded by unoccupied tiles. In the second example, the enemy has occupied tiles to the northwest, northeast, and south. Remember, there may not always be an unoccupied floor tile! *In this case, no square is selected and the enemy does not move.*

[3]They're capitalists after all

[4]So Samurais can just ignore enemies entirely, but where's the fun in that?

2.3 Items

2.3.1 Potions

In the game of CC3k, there is only one type of usable item: a potion. Potions are of two types: positive and negative. Potions can provide the player character with positive and negative bonuses as outlined below. Regardless of the potion itself, all potions are denoted on the map with a ! (exclamation mark). Unlike most rogue-likes, there is no way to store potions for later. Potions are used from the ground with the 'u' command.

Positive Potions:

- `Restore health (RH)`: increase HP by 30
- `Boost Atk (BA)`: increase ATK by 10
- `Boost Def (BD)`: increase Def by 10

Negative Potions:

- `Poison health (PH)`: decrease HP by 15
- `Wound Atk (WA)`: decrease Atk by 5
- `Wound Def (WD)`: decrease Def by 5

The effects of RH and PH are permanent while the effects of all other potions are limited to the floor they are used on. For example, using a BA potion will only boost the player character's Atk until the beginning of the next floor.

No stat may ever be below 0, and HP may not ever exceed the starting HP of the character. If HP is ever 0, the player or enemy in question dies immediately. Additionally, defence may never exceed 100. You may implement stat limits (0 or 100, as applicable) in one of two ways: Either, you can never store a value outside the range, or you can treat values outside the range as 0 or 100 as applicable.

2.3.2 Treasure

Treasure in CC3k consists only of gold. Gold comes in two forms but is always denoted '$' on the map. Treasure is accumulated with the 'u' command OR by simply walking onto it.

- Gold Pile: This gives the player +10 gold when picked up.
- Dragon Hoard: This gives the player +50 gold when picked up, but cannot be picked up until the dragon is slain.

2.4 Floors

Levels are generated to consist of the 5 chambers connected in the manner outlined in Figure 1. Random generation of the actual floor layout is an interesting problem, but NOT part of this assignment.

Note that in all cases, every room is equally likely to be chosen for random generation, so the smallest and largest room will on average have the same number of things in them. Whenever a room must be chosen, number them from left-to-right, top-to-bottom, such that 0 maps to the top left room, 1 to the top right room, 2 to the small middle room, 3 to the bottom left room, and 4 to the bottom right room.

Generation should be done in the following order:

- The player character spawns in a random room.
- The stairs spawn in a random room which does not contain the player. If reading from a file you need not enforce this.
- 10 potions are spawned, randomly chosen from the six potion types. For consistent random generation, number the six potion types in the same order as listed in Section 2.3.1

- 10 gold piles are spawned, with a 7/8ths chance for a Gold Pile, and 1/8th chance for a Dragon Hoard. A Dragon must be spawned to accompany each Dragon Hoard.
- 20 enemies, not counting dragons, are spawned with the probability 1/3 for a Grid Bug, 1/3 for a Goblin, 1/6th for an Orc, and 1/6th for a Merchant. For consistent random generation, select with a single PRNG call and number the enemy types in this order.

Multiple objects may never occupy the same position. While generating objects, if the chosen random position is already occupied, try to place that objects again by rerolling both the room and the position within the room only. In the case where a Dragon Hoard has successfully been placed, but placing the Dragon fails, reroll the position of the Dragon Hoard and try again. If the player attempts to move on to the stairway, the next level is immediately generated and displayed according to the above rules. If the player attempts to move on to the stairway on floor 5, they immediately win the game. Enemies may not stand over items, the player character, stairways, or in passages.

For full marks, your random generation of floors should abide by the following pseudocode. See above for correct constant values / possibilities.

```
spawnPlayer()
spawnStairs()
for i = 0; i < NUM_POTIONS; i++
    random number 0 to 5 inclusive
    generate potion from number according to order listed in potions section
for i = 0; i < NUM_GOLD; i++
    random number 0 to 7 inclusive
    if was 7 generate dragon hoard, else generate gold pile
for i = 0; i < NUM_ENEMIES; i++
    random number 0 to 5 inclusive
    generate the enemy according to [Grid Bug, Grid Bug, Goblin, Goblin, Orc, Merchant]
```

When generating objects, after having decided what type of object, use the following scheme to generate the location of a non-stairs object:

```
while true
  chamber = random 0 to 4, inclusive
  square = random square in chamber, numbering left to right, top to bottom
  if square valid
       use and break
```

When generating stairs, use the following scheme:

```
while true
  chamber = random 0 to 3, inclusive
  if chamber >= player chamber
      chamber += 1
  square = random square in chamber, numbering left to right, top to bottom
  if square valid
       use and break
```

2.5 Combat

By default, all enemies except for Merchants are hostile to the player character (except Samurai, for which all enemies act like Merchant). Enemies will strike the player whenever they are directly adjacent to the enemy instead of moving. Enemies will not pursue the player character: If the player moves away, they go back to moving randomly.

Combat is resolved as follows: Enemies will auto-attack players given the previously specified criteria. The player character can (but is not required to) attack any adjacent tile.

```
|-----------------------------------------------------------------------------|
|                                                                             |
| |--------------------------|        |-----------------------|               |
| |..........................|        |.......................|               |
| |..........................+########+.......................|-------|       |
| |..........................|   #    |...............................|--|    |
| |..........................|   #    |..................................|--| |
| |----------+---------------|   #    |----+----------------|...............| |
|            #                 #############                |...............| |
|            #                 #     |-----+------|          |...............| |
|            #                 #     |............|          |...............| |
|            ###################     |............|   ######+...............| |
|            #                 #     |............|   #     |...............| |
|            #                 #     |-----+------|   #     |--------+------| |
|  |---------+-----------|     #           #          #              #        |
|  |.....................|     #           #          #         |----+------| |
|  |.....................|     ########################         |...........| |
|  |.....................|     #           #                    |...........| |
|  |.....................|     #    |------+--------------------|...........| |
|  |.....................|     #    |.......................................| |
|  |.....................+##########+.......................................| |
|  |.....................|          |.......................................| |
|  |---------------------|          |---------------------------------------| |
|                                                                             |
|-----------------------------------------------------------------------------|
       Class: Knight      GP: 0             Floor 1
       HP: 100/100        Atk: 50           Def: 50%     Turn: 1
```

Figure 1: Empty board showing all passages between chambers.

Damage is calculated as follows: $Damage(Defender) = ceiling(Atk(Attacker) * (100.0 - Def(Defender))/100.0)$, where Attacker specifies the attacking character (enemy or PC) and defender specifies the character being attacked. Note that a character can play the role of attacker and defender in the same turn, since a player and enemy can both attack each other.

3 Display

The display of CC3k is relatively simple, Figure 1 depicts an empty board. Walls are denoted by '|' and '–', doorways by '+', and passages by '#'. Floor tiles that can be walked upon are denoted by '.'. Chambers are denoted by the smaller polygons inside the larger rectangle. The entire map is visible at all times. Figures 2, 3, 4, depict various board states. Note that Figure 1 represents a completely empty game board and is meant to act as a reference of what the game board would look like before any generation occurs.

4 Command Interpreter

Initially, the game will demand the player enter one of the specified classes (see sample implementation for specific prompt). Play will then continue in the specified way until the player restarts, reaches the end of floor 5, the PC dies, or the player quits. If the player reaches the end of the game or their character is slain, the game should give them the option of playing again or quitting.

The following commands can be supplied to your command interpreter:

- **no,so,ea,we,ne,nw,se,sw**: moves the player character one block in the appropriate cardinal direction if square is empty. Picks up treasure if square has treasure.

```
|-----------------------------------------------------------------------------|
|                                                                             |
| |--------------------------|        |-----------------------|               |
| |.......@..........X.......|        |..g.......g..........$.|               |
| |....g...............g.....+########+.......................|-------|       |
| |............X.............|   #    |..........X..........g.........|--|    |
| |.......O.............M....|   #    |..X...............................|--| |
| |----------+---------------|   #    |----+----------------|...............| |
|            #                 #############                |........$......| |
|            #                 #     |-----+------|         |..g......D.....| |
|            #                 #     |.....g...X..|         |...............| |
|            ###################     |.....>......|   ######+....g..........| |
|            #                 #     |.....g...X..|   #     |...............| |
|            #                 #     |-----+------|   #     |--------+------| |
|  |---------+-----------|     #           #          #               #        |
|  |.....................|     #           #          #         |----+------| |
|  |....g........g.......|     ########################         |...........| |
|  |..........$..........|     #           #                    |...$..$....| |
|  |.....M...............|     #    |------+--------------------|...........| |
|  |............M........|     #    |.......................................| |
|  |................!....+##########+.....X......O....g..............M......| |
|  |.....$...............|          |..........................M............| |
|  |---------------------|          |---------------------------------------| |
|                                                                             |
|-----------------------------------------------------------------------------|
        Class: Knight      GP: 0              Floor 1
        HP: 100/100        Atk: 50            Def: 50%     Turn: 1

What will you do?
```

Figure 2: A board showing a (potentially) generated board.

```
|-----------------------------------------------------------------------------|
|                                                                             |
| |--------------------------|        |-----------------------|               |
| |.......@..........X.......|        |..g.......g..........$.|               |
| |.......O............X.....+########+.......................|-------|       |
| |............X.............|   #    |..........O..........M.........|--|    |
| |.......O.............M....|   #    |..!...............................|--| |
| |----------+---------------|   #    |----+----------------|...............| |
|            #                 #############                |........$......| |
|            #                 #     |-----+------|          |..X......D.....| |
|            #                 #     |.....X...$..|          |...............| |
|            ###################     |.....>......|   ######+....!..........| |
|            #                 #     |.....O...!..|   #      |...............| |
|            #                 #     |-----+------|   #      |--------+------| |
|  |---------+-----------|     #           #          #               #        |
|  |.....................|     #           #          #          |----+------| |
|  |....O........g.......|     ########################          |...........| |
|  |..........g..........|     #           #                     |...$..$....| |
|  |.....g...............|     #    |------+--------------------|...........| |
|  |............g........|     #    |.......................................| |
|  |................!....+##########+.....g......!....O..............M......| |
|  |...M.X...............|          |..........................X............| |
|  |---------------------|          |---------------------------------------| |
|                                                                             |
|-----------------------------------------------------------------------------|
        Class: Knight      GP: 0              Floor 1
        HP: 100/100        Atk: 50            Def: 50%     Turn: 1

What will you do?
a ea
You attack the Orc with your Sword of Segfault for 35 damage!
The evil Orc strikes you for 15 damage!
```

Figure 3: A board showing combat between a Knight and Orc

```
|-----------------------------------------------------------------------------|
|                                                                             |
| |--------------------------|        |-----------------------|               |
| |.>.........!..........O...|        |.......................|               |
| |..............$.....!..$..+########+.............X.........|-------|       |
| |......X...............g...|   #    |.......................$.X.....|--|    |
| |...X......................|   #    |..................................|--| |
| |----------+---------------|   #    |----+----------------|......$........| |
|            #                 #############                |........!......| |
|            #                 #     |-----+------|         |..............X| |
|            #                 #     |!..O.$....@!|         |...............| |
|            ###################     |....!.$.O...|   ######+.....g.........| |
|            #                 #     |..X.........|   #     |......X........| |
|            #                 #     |-----+------|   #     |--------+------| |
|  |---------+-----------|     #           #          #              #        |
|  |.......!....g........|     #           #          #         |----+------| |
|  |..........O..........|     ########################         |...........| |
|  |O...............$....|     #           #                    |..$........| |
|  |...........O.........|     #    |------+--------------------|...........| |
|  |......X..............|     #    |....$.................................X| |
|  |....................!+##########+...g.....................!.............| |
|  |.......X.............|          |.....$.!...............................| |
|  |---------------------|          |---------------------------------------| |
|                                                                             |
|-----------------------------------------------------------------------------|
        Class: Knight      GP: 0             Floor 1
        HP: 100/100        Atk: 50           Def: 50%     Turn: 1

What will you do?
u we
There is nothing there to use!
u ea
You chug the Wound Attack potion.
```

Figure 4: A board showing failed and successful potion usage. Failed usage does not reprint the board.

- **u <direction>**: uses the potion or pickup the treasure indicated by the direction (e.g. no, so, ea).
- **a <direction>**: attacks the enemy in the specified direction, if the attack would hit an enemy. For the Samurai and Knight, the enemy must be 1 square away. For the Wizard, there must be an enemy somewhere along that direction.
- **k, w, s**: specifies the class the player wishes to be when starting a game.
- **r**: restarts the game. All stats, inventory, and gold are reset. A new class should be selected.
- **q**: allows the player to admit defeat and exit the game.

In addition, you must support two additional commands for ease of testing (and you may find them useful):

- **stopwander**: causes all enemies to stop moving permanently.
- **stopdeath**: The player cannot die, even if they are at 0hp. The game only ends when the player wins, restarts, or quits.

Note that the board should be redrawn as appropriate every time a command is entered. The **stopwander** and **stopdeath** cheat codes apply until the current game session ends.

5 Ending the Game and Scoring

A game session ends in one of the following ways: the player character reaches the stairs on floor 5, the player character's health reaches 0, the player restarts the game or quits.

A player's score is only generated in the first two of the above cases. Score is determined by the amount of gold they have collected in the current game session.

6 Command Line options

Your program must accept two optional command line arguments: a number and a file name, which may appear in any order (you may assume that the file name is not a number). The number specifies a seed for your random number generator to use so that it is possible to generate repeatable output. If no value for the seed is specified, the random number generator is initialized with an arbitrary seed value. To generate random numbers, you must use the **PRNG class** from `prng.h`, which provides an encapsulated random number generating algorithm. The file name specifies a file to read floor layouts from. Floor layouts will appear exactly how they would be printed, with the exception that potions and gold are represented by numbers as follows: 0 - RH, 1 - BA, 2 - BD, 3 - PH, 4 - WA, 5 - WD, 6 - Gold Pile, 7 - Dragon Hoard. You do not need any error checking whatsoever for reading in the input file. Assume a valid 5-board input file.

7 When all else fails...

This is a relatively complex project with many components and a fair amount of random generation. Accordingly, if you find yourself running out of time or having trouble here's our suggestion for priorities:

- Reading in (from a file) and printing a floor.
- Implement the Knight and the Orc enemy (as both have no special ability) with movement and combat.
- Make the Orc enemy drop the Gold Pile, and have the Gold Pile give +10 gold when picked up.
- Introduce additional enemies, classes and items, in loosely that order.
- Get stats and interaction working (combat, item use, etc) for these.

- Random generation of items/enemies on floors. You can get full random generation marks even if every enemy acts like an Orc and every item acts like a Gold Pile.

Accomplishing the first three points should reward you with at least 50% (assuming you have used object-oriented principles to get that far). That is, you will get a higher mark for submitting something that implements a subset of the requirements well than submitting something that attempts to implement all the requirements without any of them working quite correctly.

8 Submission Instructions

This assignment should be done by a team of two people because of its size. Both people receive the same grade (no exceptions). **Only one member of a team submits the assignment.** The instructor and/or instructional-coordinators do not arbitrate team disputes; team members must handle any and all problems.

Please follow these guidelines carefully. **Review the Assignment Guidelines and C++ Coding Guidelines *before* starting each assignment. Each text file, i.e., `*.*txt` file, must be ASCII text and not exceed 500 lines in length, where a line is a maximum of 120 characters.**

Name your submitted files as follows:

1. `*.h,cc,C,cpp` – code for this assignment. The program must be divided into separate compilation units, i.e., `.h` and `*.cc,C,cpp` files. **Program documentation must be present in your submitted code. Output for this question is checked via a marking program, so it must match exactly with the given program.**

2. `cc3k.testtxt` – test documentation for the assignment, which includes the input and output of your tests. Poor documentation of how and/or what is tested can results in a loss of all marks allocated to testing.

3. `cc3k.pdf` – UML class diagram for your program. Do not get fancy and use complex tools to draw this picture. Any simple drawing tool should be able to generate the boxes and text, and generate `.pdf` output. **The picture must fit on a standard 8.5x11 page and may appear in either portrait or landscape format.**

4. `Makefile` – construct a makefile with target `cc3k`, which creates the executable file `cc3k` from source code in the makefile directory when the command This `Makefile` must be submitted with your assignment and is used to build the program, so it must be correct.

Follow these guidelines. Your grade depends on it!