A Note on Assessment

Assessment of the assignment will be based on a newly developed scheme called (tentatively) 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 (or learning outcomes) that need to be satisfied in order to obtain each particular grade for the course. Each achievement needs to be demonstrated to a teaching assistant. For the purpose of this document, we will refer to this process as ticking off achievements.

Fair Warning

The assignment description is very large and you are not expected to complete it all. Instead, you are to engage in three iterations of an iterative development process, scaled down to the time you have available. Your key objective is to tick off achievements—designing the system described here is means for doing this.

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. Due to the tentative nature and financial risk of this particular venture, management have decide that little effort should be put into designing the game platform. Developers, however, need to be aware of this future possibility when making design decisions.
2 Current Assignment: DungeonQuest

The goal of this project is to develop a computer game inspired by the board game DungeonQuest. The specification of the computerised version of the game is based on the game’s manual—the computer game should similarly be a turn-based, multi-player board game, but the manual need be viewed only as providing guidelines. Any reasonable improvement is acceptable. In any case, a number of improvements to the game have been requested by management to make it more appealing to gamers. These 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/exits 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 only accessible from the lowest floor.
- The number of floors is set at creation time of the dungeon. 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 by a server managing the games.
- 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 (though presumably, they know where most of the traps are and avoid them).
- 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.

\[1\text{http://www.fantasyflightgames.com/ffg_content/dungeonquest/support/}
\text{dq-rulebook.pdf}\]
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’s in the game proper), 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.

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.

The following aspects are central to the game and should be considered:

- interaction with the player from a Model-View-Controller perspective—note that you should not design or implement a GUI, just design the underlying object
- movement of all living entities
- teleportation
- combat.
• Although Quagon is aiming towards designing a game platform, the additional costs of designing such a game platform at this stage mean that you need to avoid spending too much effort on this. Instead, the initial development effort should focus on the DungeonQuest game, but 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. At this stage, this could be handled by allowing your design to facilitate the easy addition of new objects, rooms, monsters, etc. to the game.

• 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.

• The software system described is large and it is unreasonable to expect that you will be able elicit all the requirements, design all aspects and implement the whole product within the duration of the course. You are not expected to! You are expected to produce enough so that you can tick off achievements.

• During the course you will learn ways to evaluated 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.

4 Development Process

You are required to follow an iterative development process, where each iteration lasts for approximately two weeks. During each iteration you must perform some analysis, design and implementation—at the early stages, the implementation
will serve as an exploration only some of the more rudimentary aspects of the design. Later iterations will build on earlier ones, and upon feedback provided by discussions with the teaching assistant at your group meetings.

The following is a rough guideline of how the iterations will proceed.

First Iteration Perform a requirements analysis based on this document and the game rules. Elicit brief use cases. Prioritise use cases and select a few to elaborate in more detail. Focus on the domain model (identify the most important domain concepts and how they relate to each other). Perform a design based on the chosen use cases. Implement a key part of this design, ultimately to experiment to see whether it makes sense, thereby providing feedback for future iterations.

Second Iteration Continue the requirements analysis to capture any missed cases. Select another collection of requirements to refine into use cases. Adapt the design to incorporate these—and any feedback provided either by the teaching assistant or based on results of implementation experiments. Expand the implementation.

Third Iteration Elicit more requirements and expand upon them, if required. Extend design to handle additional requirements not previously considered and adapt based on feedback provided by TA and results of experiments. Expand the implementation and prepare final documentation.

5 Contact Moments

There are three kinds of contact moments related to the assignment: weekly meetings, two whole class tutorials 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 tick off 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.

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, you need to send the stuff you want to talk about to the TA and inform the TA of the achievements you wish to tick off.

- You need to come up with a plan of how to tick off the achievements. This does not need to be written down, but it must be clear to the team. This could include who says what.

- Achievements can only be ticked off 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!

Whole Class Tutorial The last two lecture slots will be run as tutorial sessions, in which you will have the opportunity to ask questions about design. Ideally, all teaching staff will be available at these sessions to help you!

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.

6 Deliverables

There are three deliverables that need to be produced for the project.

1. First snapshot of design. Due **midday, Friday 15th November, 2013**.

2. Second snapshot of design. Due **midday, Friday 29th November, 2013**.

3. Final version of design. Due **midday, Friday 13th December, 2013**.

At each iteration you are required to submit the current version of your design documentation. For the first two iterations, the submitted documents are used purely for the purpose of feedback and as a kind of check-pointing—as successive iterations will likely present improvements to the design, having a check-pointed version will make it easier to demonstrate and discuss such improvements—some achievements depend on having submitted at least one snapshot. This means that the first two snapshots may be somewhat informally presented.

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 16–20 December, 2013.
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 tick-off all achievements. This applies also if your design is delivered late (without prior negotiation).

For whatever documents you produce, it is important for iterations two and three to highlight the changes made to the outcome of the previous iterations—that is, document what improvement and redesigns have been made.

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

7 Modalities

1. You will work on the assignment in a group with four or 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, ...).

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. Instead focus on key elements, design and implement them, and then expand upon them in subsequent iterations. The design activity must be seen as a vehicle for learning about software design, though ultimately the as a means by which the group will be able to tick off 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 20 December, 2013.

Do not leave everything until the last minute.