Concordia University
Department of Computer Science
and Software Engineering

Advanced Programming Practices
SOEN 6441 --- Winter 2017

Project Build 1 Grading

Instructions for Incremental Code Build Presentation

You must deliver an operational version demonstrating a subset of the capacity of your system. This is about demonstrating that the code build is effectively aimed at solving specific project problems or completely implementing specific system features. The code build must not be just a "portion of the final project", but rather be something useful with a purpose on its own, that can be demonstrated by its operational usage.

The presentation should be organized as follows:

1. Brief presentation of the goal of the build.
2. Brief presentation of the architectural design of your project.
3. Demonstration of the functional requirements as listed on the following grading sheet.
4. Demonstration of the use of tools as listed on the following grading sheet.

You are graded according to how effectively you can demonstrate that the features are implemented. If you cannot really demonstrate the features through execution, you will have to prove that the features are implemented by explaining how your code implements the features, in which case you will get only partial marks.

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 the functioning of your code, or your required usage of the tools/techniques.

Identification

Team	Evaluator	Signature	Date

Grading

Presentation		**5**
Effectiveness, structure and demonstrated preparation of the presentation		2
Fluid exposition of knowledge of code base/clarity of explanations		3
Functional Requirements		**30**
Character creation/editing		**12**
User-driven creation/editing of a validated *fighter* character following the d20 game rules: level, ability scores, ability modifiers, hit points, armor class, attack bonus, damage bonus, multiple attacks, owned items (chosen from the items saved in the items file).		4
Inventory view, allowing to view worn items slots (one of each: armor, ring, helmet, boots, belt, sword, shield) and backpack. Allow to equip or unequip items, i.e. move items between worn item slots and the backpack.		3
Item enhancement mechanism: worn items have an effect on the character's abilities.		3
Saving/loading a character to/from a file (the characters file).		2
Map creation/editing		**8**
User-driven creation/editing of a validated map of user-defined size by placing elements such as entry/exit door, walls, chest (containing items chosen from the items saved in the items file), and characters (friendly or hostile, chosen from the characters saved in the characters file).		4
Saving/loading a map to/from a file (the maps file).		2
Verification of map correctness before saving (at least 3 types of incorrect maps).		2
Campaign creation/editing		**5**
User-driven creation/editing of a validated campaign that links the entry/exit doors of maps (chosen from the maps saved in the maps file) as a connected directed graph.		**3**
Loading/saving a campaign to a file (the campaigns file).		**2**
Item creation/editing		**5**
User-driven creation/editing of validated items (armor, ring, helmet, boots, belt, sword, shield).		3
Loading/saving an item to a file (the items file).		2
Programming process		**15**
Architectural design—short document including an architectural design diagram. Short but complete and clear description of the design, which should break down the system into cohesive modules. The architectural design should be reflected in the implementation of well-separated modules and/or folders.		3
Software versioning repository—well-populated history with dozens of commits, distributed evenly among team members, as well as evenly distributed over the time allocated to the build. A tagged version should have been created for build 1.		3
API documentation—completed for all files, all classes and all methods.		3
Unit testing framework—at least 10 relevant test cases testing the most important aspects of the code. Must include tests for: (1) wearing items should correctly influence the character's abilities; (2) map validation; (3) character cannot wear more than one item of each kind.		3
Coding standards—documented description of coding standard used. Consistent and proper use of code layout, naming conventions and comments, absence of "commented out" code.		3
Total		**50**

Notes

Additional Narrative Specifications

Characters

The baseline of this assignment is to create characters belonging to the *fighter* class. The character constructor must automatically generate the following, based on the level (provided at creation time) and class of the character: (1) ability scores (generated randomly using the 4d6 generation method) and ability modifiers, (2) hit points (based on constitution modifier and level), (3) armor class (based on dexterity modifier and worn armor), (4) attack bonus (based level and strength/dexterity modifiers), (5) damage bonus (based on strength modifier, only for melee weapons). The character must be able to wear one item of each of the following kinds: armor, shield, weapon, boots, ring, belt, helmet, each of which should properly enhance the ability score modifiers of the character. See more details below for items.

Maps

Creation of custom maps of variable size to be determined prior to the creation of the map. It is advised (for simplicity) that you design the map as a grid, where each grid cell is either (1) a default floor cell where a character can move or (2) a wall where a character cannot move, or (3) an occupied cell containing a character, opponent, chest, or a door. The editor should be designed to allow the creation of a blank map given height and width chosen by the user, and provide functionalities to set any cell to anything it might eventually contain as stated above.

Campaigns

A campaign is what is played by a character. A campaign is essentially a connected directed graph where the nodes are maps, and edges represent navigation between the exit door of a map and the entry door of another map. The campaign editor should allow the user to select maps saved in the maps file and connect them to form a directed connected graph.

Items

Items can be of type helmet, armor, shield, ring, belt, boots, and weapon. The character is allowed to wear at most one of every type of object, and has a backpack that can contain 10 additional items. Items may be enchanted with a +1 to +5 enchantment bonus that upon wearing will eventually influence one of the character's ability scores (Strength, Dexterity, Constitution, Intelligence, Wisdom or Charisma), armor class, attack bonus, or damage bonus. Different types of items should provide only with a certain possibility of enhancement types as per the table below.

Item	May increase either
Helmet	Intelligence, Wisdom, Armor class
Armor	Armor class
Shield	Armor class
Ring	Armor class, Strength, Constitution, Wisdom, Charisma
Belt	Constitution, Strength
Boots	Armor class, Dexterity
Weapon	Attack bonus, Damage bonus