Project

Aims:

- Gain experience working in a small team-based environment
- Appreciate issues in user interface design
- Learn practical aspects of graphical user interface programming
- Learn more about the Java class libraries
- Learn the application of design patterns.

Due Dates:

Milestone 1: 11:59PM Sunday 26th August (Week 5) (Feedback: Week 6 Lab)

Milestone 2: 11:59PM Sunday 16th September (Week 8) (Feedback: Week 9 Lab)

Final milestone: 11:59PM Friday 19th October (Week 12) (Demonstration: Week 13 Lab)

Moodle Peer assessment: 11:55PM Sunday 21st October

Value: 25%

Overview

You have received a request from a client for an application that enables the creation and playing of dungeon-style puzzles. You will form a team of 4 (in some cases 5) people from your tutorial and follow an agile development process to design and implement a desktop Java application that satisfies the requirements of the client (see below). The final piece of software you deliver is expected to be of high quality and demonstrate the knowledge and skills you

have acquired in this course.

Teams

Your first step in this project will be forming a team with other members of your tutorial group. You will do this under the guidance of your tutor in the week 4 tutorial. While they will try to accommodate your preferences for teammates, they may have to make adjustments to ensure no groups have less than 4 members.

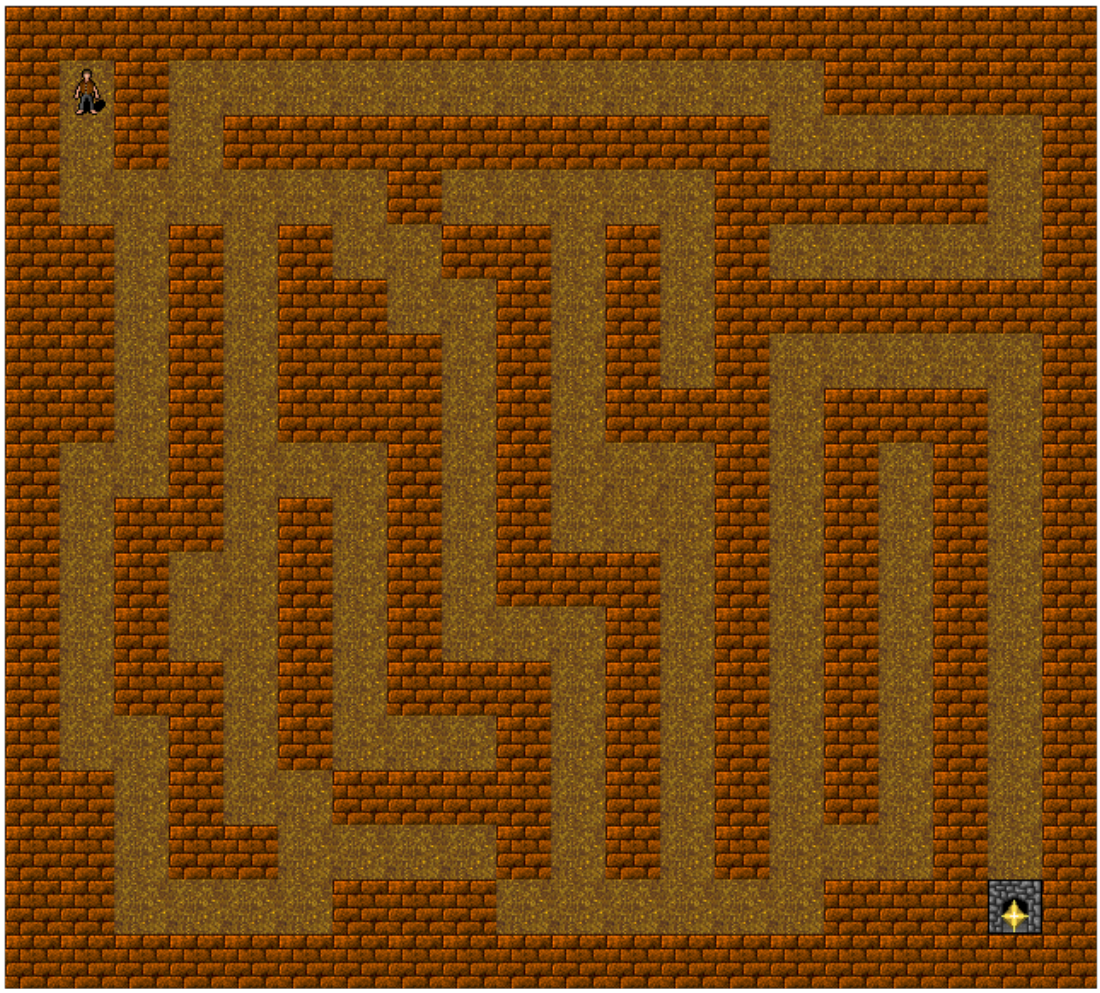
Once your team has been formed **one** member will need to register it:

1. Go to https://cgi.cse.unsw.edu.au/~cs2511/github/run.cgi/login and sign in
2. Click on the 'Group Work' button. This will show you a form that you must fill out
3. Select your tutorial from the left drop-down menu and think of a team name (do NOT use any emojis or similar special characters)
4. Submit the form.
5. Follow the link in the notification to add your other team members

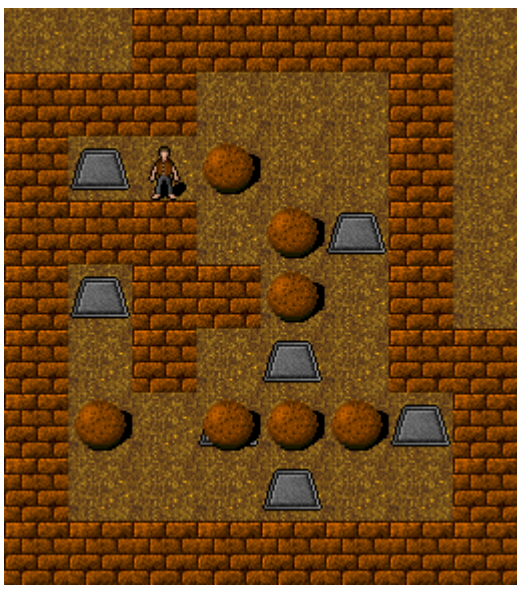
This will create a team within the `unsw-cs2511` GitHub organisation. You will need to create a repository for the team that you will use for development and collaboration.

Preliminary client requirements

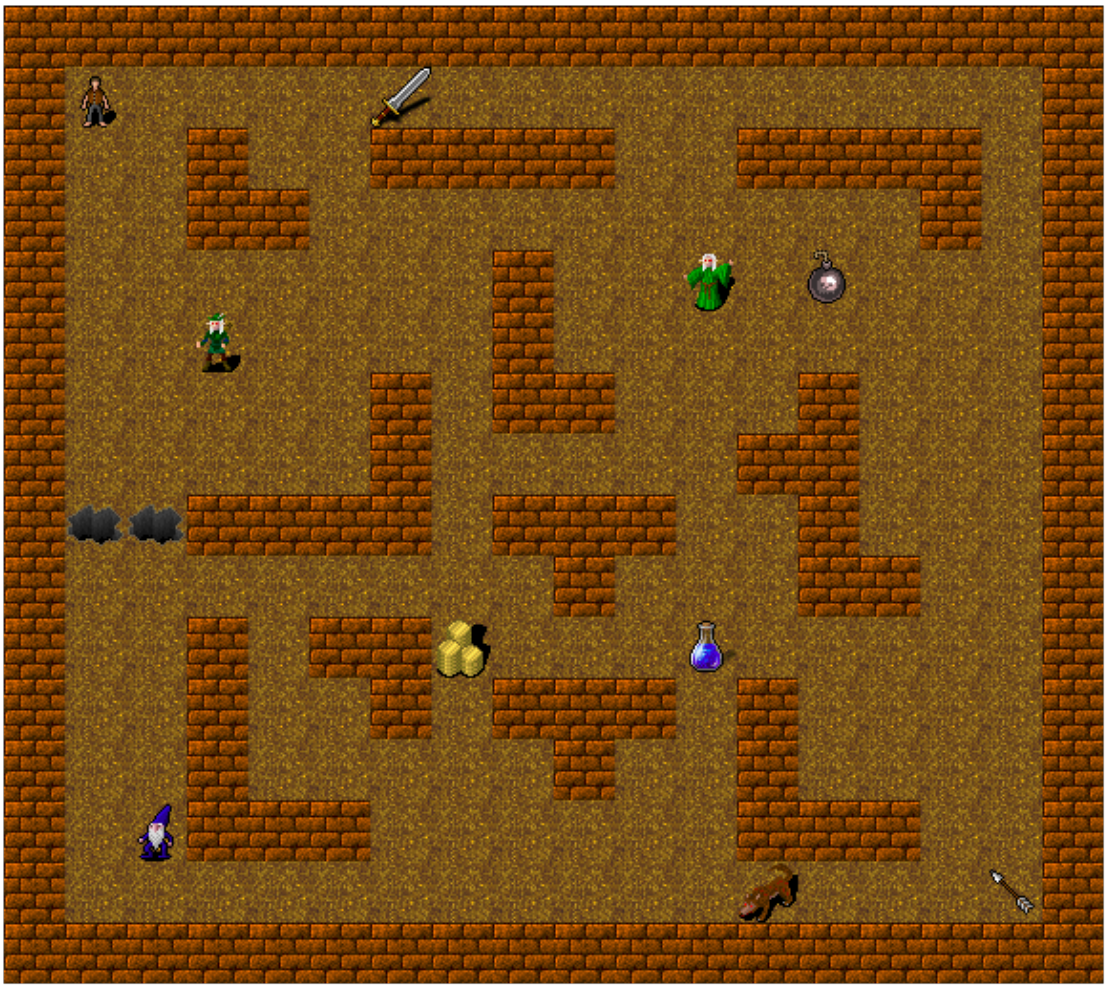
The client desires an application that works in two modes. In the first mode, the user moves a player around a dungeon and tries to overcome various challenges to "complete" the dungeon. The simplest form of such a puzzle is a maze, where the player must find their way from the starting point to the exit.



More advanced puzzles may contain things like boulders that need to be pushed onto floor switches,



or enemies that need to be fought with weapons, potions, or treasure.



In the second mode, the user designs their own dungeons. They should be able to place various walls, obstacles, enemies and other items in the dungeon to build puzzles. The client wants this to be tightly integrated with the first mode, so it should be possible, for example, for the user to try out an incomplete dungeon to help them while designing.

Dungeon layout

To be specific, each dungeon is defined by a grid of squares, each of which may contain one or more entities. The different types of entities are as follows:

Entity	Example	Description
Player		The player can be moved either up, down, left, or right into adjacent squares, as long as another entity does not stop them (e.g. a

Entity	Example	Description
		wall).
Wall		Blocks the movement of the player, enemies, boulders and arrows.
Exit		If the player goes through an exit the puzzle is complete.
Treasure		Can be collected by the player.
Door		Exists in conjunction with a single key that can open it. If the player holds the key, they can open the door by moving through it. After opening it remains open. The client will be satisfied if dungeons can be made with up to 3 doors.
Key		Can be picked up by the player when they move into the containing square. The player can carry only one key at a time, and only one door has a lock that fits the key. The key disappears once it is used to open its corresponding door.
Boulder		Acts like a wall in most cases. The only differences are that it can be pushed by the player into adjacent squares and can be destroyed by a bomb. The player is only strong enough to push **one** boulder at a time.
Floor switch		Switches behave like empty squares so other entities can appear on top of them. When a boulder is pushed onto a floor switch, it is triggered. Pushing a boulder off the floor switch untriggers it. See below for how switch triggering affects dungeon

Entity	Example	Description
		completion.
Unlit bomb		The bomb is picked up by the player when they move into the square containing it.
Lit bomb		The player can light the bombs they have collected and drop them. The fuse burns down for a short fixed period of time before the bomb explodes. Upon explosion, any boulders or enemies in the squares immediately to the left, right, above or below are destroyed. If the player is in one of these squares they die.
Pit		If the player falls into a pit they die. Boulders pushed into a pit disappear.
The hunter		This enemy constantly moves toward the player, stopping if it cannot move any closer. Like all enemies, the player dies upon collision with them.
The strategist		An enemy that moves towards a square the player is likely to move to next.
The hound		An enemy that assists the hunter by positioning itself such that the player is between it and the hunter.
The coward		An enemy that behaves like the hunter when far away from the player, but runs away when it gets close. Like all enemies, the player dies upon collision with them.
Sword		This can be picked up the player and used to kill enemies. Only one sword can be carried at once. Each sword is only capable of 5 hits and disappears after that. One hit of the

Entity	Example	Description
		sword is sufficient to destroy any enemy.
Arrow		Can be collected by the player and at will either left, right, up or down. Enemies are destroyed if they are hit with an arrow. Arrows are destroyed upon collision with anything. There is no limit to how many arrows the player can carry.
Invincibility potion		If the player picks this up they become invincible to all bombs and enemies. Colliding with an enemy should result in their immediate destruction. Because of this, all enemies will run away from the player when they are invincible. The effect of the potion only lasts a limited time.
Hover potion		Gives the player the ability to hover and fly over pits. This potion lasts until either the player is destroyed or the dungeon is complete.

Dungeon completion

Dungeons can be completed in different ways, depending on what entities are in them and choices made by the designer.

If there are exits, then the puzzle can only be completed by the player entering one of them. However, if there are no exits, then the dungeon can only be completed by some conjunction of the following:

- Destroying all enemies.
- Having a boulder on all floor switches.
- Collecting all treasure.

In the dungeon designing mode, the user should be able to select one or more of these as completion conditions.

Visual design

The client has given you free reign over the visual design of the program. In the above table, there are some example assets you may use, but you do not have to. You can find assets elsewhere or even create your own. The examples above came from here.

Requirement analysis (Milestone 1)

For this initial milestone, you are to model the requirements of the client as user stories on a taskboard in GitHub. Use the "Basic kanban" template that GitHub provides. You will show these user stories to your tutor in the Week 6 lab, where they will give feedback.

Prior to your lab session, your team must schedule collaboration sessions, to have initial high-level visioning discussions during which you will brainstorm to identify epic stories from the client's requirements and their key features, break-down high-level user stories to smaller user-stories and detail each user story to identify key conditions of satisfaction. Each epic story and user story should have its own card. Your team is expected to produce:

1. High-level epic stories from the problem statement
2. Each epic story broken into user stories â€“ Each user-story must define:
    - a short description of the feature based on Role-Goal-Benefit template (Refer to the RGB model described in the lectures)
    - an estimate for the implementation of the user story in user story points (e.g. 4 points, where each point takes 2.5 hours).
    - priority of implementation
    - acceptance criteria for each user story as a checklist on the card (Refer to the 3 Câ€™s model described in the lectures)

As you progress through the rest of the project, you will keep your taskboard up to date, checking off tasks that have been completed and moving stories from "To do" into "In Progress" and finally into "Done". You may also wish to convert cards into "issues" so that you can attach labels or assign them to individual team members.

Domain modelling and backend implementation (Milestone 2)

Based on your requirements analysis and taking into account the feedback from your tutor, you will produce a domain model for the backend component of your project in the form of a UML class diagram.

For this milestone, you are also required to implement the backend and write JUnit tests to test its functionality.

In deciding on a design for your project, you are expected to apply at least 3 of the following design patterns:

- Builder
- Factory
- State
- Strategy
- Decorator

It is up to you where they are applied, but you will be expected to justify how and why you used them to your tutor as part of the marking process.

In addition to your implementation, you are also required to include rigorous JUnit tests that you have used to ensure your backend functions as expected.

In the week 9 lab, your tutor will ask you questions about your design and implementation and give feedback you can use as you continue to work on it. They will also get you to run your JUnit tests.

UI design and implementation (Final Milestone)

Applying the ideas and concepts from user-centric design and considering the usability heuristics covered in the week 9 lectures, you are to design and implement the user interface component of the application.

Use MVC for the general architecture of your application. The puzzle creator and the interactive puzzle player should be separate views, though they will

share some common components. You will be using JavaFX to implement the UI. **You cannot use any 3rd party graphics or UI libraries. (e.g. OpenGL)**

This final milestone will be a culmination of all the work done in the previous milestones. You have the opportunity to improve upon your design based on feedback from your tutor as well as extend it with your own ideas. Note that to get high marks for extensions you will need to consider how they impact the user. Extensions that are technically complex, but do provide the user with any real benefit are not considered good extensions. You can, and should, create additional user stories to model the requirements of these extensions.

Assessment

You will be assessed on your ability to apply what you have learnt in this course as well as your ability to work in a team and produce a significant piece of software.

In cases where the client has not been explicit in their requirements, you will need to make your own design decisions with your team. It is important that you communicate regularly with your team members as well as document your design to avoid confusion about the choices you have made. You will keep a list of assumptions you have made and submit them as part of your final submission. You will also be asked to justify these decisions when you demonstrate in week 13.

You are expected to use git appropriately by committing consistently and using feature branches to manage significant changes. While we do not specify a formal software process, it is *highly* recommended that you take what you have learnt from COMP1531 and apply it here, following the principles of XP. It is an important part of this project for you (as a team) to manage your own time. You need to start early and work consistently: do not be overly optimistic about what you can achieve (remember you are all doing other courses with other assignments). It is essential for you as a team to monitor your own progress and it is often necessary to revise the plan as you progress. The most basic requirement is to develop a minimum viable product. Extensions can be added according to the time available.

While it is up to you how to divide the work between you, all members must

take on the developer role and work consistently throughout the project. Each team member *generally* receives the same mark, but if the moodle peer assessment tool or the logs of your GitHub repository indicate an imbalance in the amount of work done, the total mark may be scaled to match actual contribution.

Submission

Milestone 1

You should have all your user stories entered into a project taskboard on your GitHub repository. Tutors will check the date on your last change to the board to ensure no changes were made after the deadline.

Milestone 2

The code you wish to submit for this milestone should be in a branch named `release`. You should not push to this branch after the deadline. Your tutor will check to make sure you have not done this. You may continue to push to other branches as you work on your project between the deadline and when you demonstrate to your tutor.

Your UML diagram should be a PDF file at the root of your repository named `design.pdf`.

Final Milestone

Whatever you have in the `release` branch of your repository at the time of the deadline for this milestone will be considered your final submission. Include the following deliverables:

- The complete working code for your application
- An up to date UML class diagram
- A list of assumptions you have made

Your code should be set up such that it can easily be imported into eclipse. The simplest way to do this is to make your repository the root of your eclipse

project.

You will demonstrate your application to your tutor in Week 13. You may be asked to justify your design decisions and explain how you worked as a team.

Peer assessment

To ensure fair allocation of marks, you are required to complete a form on Moodle where you specify how much each team member contributed to the project and any comments you have about them. Information on how you can access this form will be released closer to Week 12.

Marking criteria

Your project will be given a mark out of 100 and scaled down to 25% of the course mark.

The marks are allocated as follows:

- Milestone 1 (15%)
- Milestone 2 (25%)
- Final milestone (60%)

Below is a *rough* guide for how you will be assessed for each milestone.

Milestone 1

Criteria	Mark	
Stories	0-3	User and epic stories not in RGB format and/or vague or ambiguous
	4-6	User and epic stories in RGB format, but with unclear benefits, goals or acceptance criteria
	7-9	Acceptance criteria lacking specificity
	10-12	Unambiguous and clear user stories with concrete acceptance criteria

Criteria	Mark	
Planning	0-1	No or only some user stories have points or priorities
	2-3	User stories have appropriate story point values and priorities

Milestone 2

Criteria	Mark	
Completeness	0-2	No or largely incomplete backend
	3-5	Backend implements only a few entities
	6-7	Backend implements most of the entities
	8-10	Backend implements all (or almost all) of the entities
Testing	0	No JUnit tests
	1-2	JUnit tests for a few of the smaller classes
	3-4	JUnit tests for behaviour of a few entities
	5	JUnit tests for behaviour of all entities
Design	0-4	Messy design and diagrams and/or design inconsistent with code
	5-7	Clear design and diagrams with partial adherence to design principles and patterns
	7-10	Clear design and diagrams fully adhering to design principles and patterns and conforming to code

Final milestone

Criteria	Mark	
Completeness	0-2	No or largely incomplete project
	3-5	Dungeons can only be played or created with a few entities
	6-8	Dungeons can be played or created with most of the entities
	9-10	Dungeons can be played or created with all (or nearly all) of the entities
Correctness	0-3	Serious bugs/system crashes during demo
	4-7	Minor bugs/some strange behaviour, e.g. runs only on one platform
	8	No bugs seen during demo/testing
Versioning	0-1	No or little use of feature branches or consistent commits
	2	Consistent commits, merges, feature branches, meaningful commit messages and names to feature branches
Design	0-5	Messy design and diagrams and/or design inconsistent with code
	6-10	Messy diagrams and/or poor application of design patterns
	11-15	Clear design and diagrams with partial adherence to design principles and/or poor application of design patterns
	16-20	Clear design and diagrams fully adhering to design principles and conforming to code, and correct application of design patterns
Interaction	0-2	Very basic user interface or not using MVC

Criteria	Mark	
	3-5	Interface that makes it possible to play and design dungeons, but is slow or awkward.
	6-7	Interface that is easy to use.
	8-10	A product that is engaging, intuitive and fun to use.
Extensions	0-2	No extensions or only very basic extensions
	3-4	One extension that represents some technical consideration
	5-7	One extension that represents some technical as well as user interaction consideration
	8-10	Multiple extensions that represent some technical as well as user interaction consideration