FIT2099 Design Rationale Assignment 3

This design rationale differs from the previous design rationale in that there are new changes and new requirements.

Fixed requirements

Requirement 1: New Map and Fog Door

Requirement 2: Updated Bonfire

1. Implementation of new Anor Londo's Bonfire

I put the new Anor Londo Bonfire at a location in the 2nd map and initially it is unlit/not activated, the only action available when the player encounters it, is to light the Bonfire.

Once the bonfire is lit, Anor Londo Bonfire is added to a list of other existing bonfires in the class Bonfire. Bonfire class just stores an array list of all lit bonfires available to the player and has the functionality to add bonfires to the list as well. The Player class creates an attribute Bonfire() hence it is linked to the player and other classes are able to access the list of bonfires if they have a Player attribute. Anor Londo Bonfire also has the same action resetting action as the Fire Link Shrine, hence I have used the same class BonfireResetAction for Anor Londo Bonfire. However I had modified BonfireResetAction to cater for any Bonfire instead of initially the Firelink Shrine only by having a string parameter indicating which bonfire the player is currently resting at. I decided to use the same classes for the resting class because in the future if there are many more bonfires that require a resting action, it just reuses the same one. Hence not repeating unnecessary code.

2. Teleportation of Bonfires/Dying Respawn

Requirement 3: Aldrich the Devourer (Lord of Cinder)

For implementing this new Lord of Cinder. I almost copy from the last one "Yhorm", but this one has it own attributes, it will hold a Longbow when it has been generated, for this long ranged weapon it will check 7*7 radius near the Giant, and after Devouer died it will drop a Cinder of Devourer on the map which extends Item and will be checked in vendor, if it is contained in player's inventory, it can be traded to Longbow weapon.

For Player to using the Longbow weapon, it will be checked if the player is holding it in playTurn, if yes, it will search all enemies in 7*7 radius, and the direction message will become the coordinates of enemies location to avoid multiple enemies being in the same direction.

But there is a bug here, for engine design, the actions menu only able to show 26 actions, but when a player is using Longbow and if there are enough enemies, the actions list will be much more than 26, and I don't know how to fix this since we can't change anything in game engine.

New Class: Devouer, Longbow

Structured Mode

Requirement 4: Mimic/Chest

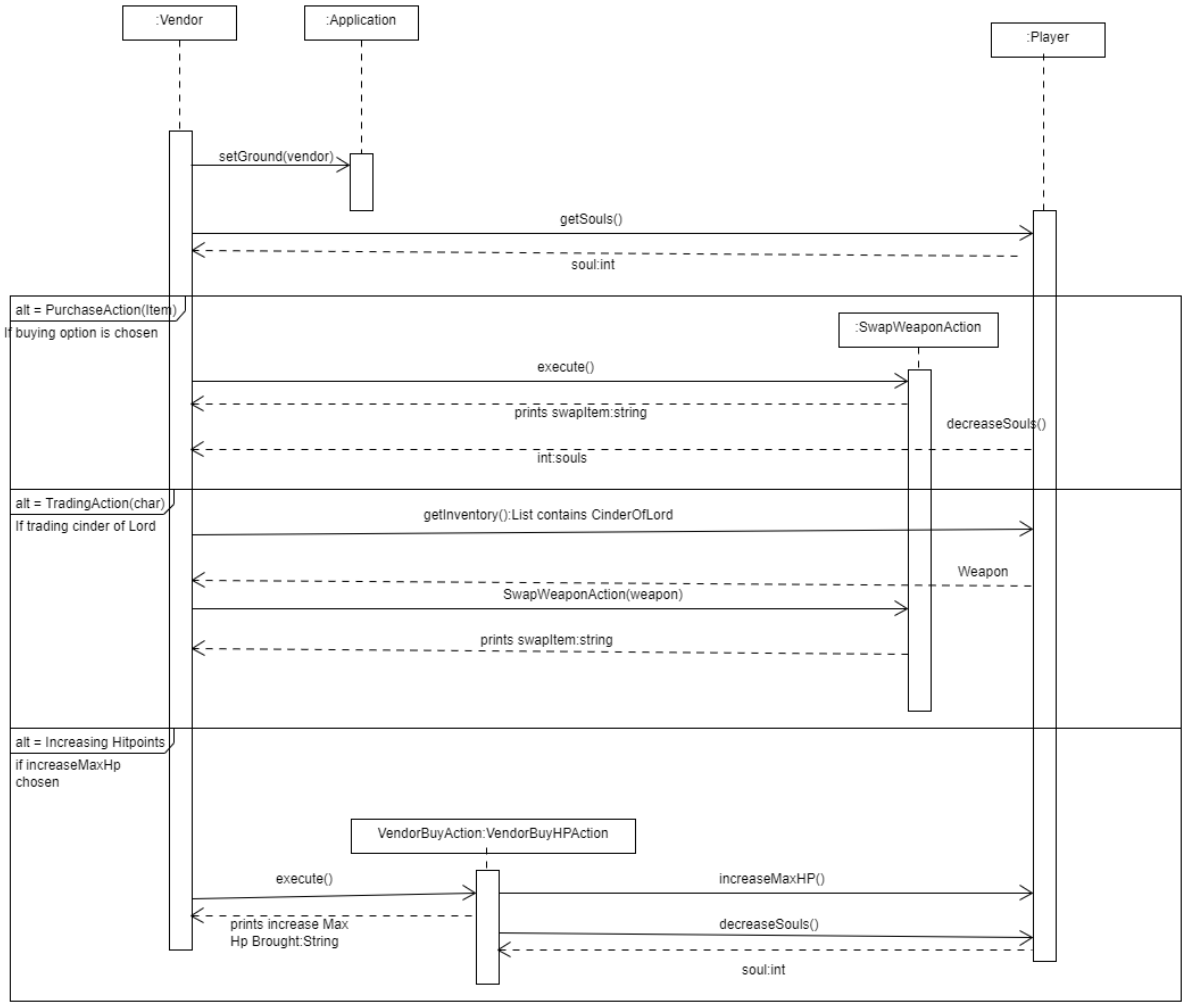
For implementing the chest, I made an Ambiguous class for it. It will override get allowable action, when a player near it, Open the chest action will be displayed in the action menu. After this action player will get a description, one is "Unfortunately, it becomes a Mimi" another one is "You are lucky, it becomes a token of souls" and will mention how many souls for this token, it has 100,200 and 300 souls randomly.

For implementing the Mimi, it extends Actor class and overrides the hurt method which will check if it died, drop the token of souls as the function above. It also has a different way to attack, so I made a kick attack action class which changes the description of the attack verb, and calls it in playTurn of Mimi.
New Class: Ambiguous, Mimi, KickAttackAction

Requirement 5: Trade Cinder of Lord

I implemented trading of the Cinder of Lords by having a PurchaseAction and TradingAction that Vendor calls.
PurchaseAction(Item) uses a parameter,item, that the player wants to buy/swap weapons with. It checks if the player has enough souls and deducts the souls from the player if the number of souls that the player has is equal or higher than the cost of the weapon in souls. This requirement was done in Assignment 2, however I made the specific buying classes ie. VendorBuyBroadswordAction etc into 1 class, PurchaseAction. Hence for future references we can just input the item that we want to buy instead of having individual classes for each purchase item. This follows the principle of not repeating code and reusing the same codes as much as possible. It also decreases dependency compared to the initial implementation from Assignment 2, as now it no longer depends on a parent class VendorBuyAction.
TradingAction(character) is used to trade cinder of lord items after each lord of cinder is defeated. The trading class checks what the parameter is, (used to indicate what Cinder of Lord to be traded with). This class specifies what the cinder of lord and the name of the lord will be, hence it is not really ideal for future references or a large number of lords of cinders. This is because we would have to create attributes of all the cinder of lords for every existing lord of cinder in the game.

Updated Vendor Diagram



Class Diagram