|epcc|

MSc in High Performance Computing
Software Development Coursework

Alistair Grant

January 8, 2016

1 Introduction

The assessed coursework for Software Development is intended to apply and evaluate the processes and techniques covered in the course. The coursework should reinforce the points made in the course and allow you to realise issues that come in applying theory in practice. This does not mean that you have to stick only to the techniques in the course, prior experience and techniques should always be considered valid (where proven to work and be reliable). The aim of the assessment is to look beyond just the coding aspects of software development, to consider the software development process as a whole.

1.1 Deck Building Card Game

The objective of the coursework is to analyse and improve a prototype version of a deck building card game written in Python. Deck building games are where players are provided with a set of initial starting cards and using these they can "buy" additional cards from a shared deck to augment their own sets.

Depending on the type of game, each card defines quantities such as strength, health, wealth and the cost to buy the card. Buying cards allows a player to improve their strength to attack their opposing players or to increase some other quantity, like their wealth. Each player can build a deck according to their own play style and what cards are available at each step of the game.

In the deck building game used in this coursework, each player starts with a health level. The goal is for a player to increase their wealth and strength, by buying cards, and playing these cards to "attack" their opponent to reduce their health. A player wins if they reduce their opponent's health to zero.

Each card in this game has four values associated with it:

- Name - what the card is called e.g. Squire, Serf, Baker, Thug etc.
- Strength - the contribution of this card to an attack.
- Money - the contribution of this card to the available funds to buy new cards.
- Cost - how much money it costs to buy this card.

An example card is Squire:

- Name - Squire
- Strength - 1
- Money - 0
- Cost - 0

Another example card is Serf:

- Name - Serf
- Strength - 0
- Money - 1
- Cost - 0

Each player starts with a deck of 10 cards, and each player has the same starting deck of 2 Squires and 8 Serfs. These provide each player with a small amount of money and attack strength.

Between the players there lies central line of 5 cards. The remaining cards are held within a main deck. Players can buy cards from the central line, and, if a card is bought, then it is replaced by the next card from the main deck, if there are any cards left.

The game continues until either the central line of available cards is empty or one player is reduced to zero or less health.

Each player has:

- Deck - where a player draws their cards from. Initially the deck is 10 cards, but they may buy more.
- Hand - the cards (normally 5), that a player has available to play during their turn. These are drawn from the player's deck.
- Active Area - the cards played by the player during their turn (where players put cards into play). The player can play 1 or more cards from their hand.
- Discard Pile - where played cards are put at the end of the player's turn and where cards the player has bought are also put.

When it is a player's turn, they have the following options:

- Play one or more cards - the player puts 1 or more cards from their hand into the active area. Money to buy more cards and attack strength to attack opponents is calculated by the sum of these quantities on the cards in the active area for the player.
- Attack - this will attack their opponent, reducing health by the current attack strength (the sum of the strength of cards in the active area for the player).
- Buy - buy one of more cards from the central line based on current money value (the sum of the money on the cards in the active area for the player). Any bought card is put into the player's discard pile.
- End Turn - pass turn to the opponent - all the player's cards in the active area are put into their discard pile, and the player draws a new hand from their deck. If their deck is empty, then the discard pile is shuffled and moved to the deck.

To attack the opponent or buy cards, a player must have cards in their active area.

As an example of a player's turn, imagine that the central line of cards has the following cards:

- Name Caravan costing 5 with attack 1 and money 5
- Name Tailor costing 3 with attack 0 and money 4
- Name Baker costing 2 with attack 0 and money 3
- Name Baker costing 2 with attack 0 and money 3
- Name Thug costing 1 with attack 2 and money 0

Player One has a hand with 2 Squires and 3 Serfs and decides to play all of these. They move these cards into the active area. Together, these cards give the player Attack Strength 2 and Money 3.

Player One chooses to buy the Tailor card. This card is transferred from the central line into the player's discard pile.

Player One then chooses to attack their opponent. Their opponent's health is reduced by the active attack strength of 2.

Player One now decides to end their turn. They put all the cards from the active area into their discard pile. They then draw a new hand from their deck.

2 Assessment

The assessment is broken into three submissions across the semester. This will allow the opportunity for you to receive feedback between submissions.

Note - do not submit all parts at the same time, this will not get marked any sooner and you will miss the opportunity to improve on past performance.

The three components to the coursework are:

- Planning and Risks
- UI and Evaluation Plan
- Improvements and Reflection

2.1 Planning And Risks

The first task is to analyse the code and identify its defects. For each defect, propose a solution. Once you have your set of defects and solutions, create a plan for implementing some of these solutions which contains estimates of time and effort for each solution. This plan should contain a risk analysis for the work you propose.

The first submission is a document summarising the following:

- defective aspects of the code
- possible corrections
- a plan for correcting some of the defects - with time and effort estimation
- risk analysis for the work

Note: - it is not required to fix everything in the code - but the changes you choose to make must be justified and must make a significant improvement to the code.

Note: - you can ask the course organiser if what you are planning to do would constitute significant improvement.

This document should include a Gantt chart or similar plan to illustrate how you have planned your workload.

This component of the submission is worth 30 marks out of 100 marks total for the course.

2.2 UI and Evaluation Plan

The supplied code has a very basic command line user interface (UI). The task in this component is to design and prototype a new and improved UI, note that this does not have to be a functional UI prototype.

The UI can be in any form you want to design - it must have an accessible design such that it can be evaluated. The UI design choices should be explained.

The second part of this submission will be a plan for how you will evaluate the usability of the UI design. This can be done in any fashion, but it must be clearly laid out as to what is being evaluated, how it is being evaluated, who is evaluating and how the results will be interpreted.

The second submissions should contain:

- UI prototype and design
- summary of design choices
- UI evaluation plan

This component of the submission is worth 30 marks out of 100 marks total for the course.

2.3 Improvement and Reflection

The main part of this submission should be a new version of the code, implementing the changes you proposed in Planning and Risks. Your submission should include any required files such as build and/or test files. The submitted code must be runnable and be able to be used to play games.

Note: The code submission should be a link/reference to a source code control system such as GIT or SVN. The repository should contain everything needed to compile the code - including build files which get any required dependencies. Non-working or non-compiling code will be marked down accordingly.

Please note that your code (and any supplied tests) must satisfy the following:

- It must be possible to build and run your code and tests on the Physics Computational Lab machines.
- If your code or tests requires additional packages or software to build or test that are not already available on the Physics Computational Lab machines then you should document what these packages or software are and how to set them up - this should include installation and run scripts which will create the required environment for the code to run.

To accompany the code a short report is required which should contain a summary of the changes to the code that you have made and the benefits they give to the code. The report should also contain your personal reflection on the whole process, including how your actual work compared with the plan you created at the start and did you encounter any risks you had not considered.

The short report should include a future enhancement you think would improve or add value to the code - you should summarise the purpose of your proposal and highlight any risks or impact that it may have on the existing code.

The final submission should contain:

- Improved Code
- Summary of Changes
- Reflection on Process
- Future Enhancement

This component of the submission is worth 40 marks out of 100 marks total for the course.

3 Advice

Note Important - this exercise does not rely on being fluent in Python - no matter which language was used some students in the course would be unfamiliar with it - good practice and conceptual understand-

ing are the key elements in this exercise which carry across languages and tasks.

The code provided is a basic game with command line interface. The implementation does work and will play a game. However, the code is badly implemented and is difficult to maintain and may contain errors and bugs.

As stated the objective of this assessment is to rework this code. This will involve doing an analysis of what is wrong with the code, planning what will be done to correct it, risk analysis, scheduling, prototyping a UI (does not need to be functional) and improving the code.

In this exercise it is important to remember that the code must work at the end of the process, but it is not necessary to fix all of the problems in the code. You have to balance the workload with the workload given to you in other parts of this course and in other courses.

This exercise is not intended to be worked on during the teaching sessions of this course. Questions about the exercise can be asked in person or by email to Alistair Grant (alistair.grant@ed.ac.uk) or submitted via Learn.

There is no single correct submission to this coursework - take in account the general and individual feedback you will receive to improve your performance during the course.

4 Where and When

4.1 Source Code

The source code for the assessment is available in the Software Development module on Learn under the Assessment link. There is a Python source file for the deck building card game, called dbc.py.

Note: This code currently runs on the Physics Computational Lab machines and has been tested under Python 2.7.

To run the code, use the following command:

```
python dbc.py
```

This should result in output similar to:

```
Available Cards
Name Caravan costing 5 with attack 1 and money 5
Name Tailor costing 3 with attack 0 and money 4
Name Baker costing 2 with attack 0 and money 3
Name Baker costing 2 with attack 0 and money 3
Name Thug costing 1 with attack 2 and money 0
Supplement
Name Levy costing 2 with attack 1 and money 2
Do you want to play a game?:
```

4.2 Deadlines

- Planning and Risks - 5pm 05/02/2016
- UI and Evaluation Plan - 5pm 04/03/2016
- Improvement and Reflection - 5pm 25/03/2016

Feedback and provisional marks should be available within fifteen days of the deadlines above.

4.3 Late Submissions Policy

The full submissions policy can be found in the Taught Assessment Regulations for Academic Year 2015/16. Below is a summary of the main guideline:

- If assessed coursework is submitted late without an accepted good reason, it will be recorded as late and a penalty will be exacted. For coursework that is a substantial component of the course and where the submission deadline is more than two weeks after the issue of the work to be assessed, that penalty is a reduction of the mark by 5% of the maximum obtainable mark per calendar day (e.g. a mark of 65% on the common marking scale would be reduced to 60% up to 24 hours later). This applies for up to five calendar days (or to the time when feedback is given, if this is sooner), after which a mark of zero will be given. The original unreduced mark will be recorded by the School and the student informed of it. Such work, if completed satisfactorily before feedback is issued, is regarded as completed when completion alone is a criterion for success.
- If you submit any coursework by its deadline then it will be marked, and you cannot submit an amended version later on, to be marked.
- If you do not wish the submitted version to be marked after the deadline, and intend submitting an amended version after the deadline, then it is your responsibility to let the course organiser know, in advance of the deadline, that the submitted version should not be marked.