Advanced Software Design
Project Assignment

Dave Clarke
October 20, 2014

A Note on Assessment  Assessment of the assignment will be based on a newly developed scheme called achievement-driven learning. This scheme is described in more detail in the document “Advanced Software Design: Achievements.” In that document there is a list of achievements that need to be demonstrated to a teaching assistant in order to obtain each particular grade for the course.

Fair Warning  The system-under-design is deliberately very large and you are not expected to produce a complete design for all of it. What you are expected to do is design a well-chosen, non-trivial subset of the system. The design must support the basics of the system (you cannot omit monsters, for example) and a non-trivial subset must be worked out in detail. Ultimately, your design must be sufficient to support the process of demonstrating achievements — almost all achievements relate to the system-under-design.

1 Background

You are the member of a small software development team working for the Quagon software company. The company has decided to start developing computer games. Initially, Quagon will develop a single game (described below) for both on desktop machines and mobile devices. If financially successful, the game will be followed up by a whole family of games built upon the same game platform—which will make it easier to implement games in the future. Management has decided that little effort should be put into designing the game platform, but shrewd developers realise that thinking about the design of the game platform early in the development process will avoid costly redesign in the future.

2 Current Assignment: DungeonQuest

The goal of this project is to develop a computer game inspired by the board game DungeonQuest.1 The specification of the computerised version of the game is based on the game’s manual—the computer game should similarly be be a turn-based, multi-player board game, and the manual need be viewed only as providing guidelines on the game’s behaviour. In any case, a number of

---
1http://www.fantasyflightgames.com/ffg_content/dungeonquest/support/dq-rulebook.pdf
improvements to the game have been requested by management to make it more appealing to gamers, so these need to be considered in the design. These improvements are the following:

- The dungeon should be three-dimensional rather than two. To this end, rooms may have stairs leading to levels above or below. Some rooms might have stairs both down and up. The roof should be one big room (with arbitrary entries/exports from the below floor) and any movement in this room should be possible to do within one turn. Monsters may not enter this room. Catacombs are also present and are only accessible from the lowest floor.

- The number of floors is set at the start of the game. In order to facilitate increased size of the dungeon, it should be possible to set the duration of the game.

- To allow faster travel, a teleportation mechanism should be added. This consists of two rooms connected by a physics-defying wormhole. The rooms hosting the two ends of the teleportation mechanism are randomly selected at the start of the game, and fixed for the duration of the game.

- The game allows multiple players to play at the same time, and this should be supported the platform-under-development.

- Monsters are free to move about the dungeon in a turn-based fashion, interacting with all of its chambers. Thus, monsters may also fall into traps and encounter each other and fight and so forth.

- Each move has a time limit. Whenever the time limit expires, the player automatically passes. Monsters are then free to move about. Any other time-dependent events can also be (de)activated.

- The game supports a trading system. This extension requires changing loot cards into not just being a random trinket with an attached gold value, but potentially also being valuable items with consequences in the game. For example, a loot card can be an armour plate that protects against certain kinds of traps or protects the bearer in combat, or a healing potion, etc. Think about whether this requires any substantial changes to other things as well. It might.

Clearly, a player should be able to sell and buy loot. If two players are near each other (in the same room, or adjacent rooms, depending on your game rules), they can choose to trade items. They should be able to chat to each other to haggle and barter, and trade items for other items.

Except for “classic loot” (the one that is defined in the original game), the system does not set a price on items. Rather, the players must do so themselves allowing the market to control pricing of any item. Trading happens through a two-phase proposal:

- Player A offering up something from her inventory
- Player B offering up something from his inventory in exchange
- Player A accepts or rejects the deal.
Due to the presence of the trading extension, we are expecting people to be buying and selling virtual goods for actual money. Consequently, it is very important that player’s character is not affected by hardware or connection failures, i.e., a character should not be killed or lose equipment if the connection to the server goes down or if a server crashes, since that would mean losing real money.

Quagon are anticipating a large number of players running around in the same dungeon, and multiple dungeons running in parallel. If the system lags, the user experience is disrupted, and consequently, people will stop playing. As this hurts business, the system needs to be designed to scale to such requirements.

Because Quagon is aiming towards designing a game platform, design choices need to be made as flexibly as possible so that in the future the code can be reused as the basis for a game platform. For instance, this could be handled by allowing your design to facilitate the easy addition of new objects, rooms, monsters, etc. to the game.

3 Guidelines

Here are a number of guidelines to follow when developing your design.

- It will be difficult to model the game in its entirety, especially if you are to handle all the additional requirements described above. This means that certain choices will need to be made in your design process limiting what aspects of the system will be considered in detail.

A design will be considered insufficient if it does not satisfy the following criteria:

1. The design features all the basic concepts and associations that the base game requires as well as the basic Model-View-Controller structure required to make implementing the game possible.

2. The design includes in fine detail either:
   - the combat system (Combat Cards, Monsters)
   - movement (The board, trap and door cards etc)
   - setup (building the board, distributing the cards and tokens, sun track who goes first etc)

3. The design accounts for approximately two thirds of the improvements to the game listed above.

- Start with a domain model.

- Handling multiple players will require a client-server distinction in the design. How to structure the server aspect of the system is entirely up to you. Key considerations is where information is stored, how to synchronise information across machines, and how to deal with network failure or crashing clients. In your model, it should be clear what parts of the model are on the server side, and what parts are on the client side. Most likely there will be some reuse of concepts and classes on both the server and clients, but some things are perhaps better represented differently. Aspects of the system that deal with information available over the network need to be represented or documented somehow.
• To cater for the mobile version of the game means managing unreliable connectivity and reduced bandwidth with the server. One trade-off that will need to be made is between playability and game-state consistency. If a connection to the server is lost, stopping the game would make it unplayable, but continuing without connectivity could make the game-state inconsistent when it is synchronised with the server. How can this be resolved?

• DungeonQuest is a complicated game, and while mapping it to a software platform you may discover ways of making the game better. All reasonable deviations from the original game design will be considered valid.

• During the course you will learn ways to evaluate and improve your design. These will enable you to improve your design as the course progresses. It is important to keep track of what changes you make and why.

• Although the course is mostly about the design of the system, rather than its implementation, coding up some of the system can help identify design weaknesses.

• The design work that you do will be evaluated by the teaching assistant in the weekly meetings. The teaching assistant will base his opinion not only on the actual design you produced but also on your justifications for why your design is the way that it is. Documenting these justifications will thus greatly improve your interactions with the teaching assistants and your final results.

• The teaching assistants keep track of the whether everyone in the team is participating during meetings. Failure to participate will affect your grade.

4 Contact Moments

There are two kinds of contact moments related to the assignment: weekly meetings and the final design review.

Weekly Meetings  Each week there will be a meeting with a teaching assistant (TA) to obtain assistance, discuss your design, to obtain feedback, and to demonstrate achievements. Meetings will last up to an hour. If you need additional help, ask for it—this is one of the roles of the weekly meetings.

A web software system has been developed to track your progress through the achievements. The following guidelines will help you use your weekly meetings wisely. They are not rigid, common sense should prevail.

• Before the meeting, ideally 24 hours before so that the TA has time to prepare, you need 1) inform the TA which achievements you wish to demonstrate, and 2) send the TA your current design documents.

The web software can be used for this purpose.

• You need to come up with a plan of how to demonstrate the achievements. This does not need to be written down, but it must be clear to the team. This could include who says what. Demonstrations must be concise, so you need to know what you are talking about.
 Achievements can only be demonstrated at the start of the meeting. The remainder of the meeting can be used to improve your design and understanding.

- You are entitled to one one hour meeting per week. Use the time wisely.

Arrange the meeting times personally with your TA based on your availability. The time can change weekly if required!

**Final Design Review**  The goal of the final design review is to review the design of another team and compare it to your team’s design. Show us that you have a deep understanding of the problem, that you can understand others’ designs, and that you can question them appropriately. We are not interested in notational bugs, but in semantics! Similarly, don’t quibble over choice of words, unless it actually makes a difference to the interpretation.

You are expected to hand in your design prior to such sessions and to study the other group’s design. **You will be provided with another group’s assignment to review.**

### 5 Deliverables

You need to submit your ongoing design documents to the teaching assistants each week. The designs submitted are not graded, but are used for obtaining feedback and demonstrating achievements. Most achievements cannot be demonstrated without supporting design documentation.

In order to participate in the final design review, you need to submit your design documents (to your TA) by the following deadline.

**Due date final design:** **Midday, Friday 13th December, 2014.**

The final version of the design will be used by another team in order to perform a comparative analysis of the designs. This document needs to be much better presented. It is a good idea to present your designs as a textual document describing the diagrams and other artefacts (which may appear as appendices referred to by the text.) The text itself need not be overly long, though it should highlight key aspects of your designs and indicate key choices made, to make it understandable to a third party.

The design review will take place during the week 15–19 December, 2014.

If it is judged by the lecturer and teaching assistants that your document is inadequate for others to review, due, for example, to severe incompleteness or poor presentation, then your team will not participate in the review process, which will affect your ability to demonstrate all achievements. This applies also if your design is delivered late (without prior negotiation).

**Failure to submit design documents will make it difficult to demonstrate certain achievements. Therefore submitting something is always better than submitting nothing.**

### 6 Modalities

1. You will work on the assignment in a group with (ideally) five members. If the size of your team drops below 3 members, your team will be merged with another team.
2. You are strongly encouraged to create groups with members of mixed backgrounds (programmes, experience, Swedishness, ...).

3. If you have not signed up for a group before the second lecture, you will be assigned to one.

4. Your time should not be spent getting as much of the system design completed as possible. The design activity must be seen as a vehicle for learning about software design, and as a means by which the group will be able to demonstrate achievements.

5. Deadlines are absolute, unless negotiated well in advance (more than a week beforehand).

6. In addition to the final design document, you are required to provide all of your auxiliary documents and code along with your submission for the team reviewing your design to examine as they see fit.

**Final Deadline**

The last date to tick off achievements is 19 December, 2014.
Do not leave everything until the last minute.