EE6411/ED5021 Programming Project

1. Overview

This programming project is to be undertaken in groups of two to three students – please add your groups to the "Wiki" section of the module's SULIS page before 15:00h on Tuesday, 12.11.2013. Any students that haven't put down their groups by this deadline will be assigned by me into groups.

Your task is to write a simple Fantasy Game played through a text interface. The aim of the game is to collect as much gold as possible by defeating enemy characters. A (partially) implemented executable can be found on the module's SULIS page to give you an idea of what is expected. If you have any queries, please post them to the Questions & Answers Forum.

You have to design and implement two (dynamic) ADTs: one for the game board (size is selectable by user at run-time) and one for the list of items that a character is carrying (the amount is limited only by the character's strength). All ADTs and other entities of the game must be implemented as C++ classes. Your program must use C++ exception handling to deal with errors (improper input, lack of resources etc.).

The project is worth 30% of the module. Deadline for the project is Friday, 29.11.2013 (week 12). Solutions are to be submitted electronically via the module's SULIS page (please submit a single archive). A complete solution consists of the following:

- Formal specification of the two ADTs (as discussed in lectures and labs) in electronic format (preferable MS Word or PDF - for use of other formats contact me first to guarantee that I can read your documents).
- Overview of your class structure in an UML class diagram (in some graphics format, such as jpg, gif or pdf – do **not** submit your class diagram in a CASE tool format). However, feel free to use CASE tools (such as Visual Paradigm or Poseidon) to create your class diagrams.
- Well documented and formatted source code for the game implementation using a suitable file structure. If you are using Netbeans, please include the entire project folder in your submission, but perform a clean first to minimise its size. Otherwise, please make sure to include a suitable make file in your project. Please include all your student IDs in the name of the project!

The marking scheme for this project is as follows:

ADT Specifications	20
ADT Implementation	20
Class Structure	10
Game Implementation	50
Total:	100

2. Rules of the Game

2.1 Environment:

The game is played on a rectangular board consisting of X by Y squares - both values are freely selectable by the player at the beginning of each game. Each square can hold an item, an enemy or be empty. At the beginning of a game the board is initialised, i.e. each square is assigned either an enemy, an item or stays empty. After choosing a character the player begins the game at the starting square. The game starts on daytime. Daytime and Night-time alternate every 5 moves. The game is based on simple commands entered via the keyboard. The following commands are available (use the first character for input):

- North, South, East, West – character moves to the square in the indicated direction (north=up, south=down, east=right, west=left)
- pick up – character attempts to add item on square to inventory (if the resulting total weight exceeds the characters strength, this will fail)
- drop – drops one item (prompt user for the item to be dropped) to the current square (if the square already contains an item, request is ignored)
- attack – character attacks enemy on current square (if this results in the characters health dropping to below zero, it is defeated)
- look – prints out information about the current location (coordinates of the field and information about any enemies or items on that field)
- inventory – prints out the list of items the player is currently carrying and the amount of gold that the player has earned.
- exit – ends the game

2.1 Characters:

Characters have the following properties:

- Attack (A) – indicates the attack-force of the character
- Attack Chance – indicates the probability of a successful attack, e.g. Hobbits have only a 1 in 3 probability of a successful attack.
- Defence (D) – indicates the defensive abilities of the character
- Defence Chance – indicates the probability to defend against a successful attack, e.g. Hobbits can defend against 2 out of 3 successful attacks.
- Health (H) – indicates the health status of a character. Once the health level has reached 0 the character is defeated.
- Strength (S) – indicates the weight a character can carry.

2.2 Race:

The race defines the abilities of a character. The following races are available:

- Human
- Elf
- Dwarf
- Hobbit (Halfling)
- Orc

The following table shows the basic abilities for each race

Race	Attack	Attack Chance	Defence	Defence Chance	Health	Strength
Human	30	2/3	20	1/2	60	100
Elf	40	1/1	10	1/4	40	70
Dwarf	30	2/3	20	2/3	50	130
Hobbit	25	1/3	20	2/3	70	85
Orc*	25(45)	1/4(1/1)	10(25)	1/4(1/2)	50	130

*Value is divided into day(night), ie. Orcs are very good at night, but poor at day

2.3 Special Race Abilities:

- Human: Successful defences never cause damage.
- Elf: Successful defences always increases health by 1
- Dwarf: Successful defences never cause damage
- Hobbit: Successful defences cause 0-5 damage regardless of attack value
- Orc: During daytime, successful defences cause 1/4 of adjusted damage (ie. 1/4 of attacker's attack – defender's defence). During nighttime, successful defences cause increase of health by 1

2.4 Items:

Each character can carry items up to a weight indicated by its strength. Items modify the basic abilities of characters. A character can carry only one item of each category, except rings, of which a player can carry as many as the character's strength allows. The following items are available (category is indicated in brackets):

- Sword (Weapon), weight 10: increases attack by 10
- Dagger (Weapon), weight 5: increases attack by 5
- Plate Armour (Armour), weight 40: increases Defence by 10, decreases Attack by 5
- Leather Armour (Armour), weight 20: increases Defence by 5
- Large Shield (Shield), weight 30: increases Defence by 10, decreases Attack by 5
- Small Shield (Shield), weight 10: increases Defence by 5
- Ring of Life (Ring), weight 1: increases Health by 10
- Ring of Strength (Ring), weight 1: increases Strength by 50, decreases Health by 10

2.5 Combat Rules:

Enemies have the same characters/types as players. Whenever a player attacks an enemy, the enemy will immediately afterwards attack the player (unless the enemy has been defeated on the last attack). For each attack the following rules apply: First, the attacker resolve the attack success determined by the attack chance of the attacker's race. On successful attack, the defender resolves the defence success determined by the defence chance of the defender's race. If the defence fails, the attack amount of the attacker (adjusted by the defence value of the defender) is subtracted from the defender's health value. On a successful defence, the damage is determined by the character's race (see special race abilities).

Combat example: Hobbit Frodo (A20D20H70) is attacked during daytime by Orc Gargoyle (A25(50)D10(30)H50). Assuming Gargoyle succeeds with his attack (1/4 chance) and Frodo's defence succeeds (2/3 chance), then Frodo's health is reduced by random value 0-5 (race ability). If Gargoyle's attack fails, Frodo's health stays unchanged and if Frodo's defence fails his health is reduced by the attack value minus his defence (A25-D20 = 5).

Combat example 2: Hobbit Frodo (A20D20H70) attacks during daytime Orc Gargoyle (A25(50)D10(30)H50). Assuming Frodo's attack succeeds and Gargoyle's defence fails, then Gargoyle's health is reduced by 10 (A20-D10).

Above calculations assume that all characters are carrying no items. Any worn items will change the outcome of the attacks according to their modifications. If the health of any character drops to zero or below, that character is defeated. If the player defeats an enemy he will gain the enemy's basic defence value in gold, i.e. if a player kills a basic human enemy that player will gain 20 gold pieces.

Note: If you want to extend the list of characters and/or items, feel free to do so. However, above character/item lists are the minimum for a successful project.

3. Miscellaneous & Hints

- There is no need to include a graphical user interface or any kind of artificial intelligence (the player is the only active character).
- Make sure your ADTs (and their implementation) can handle any size – use of static arrays is not sufficient. Feel free to utilise STL classes (Standard Template Library).
- Your ADTs and your program **must** use C++ Exception Handling to deal with any errors (e.g. improper values, insufficient resources etc.).
- Your solutions **must** use object oriented programming (e.g. classes) – a solution in an imperative programming style (e.g. just main and a collection of functions, but no classes) will be deemed a failed project and will receive 0 marks.
- Proper use of inheritance & polymorphism will greatly reduce the programming effort for this project (and it will increase the virtue of your class structure and of the game implementation).
- Only standard C++ libraries are allowed – if you wish to utilise any other libraries than <iostream>, <cstdlib>, or any of the STL libraries, please contact me first to obtain permission.
- Random numbers can be generated with the function rand() defined in <cstdlib>. This function will always return the same sequence of pseudo-random numbers (Range 0-MAX_RAND, use rand()%(x+1) to get values 0-x). The function srand(int seed) can be used to set the seed of the generator. Either prompt the user for a random seed (to be set at the beginning of the game) or use some other value like the current time etc. Please note that the srand() function should be called only once for each game.
- Make sure to provided required constructors/destructor/operators for your ADTs