Assignment 1: Project Inception

Star Wars Episode IV: A New Project

For the remainder of the semester, you will be working in pairs on a relatively large software project. You'll be designing and implementing new functionality to add to an existing system that we will provide to you.

To help you manage your time on this project, we've divided it into three phases which will be assessed separately. In this phase, you'll take care of project inception: you'll set up your toolchain (git, GitHub, your IDE, etc.), decide on how to split up the tasks for the next assignment, and most importantly, make some preliminary designs for the extra functionality you'll implement in the next phase (Assignment 2). You'll be extending the system again in Assignment 3, so do the best you can to keep the system extensible and maintainable.

Getting ready

This is a large project, and you'll need a way to share files with your partner and unit staff. Instead of requiring you to submit your work on Moodle, we will provide you with a private GitHub repository that we can also read. There are a number of advantages to doing it this way:

- It provides you with a mechanism for sharing files with your partner. You'll be able to access your GitHub repository from Monash and from home – anywhere you've got internet access. Note that `git` support is built into Eclipse Neon, which has now been deployed across all our labs.
- It uses a fully-featured modern version control system – `git` – that is widely used for both industrial and hobbyist software projects.
- You don't need to submit. Instead, we'll check out your master branch at the due date (or later if you ask us to) – no need for you to do anything special.
- Your changes are automatically tracked. If you and your partner have a dispute about sharing the workload, unit staff can easily see whether you're adhering to your agreed tasks and timelines.

In order to get your repository created, we'll need you to register a GitHub account and let us know what it is.

Step 0: Temporary read-only access at Github

It will take us a couple of days to create your private repositories, so to let you get started immediately, we have made the code that we will put in your private repository available on Github in a public repository at `https://github.com/rgmerk/FIT2099-Assignment-1`.

You can view the code (and the design documents) on the Github website. If you want to run the code, you will need to figure out how to clone the repository (using eGit or the git tools in your IDE of choice), compile, and run it.

This is just to get you started; you will be expected to use the private repository we're creating for you to store all your work for this and subsequent assignments in FIT2099.

Step 1: Register a GitHub account

If you have not already done so, you will need to sign up for an account on GitHub. Go to `http://github.com` and fill in the signup form at the top of the page.

You do not have to use your Monash email address when you sign up, but if you do, you can sign up to receive GitHub's Student Developer pack. This gives you unlimited free private GitHub repositories and access to a lot of free[1] software and training resources from companies such as Microsoft, Amazon, and Udacity. If you enjoy hands-on learning, you'll find instructions for receiving the Student Developer Pack at `http://education.github.com/pack`.

Make a note of your GitHub ID once it has been created and verified. You'll need it for the next step.

Step 2: Find a partner in your lab and sign up on Moodle

A form has been posted on Moodle. Fill in your name, Monash email address, GitHub ID and the name, Monash email address, and GitHub ID of your partner if you've arranged for one[2]. Optionally, you may also choose a name for your team's repository.

This assignment has been designed to be done in pairs, and we will not allow you to work on your own. If you don't arrange a partner yourself, the teaching staff at your campus will allocate one to you. If there's an odd number of students in your lab, your lab may have *one* team of three – the teaching staff at your campus will decide which team.

Once we've received your details, we will create a repository for you and populate it with your starting source code and design documentation for the existing system. When this has been done, it's time to start working on the assignment proper.

Stage 1

You will be working on a text-based "roguelike" game set in the Star Wars universe. Roguelike games are named after the first such program: a fantasy game named (obviously) `rogue`. They were very popular in the days before home computers were capable of elaborate graphics, and still have a small but dedicated following. If you'd like more information about roguelike games, a good site is `http://www.roguebasin.com/`.

If you are not familiar with Star Wars, you may wish to look it up on IMDB. The first Star Wars movie to be produced was Episode IV: A New Hope (`http://www.imdb.com/title/tt0076759/`). A very comprehensive fan site is Wookieepedia, which you will find at `http://starwars.wikia.com/wiki/Star_Wars`.

As it stands, the game functions but is missing a lot of desired functionality. Over the course of this project, you'll be incrementally adding new features to the game.

For this assignment, we don't expect you to write any code yet. Instead, you must produce preliminary *design documentation* to explain how you're going to go about adding the specified new functionality to the system. The new functionality is listed below in the section headed **Project Requirements**.

We expect that you will produce *class diagrams* for all of the new features. Your class diagrams don't have to show the entire system, only the new parts, the parts that you expect to change, and enough information to let readers know where your new classes fit into the existing system. As it is likely that the precise names and signatures of methods will be refactored during development, you don't have to put them in the class diagram. However, the overall responsibilities of the class need to be documented *somehwere*, as you'll need this information to be able to begin implementation.

[1] Free as in beer, not free as in speech.

[2] Unfortunately, we can't accommodate you if you and your friend are in different lab classes as in that case it's harder for the demonstrators to know who they're supposed to mark.

To help us understand how your system will work, you must also write a *design rationale* to explain the choices you made. You must explain both how your proposed system will work and why you chose to do it that way.

You should use *sequence diagrams* (or collaboration diagrams) to help explain complicated interactions within your system. You will need to produce one for Force-users' mind control at least, since this is likely to involve multiple interactions between objects, and you may produce others if you find that it helps you explain your design. You can also supplement these with other diagram types.

The design (which includes *all* the diagrams and text that you create) must clearly show:

- what classes exist in your extended system.
- what the role and responsibilities of any new or significantly modified classes are.
- how these classes relate to and interact with the existing system.
- how the (existing and new) classes will interact to deliver the required functionality.

You are not required to create new documentation for components of the existing system that you *don't* plan to change.

Your design documents may be created with any tool that you choose – including a pen and paper if you can do so legibly. However, please upload PDFs, JPEGs or PNG format images of your design because we don't guarantee that your marker will have the same tools available.

If you want a UML diagramming tool, UMLet (`http://www.umlet.com`) is a free, easy-to-use UML diagramming tool that is particularly suitable for beginners. The diagrams in the starting repository were created using Umlet, and the source is also located there.

As you work on your design, you must store it in your `git` repository. Read on.

Git and GitHub resources and policy

You are required to use `git` for version control. You will find introductory information on how `git` works at `https://git-scm.com/doc` – view the videos at the bottom of the page in order to help you understand how to manage your repository. We can also recommend the online tutorial at `https://try.github.io`. (Note: you will not have to create your own repository for this assignment. This will be created for you by the FIT2099 staff.)

Your code *and documents* will be stored on GitHub (`https://github.com/`). More information about GitHub can be found at `https://services.github.com/on-demand/`.

We will use your project's commit log to work out who's been contributing to the project and who has not. That means that it is *very important* that you commit and push your work frequently – *including your design documentation*. It's also good practice to do this as it makes it much less likely that you'll lose work to a crash or accidentally deleting it.

Storing project artifacts elsewhere and commiting them to your repository just before the due date is NOT acceptable and will be penalized as not complying with the submission instructions. It's also highly risky - laptops get lost, hard disks become corrupt, and cloud services go offline at crucial moments (if Github goes down, it's our problem, not yours).

Your team may adopt any policy you choose for branching and merging your repository; however, you'll be marked on your master branch. If you haven't integrated your changes to master by the due date and time, there's a strong chance that they won't be marked and you will receive zero marks for that functionality.

Work Breakdown Agreement

We also require you to submit a very simple Work Breakdown Agreement (WBA) to let us know

how you plan to divide the work between members of your team.

Your WBA must explain:

- who will be responsible for *producing* each deliverable (whether it's a part of the system, an internal document, or an externally- eliverable document),
- who will be responsible for *testing or reviewing* each deliverable, and
- the dates by which the deliverable, test, or review needs to be completed.

Both partners must accept this document. You can do this by adding your name to the WBA and pushing it to the server with a commit comment that says "I accept this Work Breakdown Agreement". (Rest assured that if the WBA is changed after this time, we will be able to tell!)

We don't require you to follow a template for the WBA. A simple `.doc` or `.txt` file that contains the information we need will be sufficient. This document isn't worth marks in itself, but it is a hurdle requirement: your submission won't be accepted if there is no WBS. We will use it to make sure that you're allocating work fairly, and to make sure that we can hold individuals responsible for their contributions to the team.

We will use this document for accountability. If a team member complains that their partner isn't contributing adequately to the project, or that their contributions aren't being made in time to allow for review or debugging, FIT2099 staff will look at the partner's contribution in the logs. If we decide that the complaint is justified, this will be taken into account in the marking: students who do not contribute will be penalized and may fail, while students whose partner is not contributing adequately may have their mark scaled so as not to count any functionality that is missing due to the actions of their teammate.

Project requirements

You have been provided with a large codebase for a text-based game set in the Star Wars universe. This game is based on an engine that has been used for several years for games based in many different pop-culture universes, but the Star Wars code that lies on top of it is new.

The FIT2099 teaching team have brainstormed a set of requirements that we would like you to add to the Star Wars codebase. These are not fully-formed requirements or user stories – they're copied directly from a whiteboard, in fact – and there is a fair bit of leeway in how they are interpreted.

If you're worried about the interpretation of these requirements, we encourage you to post a question on Moodle. Teaching staff will be quick to respond, and other students will also benefit from seeing the discussion. Of course, it's a good idea to look in the forum before you post in case somebody's already posted a similar question!

Note that you are only expected to *design a solution* that can support these requirements for this initial Assignment. You'll be implementing this solution (and refactoring it if necessary) in Assignment 2. Assignment 3 will bring a new set of brainstormed requirements for you to add to the system.

Force ability

Some people[3] (including Ben and Luke) have the ability to use the Force.

People with a little bit of Force ability can resist Jedi mind control powers.

People with a lot of Force ability can control the weak-minded (i.e. people who can't use the Force) – that is, they can force them to attempt to move, on their next turn, in a direction of the Force-user's choice.

[3] This is Star Wars, so the word "people" includes aliens.

Lightsabres
Anybody can pick a lightsabre up, but only people with a lot of Force ability can wield one and use it as a weapon.

Ben Kenobi
Ben can train Luke. Training Luke has the effect of raising his force ability to the extent that Luke can wield a lightsabre.

Droids
Droids can't use the Force.

Droids have owners.

Droids follow their owners if they are able to do so.

Droids lose health when they try to move in Badlands.

Droids regain health if they use oil, or if somebody else uses oil on them.

Droids don't die, but they become immobile when their health runs out.

Immobile droids can be disassembled into droid parts.

Some people know how to repair immobile droids. Immobile droids can be repaired by using droid parts on them, which uses up the droid parts.

Healing
Drinking from a canteen should heal the drinker a little bit.

Droids regain health if they use an oil can, or if somebody else uses an oil can on them. This should *not* deplete the oil completely; unlike the canteen, there's no way to refill the oil can. PDF

Infrastructure
The engine is the way it is for good reasons. **Don't modify the engine code!** Doing this will make it much less reusable.

Submission instructions

Please put your design documents and work breakdown agreement (in one of the acceptable file formats listed earlier) in the `design-docs` folder of your private GitHub repository.

The due date for this assignment is *Friday 28th April at 5PM*. This is five days later than it says in the unit guide, to allow for repositories to be created. We will mark your assignment 1 on the state of the "master" branch of your private GitHub repository at that time.

If you wish to submit late, email your lab demonstrator and let them know the time when your assignment 1 was considered ready by your team; we will mark according to the repository at that time. However, unless a team member has applied for and received special consideration according to the faculty's Special Consideration Policy[4], late submissions will be penalized at 10% per working day late.

It is both team members' responsibility to ensure that the correct versions of the documentation are present in the repository by the due date and time. Once both teammates have agreed on a

[4] `http://www.monash.edu/exams/changes/special-consideration`

final Assignment 1 submission, do not make further commits to the master branch of the repository until the due date has passed, without the agreement of your teammate. If you want to continue to make changes to the repository for some reason, make another branch.

Marking Criteria

This assignment will be marked on:

Design completeness Does your design support the functionality we have specified?

Design quality Does your design take into account the quality properties and heuristics we have discussed in lectures, including:

- Not Repeating Yourself.
- Encapsulation.
- Information hiding.
- Cohesion (modules within the system have a clear, easily described purpose).
- Dependency control - have you minimised dependencies between modules, and made them explicit and easy to understand?

Practicality Can your design be implemented as it is described in your submission?

Following the brief Does your design comply with the constraints we have place upon it - for instance, does your design leave the engine untouched, as required?

Documentation quality Does your design documentation clearly communicate your proposed changes to the system? This can include:

- UML notation consistency and appropriateness.
- consistency between artifacts.
- clarity of writing.
- level of detail (this should be sufficient but not overwhelming)