Assignment 1 specifications

Planning and Design: Design O' Souls

Due: ~~Friday, September 3~~ Tuesday, September 7 at 11:55 AM (extended)

Melbourne time (if enrolled at Clayton campus) or KL time (if enrolled in Malaysia campus)

Learning outcomes for this assignment

This assignment is intended to develop and assess the following unit learning outcomes:

LO1. Iteratively construct object-oriented designs for small to medium-size software systems, and describe these designs using standard software engineering notations including UML class diagrams (in conceptual and concrete forms), UML interaction diagrams and, if applicable, UML state diagrams;

L02. Evaluate the quality of object-oriented software designs, both in terms of meeting user requirements and in terms of good design principles, using appropriate domain vocabulary to do so;

LO5. Use software engineering tools, including UML drawing tools, integrated development environments, and revision control, to create, edit, and manage artifacts created during the development process.

To demonstrate your ability, you will be expected to:

- read and understand UML design documentation for an existing Java system
- propose a design for additional functionality for this system
- create UML class diagrams and interaction diagrams to document your design, using a UML drawing tool such as UMLet or Visual Paradigm – you are free to choose which one
- write a design rationale evaluating your proposed design and outlining some alternatives
- use git to manage your team's files and documents

The marking scheme for this assignment will reflect these expectations

For the rest of the semester, you will be working in teams on a relatively large software project. You will design and implement new functionality to add to an existing system that we will provide to you.

IMPORTANT: A document explaining the FIT2099 Assignment Rules has been uploaded to the Assessments section in Moodle. Please read it and make sure you understand it BEFORE you begin the project – you are expected to follow what it says and will almost certainly lose marks if you do not.

Getting Started

The initial codebase will be available in a repository that will be created for you on **git.infotech.monash.edu** by Week 5. In the meantime, design documents for the system are available for you under the Assessments section in Moodle. Download these and familiarise yours

with the design.

Aiming to simulate the real-world conditions you may experience in the industry, you **will be allocated to a design and development team**. We won't create groups with students from different lab classes as this makes it less clear who's supposed to mark you.

General background

You will be working on a text-based **"rogue-like"** game. Rogue-like games are named after the first such program: a fantasy game named 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 would like more information about rogue-like games, a good site is http://www.roguebasin.com/. The initial codebase will be available in the repository mentioned above. It includes a folder containing design documents for the system.

Lore of the Game

In this assignment, we are inspired by the Dark Souls III game. We may use several similar names (characters, items, locations) and concepts. The purpose of using an actual game's concepts is to help students understand the features that may require imagination. It also helps you to immerse yourself in the world. Lastly, we believe it can **bring fun** while working on the assignments. All of the linked game contents, videos, and images belong to the respective owners and are subject to copyright. We mainly use the concepts for educational purposes and provide credit to the original creators accordingly. We may also add, alter, reduce the original content and features to make them more suitable to the game engine, unit outcomes, and assignments' time frame.

In this game, you are a **Player** that has risen from the dead to **slay enemies, get strong, and kill bosses** (Lords of Cinders).

Image 1. The Design O' Souls final game map output.

Assignment 1 and 2 Requirements

Here are the features we would like you to add to the game in the first round (assignments 1 and 2).

NOTE: In the first and second assignments, you are not required to develop the game's ending. Over the course of this project, you will incrementally add new features, update old features, and finally complete a playable game. Please **READ** all the features THOROUGHLY (**COMPLETELY**) before you start anything! Always keep in mind that these assignments are focusing on **object-oriented DESIGN**, NOT on game design.

The "**GROUP OF 3**" tag is used mainly for a group that consists of 3 people. You must design and implement those features for Assignment 1 and 2 respectively. If you are in a group of two, those features become optional. You are more than welcome to include those additional features in your works.

We add the "**OPTIONAL**" tag to help you finish the assignment with fewer features but meet all of the assignment outcomes. These optional features will help you implement a complete game that is **fun and playable** for your professional portfolio.

Requirement 1: Player and Estus Flask.

You are a Player in the game (display character: "@"), and the name of this character is "Unkindled". When the game starts, the player has **200 maximum hit points** (**100 maximum hit points** for **GROUP OF 3**). You need to show the player's health/hit points in the console, such as "Unkindled (30/100)". Initially, the Player holds a Broadsword (@see Weapons). The Player also has a unique health potion inside the inventory that can be refilled if you rest. This health potion is called Estus Flask (Health Potion). It has three charges, and each charge will heal the Player with **40% of the maximum hit points**. The Player cannot drop the Estus Flask. You need to display the number of charges in the console (in the list of actions) whenever the Player wants to drink it. For instance, **"a: Unkindled drinks Estus Flask (2/3)"** where `a` is the hotkey to execute the corresponding action (i.e. drinking Estus Flask).

Requirement 2: Bonfire

Image 2. Firelink Shrine with Bonfire "B" in the middle and "F" for the vendor.

The Player starts at the Firelink Shrine (it will be the name of Bonfire). It is an area in the middle of the map that is surrounded by the walls (displayed as **"#",** the player or any other character cannot cross a wall), and its ground is made of Floors (one **Floor** is displayed as an underscore **"_"**). You can see a Bonfire (displayed as "B") sitting exactly in the middle. Only a Player can interact with Bonfire. At the moment, the Bonfire only has one action, and we call it **"Rest at Firelink Shrine bonfire"**.

When the Player **chooses to rest**, all of the following **`RESET` features** are executed (in one turn):

- **Refill** Player's health/hit points to the maximum.
- **Refill** Estus Flask to maximum charges.
- (**OPTIONAL**) **Reset** the enemies' position, health, and skills (*@see Enemies*): in this case, reposition them at the initial locations, and refill their hit points back to maximum. However, all Undead (Hollow Soldier) do not need this re-positioning. They will be removed/**wiped out** from the map immediately. You also need to remove its `following Player behaviour` (*@see Enemies*).

Implementation expectations:

When the game starts, the **Player should hold the Estus Flask.** There will be an option to consume it in the Player's actions list. You also need to show the number of charges of the Estus Flask. Then, the Player can move to Bonfire and interact with it (either adjacent or on top of it). The Player can see a new action to rest on the Bonfire. When enemies wander around, the enemies cannot enter the Firelink Shrine.

Requirement 3: Souls (a.k.a. Money)

Souls are the currency of this game, and the Player has 0 souls at the start of the game. When the Player **slays/kills enemies** (*@see Enemies*), the Player gains a certain number of souls from them.

(**GROUP OF 3**) The Player can trade collected souls to the Vendor (@see Vendor) for swapping weapons or upgrading the Player's attributes (character stats).

Requirement 4: Enemies

In this game, you will have several enemies. When an enemy notices that the Player is in its surrounding (adjacent squares), it will **follow** the Player and starts to attack Player whenever possible. For the sake of simplicity, once the enemies are killed, they are completely removed from the game (i.e., **no corpses** around). You need to show hitpoints, maximum hit points, and the weapon that it equips in the console history log.

(**OPTIONAL**) features:

- Enemies cannot enter the Firelink Shrine because they cannot enter the Floor.
- Enemies cannot attack each other.
- Enemies can randomly use active skills from a weapon.

1. Undead (Hollow soldier)

The Undead will spawn from the Cemetery (*@see Terrains*). An Undead starts with and has **50 maximum hit points,** and it cannot be equipped with any weapons. It walks around aimlessly and will attack the Player if it stands next to it (adjacent squares). When it detects a Player, it will follow the Player until the Player dies/rest at Bonfire. Its base attack points are **20 hit points** and print either 'thwacks' or 'punches'. When the Player kills it, the Player gains **50 souls**.

(**OPTIONAL**) When it is not under attack, or not following a Player, the undead has a **10% chance** to die instantly every turn, and they are immediately removed from the map. Dying this way **WILL NOT** increase the Player's souls level. *(NOTE: We merely use this feature to clean up the map).*

2. Skeleton

You need to place several Skeletons (4 to 12 skeletons) manually and anywhere in your game map. Please calibrate the number of skeletons as much as you need. A skeleton needs to know its initial location to reset its position when the world gets reset (*@see Bonfire*). Skeletons walk around and follow the Player (plus attack) if the Player is within its radius (i.e., adjacent squares).

This enemy starts with and has **100 maximum hit points**. The Skeleton carries one random weapon (i.e., Broadsword - or Giant Axe if **GROUP OF 3**, *@see Weapons*). The attack hit-rate point & damage will follow its weapon's description. When it truly dies, it gives **250 souls** to the Player.

(**GROUP OF 3**) When it dies for the first time, it can resurrect itself with a **50% success rate** that heals to maximum hit points. It **cannot revive more than once** (in other words, remove it from the game once it dies again).

3. Yhorm the Giant (Lord of Cinder)

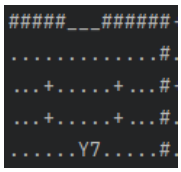

Image 3. Yhorm's chamber, where 'Y' is Yhorm the Giant, and '7' is the Storm Ruler

It is your first advanced enemy/boss in the game. It starts and has a maximum of **500 hit points**. It holds a Yhorm's Great Machete (*@see Weapons*). It will stand still in the room/hall that is provided in the base code. This boss is **weak to Storm Ruler** (*@see Weapons*). It has the following unique behaviour:

- **Ember Form (Second Phase):** When Yhorm the Giant's health reaches below half of its maximum hit points (**< 50% HP**), it **enrages** and its weapon becomes much more effective - its hit rate increases (*@see Weapons* for the detail). You must print a suitable message when Yhorm is entering this phase. **GROUP OF 3** will have an additional feature where the boss will burn the surrounding area (@see Weapons). It happens only once that is when Yhorm activates this ember form.

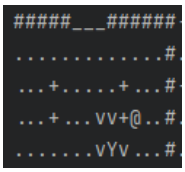

Image 4. Yhorm the Giant activates the Ember Form skill ('V' is a burning ground)

Other enemies cannot enter this room/hall due to floors that are placed at the entrance. When Yhorm the Giant **is killed**:

- **Print** an appropriate message in the console, such as "LORD OF CINDER FALLEN".
- **Give** 5000 souls to the Player
- (**OPTIONAL**) **Drop** "Cinders of a Lord" item: the Player can drop it in the Firelink Shrine to continue the story (for Assignment 3, now, it does nothing).

Reminder, Lord of Cinder **WILL NOT drop the weapon**. Yhorm also cannot be resurrected when the World gets reset.

Implementation expectations:

You must watch

FIT2099 - Boss Fight (short-ve...

Yhorm the Giant Boss fight:

- Walk from Firelink Shrine to Yhorm's chamber
- Pick up Storm Ruler (a sword)
- Fighting strategy (useful for playtesting in Assignment 2 later)
- Yhorm the Giant features

or

to understand the expectation of the boss (Yhorm the Giant) feature and the overall fight. Ideally, the Player equips Storm Ruler (*@see Weapons*), walks into the chamber, and fights Yhorm the Giant. That video might be helpful to explain the implementation of Storm Ruler's active skills (Charge & Wind Slash).

Requirement 5: Terrains (Valley and Cemetery)

There are many valleys and cemeteries.

- **Valley**: We provided several valleys in the map (with the display character **"+"**). When the player steps on it, **it instantly kills** the Player (you can imagine that the Player falls from the cliff and the Player receives a lot of damage) (*@see Soft Reset/Dying*). Other actors (allies or enemies) cannot step on it.

- **Cemetery**: Each cemetery has a **25% success rate** to spawn/create Undead (Hollow soldier)(*@see Enemies*) at every turn. Please place several cemeteries (at least five) in the game map to spawn a lot of undead in the game.

Requirement 6: Dying in this game/Soft reset

When the Player dies/gets killed, you need to print the classic Souls game phrase or any appropriate message in the console. You may optionally customise it with ASCII art to add a bit of fun. When it happens, all of the **RESET features are executed** (*@see Bonfire, those dot points when taking a rest*). Then, the game will **move**/respawn the **Player back to the latest Bonfire** that the Player has rested. At the moment, the game only has one Bonfire at the Firelink Shrine. Hence, the Player will be moved to that Firelink Shrine's Bonfire (standing on top of it).

***NOTE**: This is **NOT** the end game yet, and we don't want to see the game STOPS if the Player died (that's why it is called a soft reset).*

Image 5. A Player is one step behind to retrieve a token of soul that spawned at a previous dying location (near the valley)

Image 5: A Player is one step behind to retrieve a token of souls that spawned at a previous dying location (near the valley).

Additionally, when the Player dies, the Player's soul level will become 0. Then, a **"token of souls"** will appear at the dying spot/location. The Player can get back these lost souls only by interacting with that token of souls. This token of souls is shown as a **`$`** in the game. It stays there forever even though the World gets reset. When the Player stands next to (or on top) and interacts with it, the Player can **regain** its recent lost souls. Recovering souls this way will increment current souls that the Player has.

We have a specific scenario for **dying from falling** (when the Player steps on the Valley ground). If the Player falls into the valley (Player steps on the "+" in the game), that token of souls **must not be placed on the valley location** (even though it is the Player's dying spot). But, it will be placed one step behind. For example, Player's north direction is a Valley. The player steps on it and so the Player falls into the abyss. The token of souls will show up at the ground before the Player steps on that valley (i.e, at the south of the corresponding valley object).

(NOTE: We provide the feature above so that the Player can reach and recover their token of souls.)

(OPTIONAL for GROUP OF 3): Suppose the Player dies again somewhere in the map before retrieving it; that **token of souls** will be removed from the map and a new token of souls (with current Player's souls) will appear at the last dying location. The number of souls in that new token will always be the Player's soul before it dies. For example, the Player has 500 souls that just fell into the valley. The Player is respawned at the Firelink Shrine's Bonfire with 0 souls. The token of souls `$` is shown at that last dying location (note that it is on the last location ground that is not the valley). This token of souls is worth 500 souls now. Unfortunately, the Player with 0 souls got killed at a different location before retrieving those souls. A token of souls that contains 500 souls will be gone forever, and a new token with 0 souls will be placed on the map accordingly (at the latest dying spot). When the Player interacts with this token of souls, unfortunately, it increases 0 souls to the current Player's souls (equivalent to nothing). Optionally, when the Player rests at the bonfire, the token of souls should stay on the map.

Implementation expectations:

There are many ways to **test the dying feature** in the game:

- **Fall** from the cliff (@see Valley): You can step on the Valley's object to "fall".
- **Killed** by the enemies (@see Enemies): You can attack enemies and try to get killed by them.
- **(OPTIONAL) Burned** by the fire (@see Enemies/Weapons): Fight with Yhorm the Giant.
- Etc.

Please ensure that the current Player's souls should drop to 0, and the token of souls is created at the dying location. Next, navigate the Player to that new symbol, execute its action, and increment the lost souls back. You may watch this

to understand how the soft reset/dying mechanism works if you find it is still confusing (important for GROUP OF 3). Otherwise, see the following GIF

Requirement 7: Weapons

The Player can only bring **one weapon at a time**. The Player and enemies **cannot drop their weapons** when they die or intentionally through an action. A weapon has active and/or passive skills.

HINT: in the implementation/Assignment 2, it means that you must be brave in **refactoring** the provided code (in the `game` package, NOT the **engine!**).

Please go to Appendix A at the end of this page for further weapons details.

Requirement 8: Vendor (*GROUP OF 3*)

The souls can be traded to buy a new **weapon** and to upgrade the Player's attributes (stats) through a vendor. We may give this vendor a name: "Fire Keeper". You need to place this vendor somewhere in the **Firelink Shrine** and set it to stay at its location permanently. When the Player buys a new weapon, the weapon in the current inventory will be automatically replaced with it. Replacing the weapon will cause the old weapon to be removed from the game**.** You need to print an appropriate message when the transaction is successful or fails.

Name	Price (souls)	Usage	
Broadsword	*(@see **Weapons**)*	*(@see **Weapons**)*	
Giant Axe	*(@see **Weapons**)*	*(@see **Weapons**)*	
(OPTIONAL) **Increase Maximum HP**	200	Increase Player's maximum hit points by 25	

Table 1. List of products & services

The expectations for game expansion (Assignment 3)

We cannot reveal too much information about future expansion (Assignment 3 features) because it will fixate your design thinking. As a software engineer, you need to think ahead for uncertain features. Therefore, we suggest you always follow the best practices to maintain your system/software**.** However, the upcoming expansion will have several general features for sure: more Lords of Cinder (bosses), more unique enemies, more unique grounds, more items, etc.

What to submit for Assignment 1?

You are expected to produce the following three design artefacts in Assignment 1:

- Class diagrams (distinguish between existing classes and new classes with colours)
- Interaction diagrams (sequence diagrams or communication diagrams)
- A design rationale

You will implement your design later, but for Assignment 1, you are not required to write any code. Instead, you must produce preliminary *design documentation* to explain how you will add the specified new functionality to the system. The new functionality is specified in the **Project Requirements** section.

We expect you to produce *UML class diagrams* and *UML interaction diagrams* (i.e. sequence diagrams or communication diagrams) following **the FIT2099 Assignment Rules**. These Rules are available on Moodle.

Your class diagrams do not have to show the entire system. You only need to show the new parts, the parts 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 do not have to put them in this class diagram. However, the overall responsibilities of the class need to be documented *somewhere*, as you will need this information to begin implementation. This can be done in a separate text document if you prefer.

To help us understand how your system will work, you must also write a *design rationale* to explain your choices. You must demonstrate both how your proposed system will work and ***why* you chose to do it that way**. You may consider using **the pros and cons** of the design to justify your argument.

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

- what classes will exist in your extended system
- what the roles and responsibilities of any new or significantly modified classes are

- what the roles 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 *do not* plan to change.

Work Breakdown Agreement

We require you to create a simple Work Breakdown Agreement (WBA) to let us know how you plan to divide the work between your team members. There is more information on **WBAs in the FIT2099 Assignment Rules** (in Moodle). In this matter, everyone must contribute a **FAIR amount of code AND documentation**. It means you cannot work only on the code or only on documentation.

Submission instructions

You must put your design documents and work breakdown agreement (in **PDF** format) in the design-docs folder of your Monash GitLab repository.

The due date for this assignment is at the top of the first page. We will mark a snapshot of your repository as it was at the due time. This means that you will need to notify your marker if you finished late and let them know which version they should check out.

Unless a team member has applied for and received special consideration according to the Monash Special Consideration Policy,[1] late submissions will be penalised at **10%** per working day late. Details about special considerations can be found in the Unit Information section in Moodle.

It is ALL team members' responsibility to ensure that the correct versions of the documentation are present in the repository by the due date and time. Once all teammates have agreed on a final Assignment 1 submission, do not make further commits to the master branch of the repository until the due date has passed unless your teammate agrees. If you want to continue to make changes to the repository for some reason, create 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 programming and design principles we have discussed in lectures? Look in lecture slides, and check the Object-Oriented Fundamentals documents for principles like

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 placed 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 correctness, appropriateness, and consistency
- consistency between artefacts
- clarity of writing
- level of detail (this should be sufficient but not overwhelming)

Explanation. Can you adequately explain your design and the reasoning behind it, both in your rationale and in response to interview questions from your marker? Here are the marking criteria:

1. Clearly communicated the information – using suitable language, tone and pace.

2. Used questioning techniques to encourage views and opinions.

3. Used active listening to confirm understanding.

4. Provide possible suggestions and comments for improvement.

[1] https://www.monash.edu/exams/changes/special-consideration

Appendix A: Weapons Table

Name Name	Price Price (Souls) (Souls)	Damage Damage (HP) (HP)	Success Success hit rate hit ra

Broadsword **(Sword)**	500	30	80%
Critical Strike (*Passive*): It has a 20% success rate to deal double damage with a normal attack. *Displaying its effect is optional.*			
Giant Axe **(Axe)** **(*GROUP OF 3*)**	1000	50	80%
Spin attack (*Active*): It gives a holder an action to swing the weapon. When it is swung, any enemies that stand in adjacent squares of the holder will receive 50% of this weapon damage (i.e., 25 damage). You may print a similar message as the normal attack. The holder can activate this skill anytime, even though there are no enemies.			
Storm Ruler **(Sword)**	2000	70	60%
Only the Player can* equip *this weapon. Place this sword next to Yhorm the Giant on the map. **Critical Strike (*Passive*): It has a 20% success rate to deal double damage with a normal attack. *Displaying its effect is optional.*** ***Dullness (Passive)* (OPTIONAL): attacking enemies that are not weak to Storm Ruler will decrease its effectiveness. The damage is reduced by half (i.e., it becomes 35), but the hit rate remains 60%.** **Actions:** **1. Charge *(Active)*: An action where the holder can charge the weapon for *three (3) turns* to unleash a wind slash (*Wind Slash (Active), see below*). The Player can charge this weapon anytime. When charging, the Player will be disarmed/cannot attack anyone.** **2. Wind Slash *(Active)*: When Storm Ruler is fully charged, remove Charge action, and make Wind Slash action available. It can only be executed when the holder stands next to Yhorm the Giant (in adjacent). When it is executed, it deals 2x damage 'wind slash' to *Yhorm* with a 100% hit rate and stuns it. A stunned target cannot move for one turn (i.e., do nothing). After executing this action, the weapon will have Charge skill again.**			
Yhorm's Great Machete **(Axe)**	N/A (not applicable)	95	60%
***Only *Yhorm the Giant* can equip this weapon.** **Rage mode (*Passive*): While the holder is in rage mode/ember form, It increases the holder's success hit rate by 30% (in other words, the hit rate: 60% + 30% = 90%).** **(*GROUP OF 3*) Burn Ground (*Active*): Additionally, when Ember Form is activated, use this skill to burn the surrounding squares (use different characters to display them on the map). The burned areas will stay on the map for the next three (3) turns and hurt anyone (except the holder) that stands on it by 25 hit points. The fire can only burn the Dirt ground.**			

Last modified: Saturday, 18 September 2021, 2:11 PM

◄ Assignment 1 & 2 - Supplementary Videos

Jump to...

Assignment 1 protocol and rubric ►