Concordia University
Department of Computer Science
and Software Engineering

Advanced Program Design with C++
COMP 345 --- Fall 2016

Project Final Build Grading

1. Final Incremental Code Build Description

You must deliver an operational version demonstrating the full capacity of your system. This is about demonstrating that the code build is effectively aimed at solving specific project problems and completely implementing specific system features. The code build must not be just separated portions of the final project, but a fully operationally integrated software that can be demonstrated by its operational usage.

The presentation should be organized as follows:

1. Brief presentation of the Design, and Use of Tools as listed below under “Graduate attributes—skills”
2. Demonstration of the functional requirements as listed below under “Functional Requirements”.

You are graded according to how effectively you can demonstrate that the features are implemented. If you cannot really demonstrate the integrated features through execution, you will have to prove that the features are implemented by explaining how your code implements the features and what are the expected integration problems, in which case you may lose some marks, even if your explanations are satisfactory.

During your presentation, you have to demonstrate that you are well prepared for the presentation, and that you can easily provide clear explanations as questions are asked about your understanding of the problem being solved, the structure and functioning of your code, as well as your use of tools.

2. Team Identification

Team	Evaluator	Signature	Date	Time

3. Grading

Functional Requirements			**35**
Map and campaign creation/editing			**8**
User-driven creation of a validated map by placing elements such as entry/exit door, walls, chest, and characters (friendly and hostile). Characters placed on the map are characters that have been previously saved (see Character creation/editing). Chests should contain items placed by the user upon map creation.			3
User-driven creation of a campaign that links the entry/exit doors of different maps as a connected directed graph.			2
Loading a map from an existing file, then editing the map.			2
Verification of map correctness (at least 3 types of incorrect maps).			1
Character creation/editing			**5**
User-driven creation/editing/validation of a *fighter* character following the d20 game rules. Creation of a new character is made using the 4d6 method for the ability scores.			2
Saving/loading a character to/from a file. Only the information that cannot be calculated (i.e. ability scores, level, hit points and owned items) should be saved/loaded.			3
Play			**22**
Selecting a campaign and a player character from a list of saved ones			1
Adapting the map elements (characters and items) to the level of the player character upon entry			2
Allow the character to loot items in chests and items owned by fallen enemies.			1
Non-player characters can be friendly or enemies (see assignment 3).			1
Combat between the player character and enemies following the d20 rules using all the following factors: level, ability modifiers, hit points, armor class, attack bonus, damage bonus, multiple attacks.			5
Implementation of worn item enhancement according to the d20 rules.			2
Allow the user to toggle (i.e. being able to turn on/off) a view of any character (player and opponents) and chests content during play. Such views must update themselves automatically without user intervention as the state of what they view is changed.			2
Allow the user to toggle (i.e. being able to turn on/off) an inventory view for the player character, including worn items slots (one of each: armor, ring, helmet, boots, belt, sword, shield) and backpack (item container). The view allows to equip/unequip items, i.e. move items between worn item slots and the backpack.			2
Allow the user to toggle (i.e. being able to turn on/off) a game log that displays information about what is happening in the game: 1) the game controller records the events happening in every game phase (game setup, map loading and campaign progress, player turns switching, end phase) 2) the map records every movement made on the map, 3) the character records every attack attempted including all factors involved in its calculation and its result 4) the dice records every dice roll. All the log entries should be recorded in a unified log. It should be possible to switch on/off any of the different logging sources individually (i.e. 1,2,3,4 above).			4
Once the character steps on the exit door of a map, it goes up a level and the next map in the campaign is loaded and played; if that map was the final map in the campaign, the game is won.			2
Graduate attributes—skills			**15**
Knowledge-base	Indicator 1.3: Knowledge-base in a specific domain: demonstrated knowledge of programming principles used in the implementation.		1
Design	Indicator 4.1: Problem identification and information gathering: knowledge and correct understanding of the functional requirements and the game rules.		2
	Indicator 4.3: Architectural and detailed design: Rationale for overall project architectural structure and source code file organization. Explanation of the correct use of at least 3 design patterns such as those implemented in the individual assignments.		3
	Indicator 4.4: Implementation and validation: Correct use of C++ features leading to stable execution that has been properly tested in various situations. Runnable integration of some cppunit test cases in the project.		2
Use of tools	Indicator 5.1: Ability to use appropriate tools, techniques and resources: proficient use of tools (C++ language, libraries, project management tools, etc.) for the implementation.		2
	Indicator 5.2: Ability to select appropriate tools, techniques, and resources: justified adoption of tools in the project (compiler, IDE, libraries, project management tools, etc).		1
Communication	Indicator 7.3: Documentation: Code readability: layout, naming. Consistent use of comments. Consistent use of Doxygen code documentation.		2
	Indicator 7.4: Oral presentation: Structure and demonstrated preparation of presentation, using appropriate presentation techniques. Demonstrated knowledge of code base/clarity of explanations.		2
Total			**50**