Concordia University
Department of Computer Science
and Software Engineering

Advanced program design with C++
COMP 345 --- Fall 2014

Individual assignment #1

Deadline:	Friday, October 17th, 2014
Evaluation:	5% of final mark
Late submission:	not accepted
Teams:	this is an individual assignment

Problem statement

This is an individual assignment. It is divided into three distinct parts. Each individual student is expected to select one of these parts as his/her assignment. Each part is about the development of a part of the topic presented as the team project. Even though it is about the development of a part of your team project, each assignment has to be developed independently of the others and is not to be presented as an integrated part of the team project, or include the implementation of one another's aspects. Each member of your team is free to choose to do any part, and is expected to follow a different design approach than the other team members that have selected the same assignment topic. Note that the following descriptions describe the baseline of the assignment, and are related to the project description (see the course web page for a full description of the team project).

Part 1: Map

Implement a group of C++ classes that allow the generation of custom maps for the tower defense game. This must allow for the creation of custom maps of variable size to be determined prior to the creation of the map. As stated in the project description, it is advised (for simplicity) that you design the map as a grid, where each grid cell is either (1) a scenery cell where towers can be placed, and where the critters are not allowed to move or (2) a path cell where towers cannot reside, and where the critters are allowed to move. One and only one path cell should be assigned as the entry point of the map and one and only one path cell should be assigned as the exit point of the map. There should be a connected graph consisting of path cells that connects the entry point cell to the exit point cell. The map implementation should be designed to allow the creation of a blank map given length and width, and provide member functions to set any cell to anything it might eventually contain as stated above. Your class should have a method to verify the validity of a map, which verifies the validity of the map according to the above specifications.

Part 2: Towers

Implement a group of C++ classes that allow the creation of towers following the tower defense game rules. Towers should have the following characteristics: buying cost, refund value, range, power, rate of fire, etc. Different types of towers should be created, whose difference is mainly in their behavior, for example in the effect that their bullets have: direct damage, area of effect damage, slowing, additional damage to specific kinds of critters, etc. At least three different types of towers must be created. A tower can also have different levels, organized sequentially and with gradually increased capacities, that allow it to eventually destroy critters more effectively. A tower can be bought for an initial cost, and be further upgraded to subsequent levels for a certain cost for each level. At any time, it can be sold for a certain amount depending on its level. When operating, a tower first detects potential targets within its range, then selects one target and shoots at it, which then inflicts damage and/or applies special effects on the target critter.

Part 3: Critter group generator

Implement a group of C++ classes that allows the creation of a group of critters following the tower defense game rules. Individual critters should have the following characteristics: reward, hit point, strength, speed, level, etc. The critter group generator is a component of the game that is called at the beginning of each wave to create a group of critters whose strength is adapted to the level of difficulty of the next wave (assuming that every successive wave is of increasing difficulty). The critter group is a list of critters that will appear sequentially one after the other on the entry point of the map when the game is played. When it is its turn to act, a critter should determine where it will move, assuming that it knows where the exit point is of the map, then move at a certain speed. A critter may be attacked by towers, which will reduce its hit points, eventually dying if it reaches zero hit points. For every critter killed, the player should get a coin reward proportional to the level of the critter. Once a critter reaches the exit point, it steals coins from the player at a rate determined by the critter's strength.

Assignment submission requirements and procedure

You are expected to submit a group of C++ files implementing a solution to one of the problems stated above (Part 1, 2 or 3). Your code must include a *driver* that allows the marker to compile and execute your code on a standard lab computer. The driver should simply create a map, tower, or critter group generator and somehow demonstrate that the map, tower, or critter group generator was created following the above-mentioned specifications, and behaves as mentioned above. The use of unit testing such as cppUnit is not mandatory but encouraged. Along with your submitted code, you have to explain your analysis and design. Briefly explain the game rules involved in the creation of your assignment, citing external sources for specific game rules. Briefly describe the design you adopted as a solution. The design description can be backed-up, for example, by doxygen-generated documentation, regular code comments, or a simple diagram. The focus of this course being the coding aspect of software development, you are discouraged to submit extensive documentation, and if you do, you will not receive additional marks for doing so.

You have to submit your assignment before midnight on the due date using the ENCS Electronic Assignment Submission system under the category "programming assignment 1". Late assignments are not accepted. The file submitted must be a .zip file containing all your code. You are allowed to use any C++ programming environment as long as they are usable in the labs. No matter what programming environment you are using, you are responsible to give proper compilation and usage instructions to the marker in a README file to be included in the zip file.

Evaluation Criteria

Solution:	
Clarity and correctness of statement of game rules involved:	5 pts
Compliance of solution with stated problem:	20 pts
Simplicity and appropriateness of the solution:	5 pts
Clarity of design description:	5 pts
Programming style:	
Code readability: naming conventions, clarity:	4 pts
Coding style: `.h` and .cpp files, use of comments:	4 pts
Relevance of driver and completeness of presented results:	7 pts
Total	**50 pts**