Concordia University
Department of Computer Science
and Software Engineering

Advanced Programming Practices
SOEN 6441 --- Winter 2017

Project Description

Deadline:	Intermediate delivery 1:	March 2nd, 2017
	Intermediate delivery 2:	March 23rd, 2017
	Final delivery:	April 13th, 2017
Evaluation:	Intermediate delivery 1:	15% of final mark
	Intermediate delivery 2:	15% of final mark
	Final delivery:	20% of final mark
Late submission:	not accepted	
Teams:	the project is a team assignment	

General description

The project is to be undertaken teams of 4-5 members and consists of the building of a challengingly large Java program. The completion of the project is divided into three separate components: (1) the *First and Second Intermediate Project Delivery* are intermediate operational build of the software, effectively demonstrating the full implementation of some important software features; (2) the *Final Project Delivery* is the demonstration of the finalized version of your software. During the final project delivery, you also have to demonstrate that your code includes many of the Java features presented in the lectures, and that you effectively use the tools presented in the lectures in your project. All project deliveries are to be undertaken in the laboratory where the team presents the implemented features to the instructor following a pre-circulated grading sheet. The individual assignments will also be related to the project, but graded individually and separately from the project.

It is important to realize that the project description is purposely incomplete, and that it is one of your duties in this project to: 1) elicit and formulate all the missing details before you start the implementation, 2) limit the scope of the project according to the time that is available, 3) determine what design decisions will be made, as well as 4) what tools will be used for the implementation. These activities require some investigations and discussions that are important aspects of software development and this project.

Problem statement

The project is about writing a Java program that allows the user to play a simplistic version of a *Dungeons and Dragons* role-playing game. The game must absolutely follow the d20 gaming rules (see the d20 rules reference below). Two particularly good (and free) example of online games following these rules are *Dungeons and Dragons Online* and *Neverwinter* (see the references below).

It should be fairly obvious that it would be a Herculean feat to implement the complete set of d20 rules within a single semester course. One of your early goals is to define what subset of the d20 rules you can implement given your limited available resources, and that makes a complete and coherent set of game rules and features. Below are given some guidelines as to what is expected as mandatory features. Additionally, as this course does not have any computer graphics prerequisites, only a simplistic window-and-mouse user interface is expected.

The following describes the mandatory goals of the project in terms of game features. The main idea is that you should have a character moving on a map. That character can interact with various game elements placed on the map: enter combat (either as being the attacker or by being attacked) with non-player characters (NPCs i.e. friendly or hostile characters or monsters) on the map, use a door to go to another map, or open a chest. All game mechanics, characters, items and actions must follow the d20 gaming rules. More specifically, the game should have:

- The rendering of a grid map consisting of cells where characters (player or NPCs) can move and interact with game elements.
- The user-interactive creation of maps along with a save/load feature to/from a file. Allow for interesting maps to be created, i.e. corridors and rooms by disallowing movement in certain squares. Cells can either be a wall or part of a room. Room cells can contain a character (player or NPC), a door, or a chest. Each map has an objective, which unlocks the exit door(s) of the map.
- The user-interactive creation of a campaign, which is essentially a connected graph that connects maps together, i.e. using a door in one map leads the character to the entry door of another map. A campaign is saved/loaded to/from a file. A campaign has an objective, the completion of which ends the game.
- A set of pre-generated campaigns are offered at the beginning of the game, from which the user chooses or lets the computer chose from randomly.
- A user-interactive character editor that allows to create player characters and NPCs. The characters should be stored in a file.
- A set of pre-generated characters is available at the beginning of the game, from which the user chooses to play the game. The characters can be placed on the map as NPCs (friendly or hostile) during map creation. Characters or NPCs need only to be minimally graphically rendered, i.e. a simple symbol in a map square is sufficient. Animation is not a mandatory requirement. Characters are created using a character editor that can be used to create characters from scratch, or edit existing characters.
- During play, items (armor/shield/weapon/boots/ring/helmet), can be found in chests or on bodies of fallen characters. Upon searching a chest or body, its content is put in the searching character's own inventory. All items usable in the game are stored in an item file and can be assigned to a character using the character editor, or placed in a chest using the map editor.
- During play, a player has an inventory panel where he can view all his items, including those that are worn and those in his backpack (i.e. not worn). This panel allows the player to select and wear some equipment from his character's inventory. Only items that are worn can be used or have an effect on the character. At most one item of each kind can be worn by a character.
- Movement and targeting should be limited to a certain range. Valid target squares for movement and targeting should be highlighted in order to help the player selecting a target.
- Mandatory class is Fighter, which can attack using either a bow (or other ranged weapon) or sword (or other melee weapon), and wear light, medium or heavy armor and a shield.
- The game is turn-based (as opposed to real-time), where all turns begin with an initiative roll, then taking an action based on the board situation current at each character's turn. Each action is limited by the movement/attack ranges of the character and the chosen weapon, taking into consideration obstacles and map configuration.
- The game is single-player, and the actions of NPCs should be kept to an absolute minimum of complexity or intelligence.
- Combat mechanics should strictly follow d20 rules, i.e. use proper dice, hit points, ability modifiers, armor class, attack modifiers, etc.
- A console window should be available, where a log of relevant game information is output and can be used to properly and clearly demonstrate that the d20 game rules are followed.

References

Game rules
Dandwiki: wiki on official d20 rules
BoLS Interactive: The Hypertext d20 SRD

Implemented games
Dungeons and Dragons Online (DDO) official site
Unofficial wiki for DDO
Neverwinter official site
Unofficial wiki for Neverwinter