CPSC 2720 Fall 2021

The Heist

Team Tali

Kevin Masson, Cale Koi, Alex Hughes, Musaddique Khondoker Mohammed

November 6 2021

Table of Contents

Introduction

The player plays as Francis Bigglesworth, art collector and world class thief extraordinaire. Your current caper involves the renowned art museum: *Le Mauvais Gout*. Inside this museum holds the great art piece known worldwide, the *Roly Poly Formuoli*. In order to pull off this heist, Francis is going to need to use all his cunning to disable the museum's defenses, overcome obstacles, and get past each guard using his wits and the help of those around him.

Project Management

Team Roles

Team Member	**Design - Draft**	**Design – Final**	**Implementation - Basic**	**Implementation - Final**
Kevin	Reporting Lead	QA Lead	Phase Lead	Design Lead
Cale	Phase Lead	Reporting Lead	Design Lead	QA Lead
Alex	QA Lead	Design Lead	Reporting Lead	Phase Lead
Musaddique	Design Lead	Phase Lead	QA Lead	Reporting Lead

Team Role Responsibilities

Phase Lead:

- Checks in with other group members to ensure progress on project and communicate issues sooner than later
- Fields questions to instructors as requested by group members

- Holds the “vision” of what the project aims to have accomplished by the end of phase
- Proposes and leads team meetings
- Acts as the intermediary between group members if needed

Design Lead

- Checks design during phase to ensure it meets design principles (SOLID + DRY)
- Idealizes and proposes new design ideas
- Creates and updates UML diagrams to reflect ideas
- Brings design issues to team leader to address sooner than later

Quality Assurance Lead

- Builds test cases to check for proper functionality
- Proposes fixes and additions to program to create smoother and more functional end product
- Creates bug reports and addresses bug reports
- Updates other members or leader about bugs to be dealt with sooner than later

Reporting Lead

- In charge of updating the report to reflect our project direction through the phase
- In charge of the report being done to a satisfactory point for the reporting lead
- Makes sure the team is designing in accordance to the report document
- Records meeting notes and outlines the plan going forward for each meeting

Risk Management

Requirements/Design/Estimation

- If the project becomes too large and cumbersome to meet the project deadline
    - Decide which features are causing the problems and if they are able to be cut or downsized
    - Decide what features are necessary and what are just nice to have
- If the team underestimates how long parts will take to create

    - Consider whether the time is due to design being too complicated or will just require the extra time
    - Consider if the steps from the project being too large is needed.
    - Decide if the job can be split between different team members or if it is a single member job
- If major changes to design are needed during implementation
    - Identify how the change will change the scope of the project if the previous items (too large, taking longer than expected) come into effect
    - Set a time table for how long it should take to change
    - Consider the effect changing the design will have on the rest of the project

People

- Addition or loss of team member
    - If a team member leaves then we will determine what their workload prior to leaving was and if it changes the scope of the project, and if so consider the steps in the **requirements/design/estimation** section
    - If a team member leaves we will determine who needs to take up their people responsibilities or if it is something that can be marginally done by other roles
    - If a team member is added, the phase leader will meet with them on Discord and outline the design plan and what is done so far and what is the next thing to be added. Will show the new team member the UML and what they should expect to do. Phase leader will then meet up with them after a deadline to check and see if their work aligns with the overall project vision.
- Unproductive team members
    - If a team member that has responsibilities is not meeting deadlines or not meeting quality standards the phase leader will re-evaluate what that team member can productively do and give them responsibilities that meet their ability
    - If the team member does not intend to work on the project, phase leader will e-mail Dr. Anvik and consider it as a **loss of team member**
- Team members lacking expected technical background
    - Team will outline work that is not as technical for the team member to work on (such as narrative or flavour text) or other simple designs that meet their abilities
    - Phase leader will check on progress to ensure that the new responsibilities are within the skill level of team member

    - The lacking team member is expected to update through bug report or to phase leader asap whenever a problem occurs that they do not know how to solve and may impact the progress of other members' work
- Major Life Events
    - If a team member has a life event that they are unable to work on the project in the manner they expected to, phase leader will meet with them to check how long they will not be able to work on their work or if smaller workload is able to be done
    - If the member is unable to work for an extended period of time, consider the **loss of team member** item

Learning and Tools

- Inexperience with new tools
    - Phase leader will connect with members who are confident in the tool to meet and show them how to use it properly if it is doable within the time constraints
    - If a team member is unsure of how to use tools that are needed to progress their workload and there isn't time to teach the member then they will report to the phase lead which will communicate with other members if they are able to take a break from their workload and complete using those tools and give the other member some of their work that is within their ability
- Learning curve with tools steeper than expected
    - If the tools are harder than expected, design lead will consider if the tool is needed to complete the project. If it is then refer to **requirements/design/estimates** portion
- Tools don't work together in an integrated way
    - Refer to **learning curve steeper than expected**

Development Process

Code Review Process

Team members will post ideas for a design change in the discord. The Design Leader will review the idea and discuss it with the team when we meet next. All changes that are done to bring the code into line with the project design will be posted in the discord and approved by the Team Leader. Any changes that seem unviable will be rejected by the Design Leader.

Communication Tools

We are using a discord server as our main method of communication. The server has separate channels for all aspects of the project to enable organization. Team members will check in on discord frequently to keep up to date with the progress of the project and for voice calls to review reports.

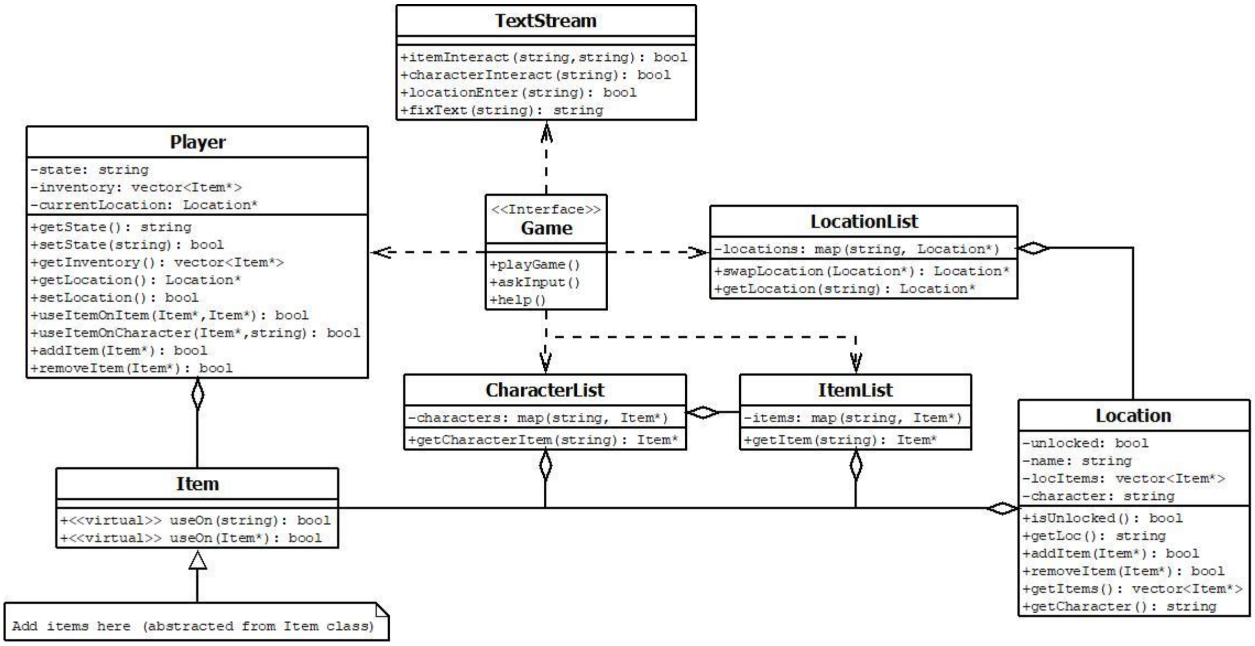
Change Management

The Quality Assurance Lead will review all bug reports. If the report is unclear, the Quality Assurance Lead will discuss it with the submitter. The bug report will then be assigned to whoever coded the buggy function. If a solution cannot be found by them then the Quality Assurance Lead will assist in fixing it.

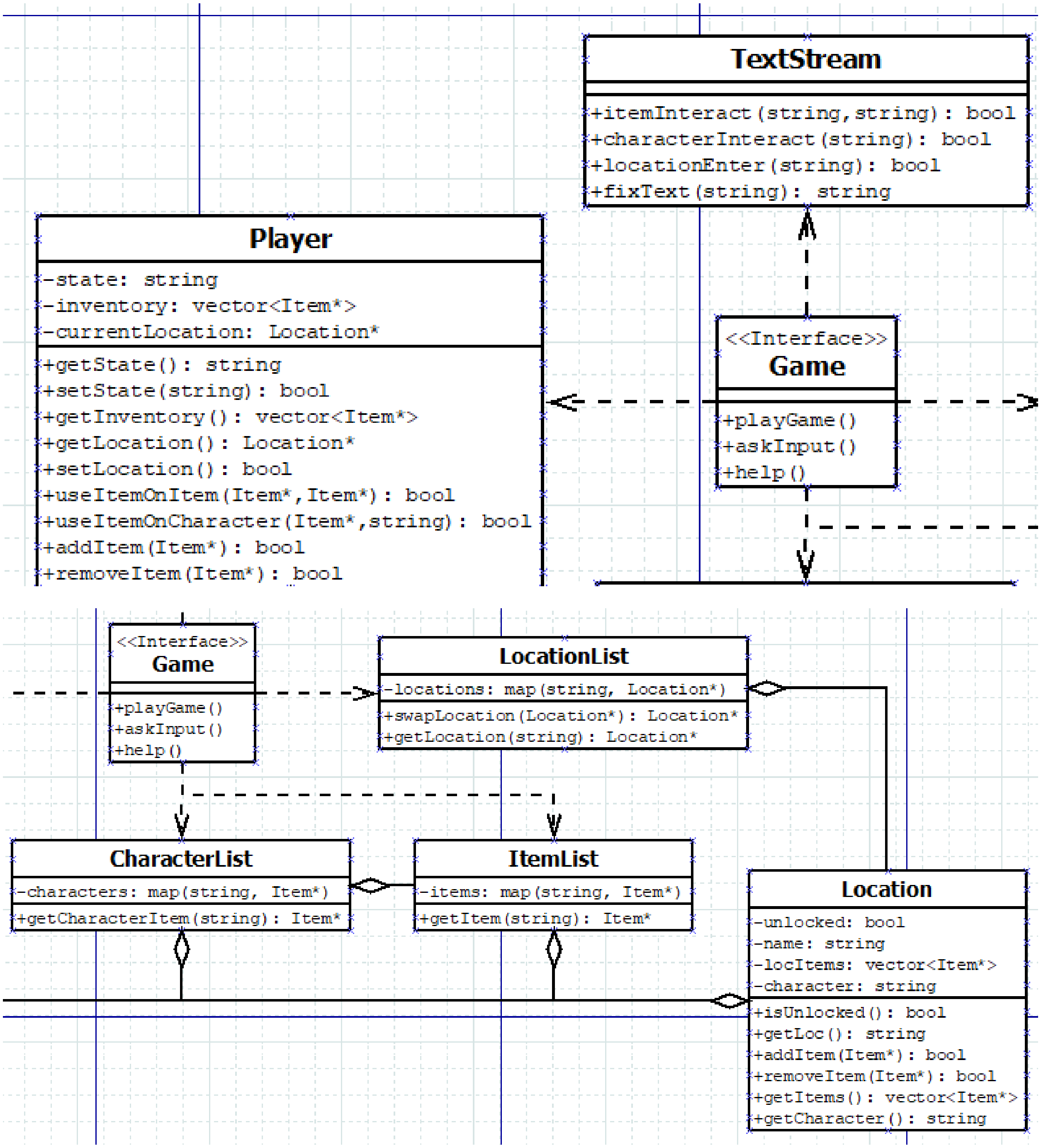
Software Design

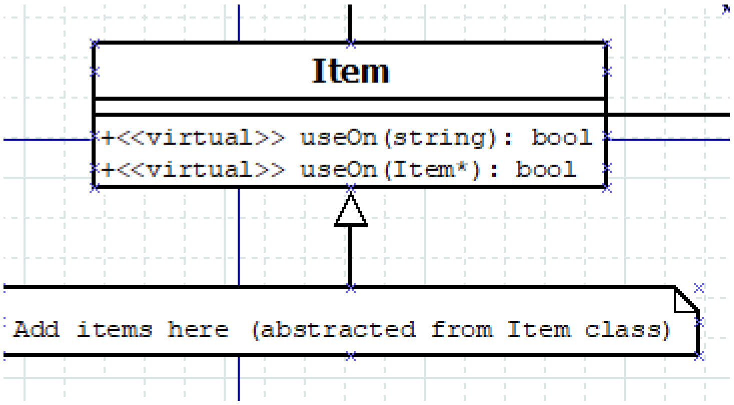
Design – Class diagrams

UML Diagram

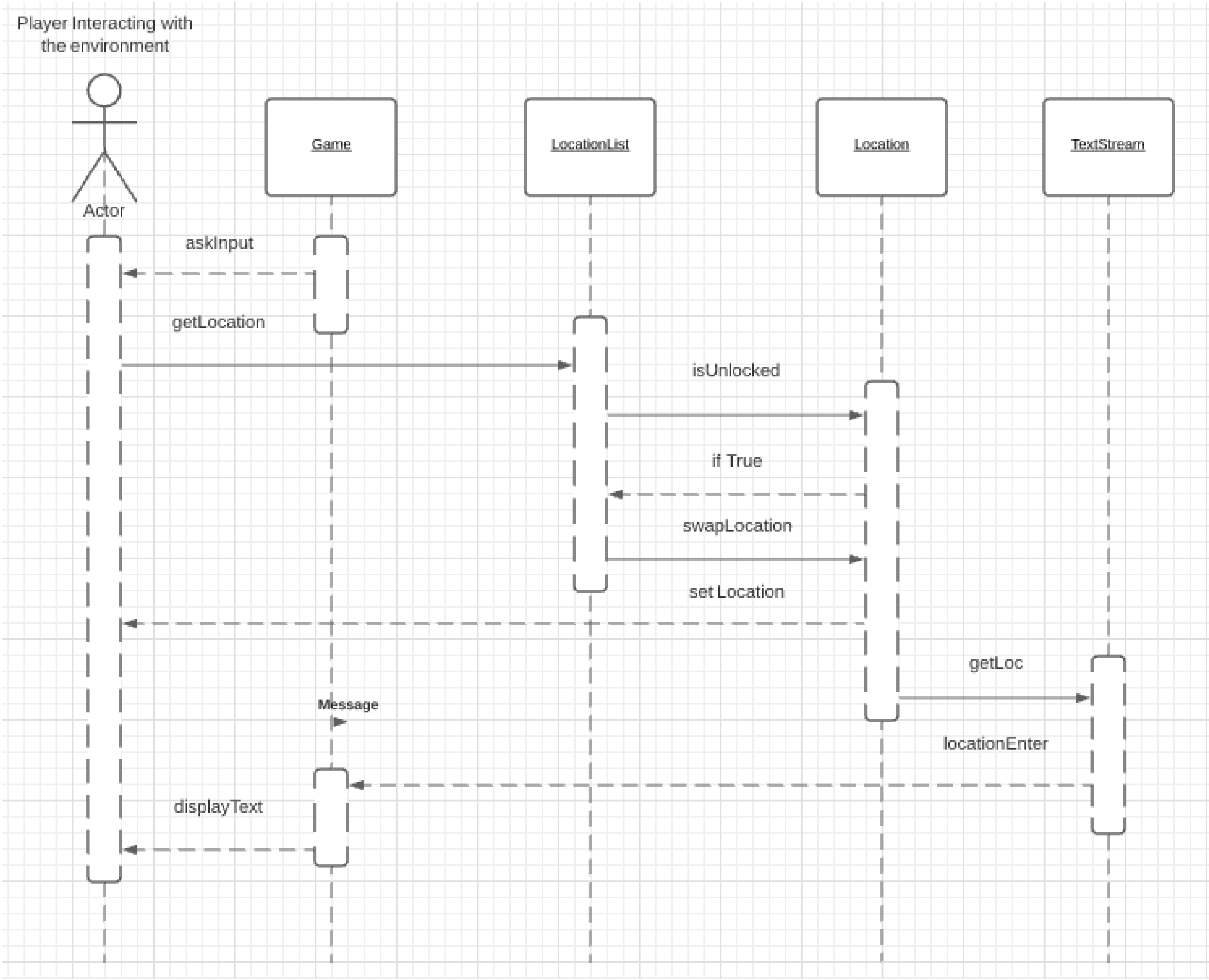

TextStream
+itemInteract(string,string): bool
+characterInteract(string): bool
+locationEnter(string): bool
+fixText(string): string
Player
-state: string
-inventory: vector<Item*>
-currentLocation: Location*
+getState(): string
+setState(string): bool
+getInventory(): vector<Item*>
+getLocation(): Location*
+setLocation(): bool
+useItemOnItem(Item*,Item*): bool
+useItemOnCharacter(Item*,string): bool
+addItem(Item*): bool
+removeItem(Item*): bool
<<Interface>>
Game
+playGame()
+askInput()
+help()
LocationList
-locations: map(string, Location*)
+swapLocation(Location*): Location*
+getLocation(string): Location*
CharacterList
-characters: map(string, Item*)
+getCharacterItem(string): Item*
ItemList
-items: map(string, Item*)
+getItem(string): Item*
Location
-unlocked: bool
-name: string
-locItems: vector<Item*>
-character: string
+isUnlocked(): bool
+getLoc(): string
+addItem(Item*): bool
+removeItem(Item*): bool
+getItems(): vector<Item*>
+getCharacter(): string
Item
+<<virtual>> useOn(string): bool
+<<virtual>> useOn(Item*): bool
Add items here (abstracted from Item class)

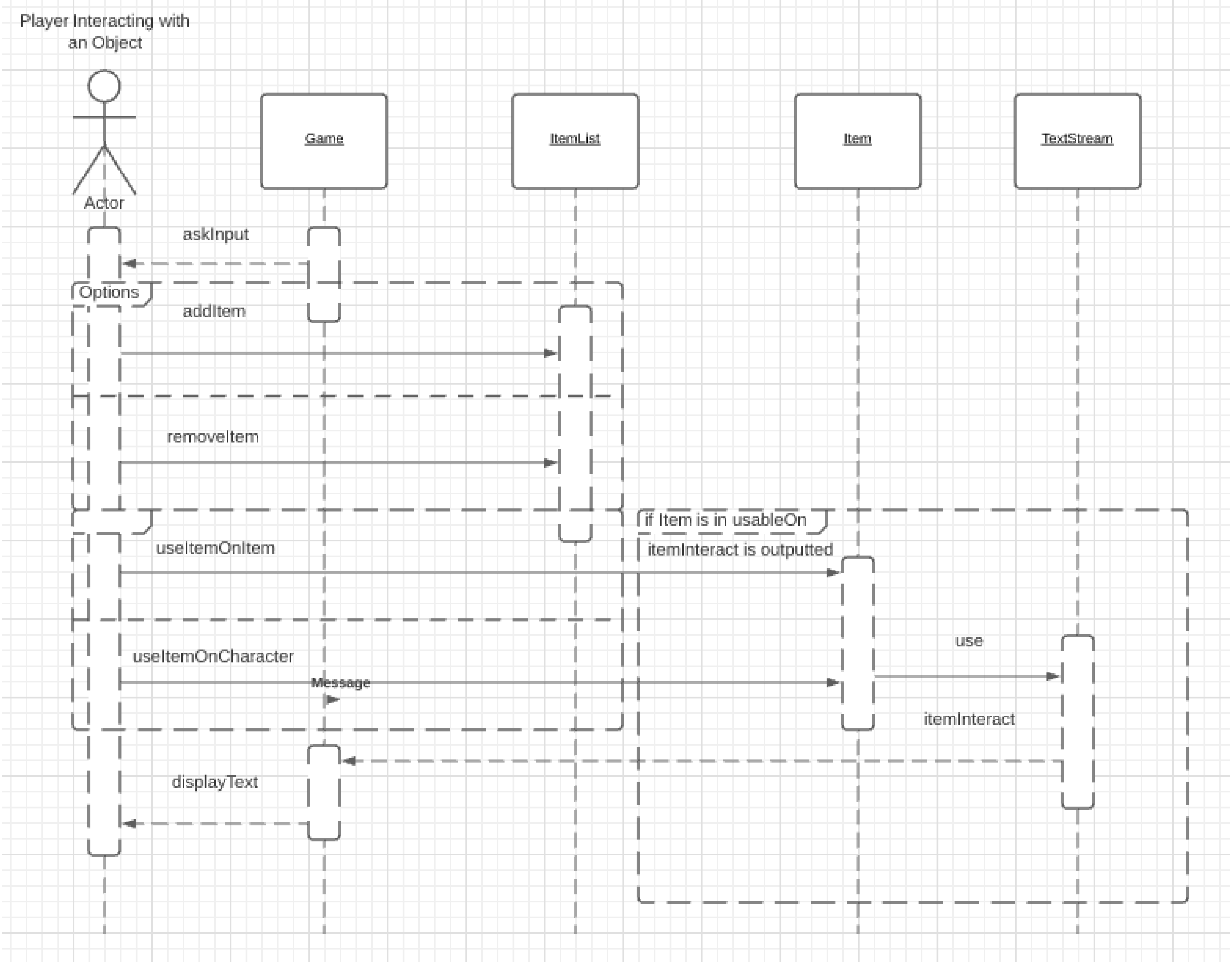
Close Up UML

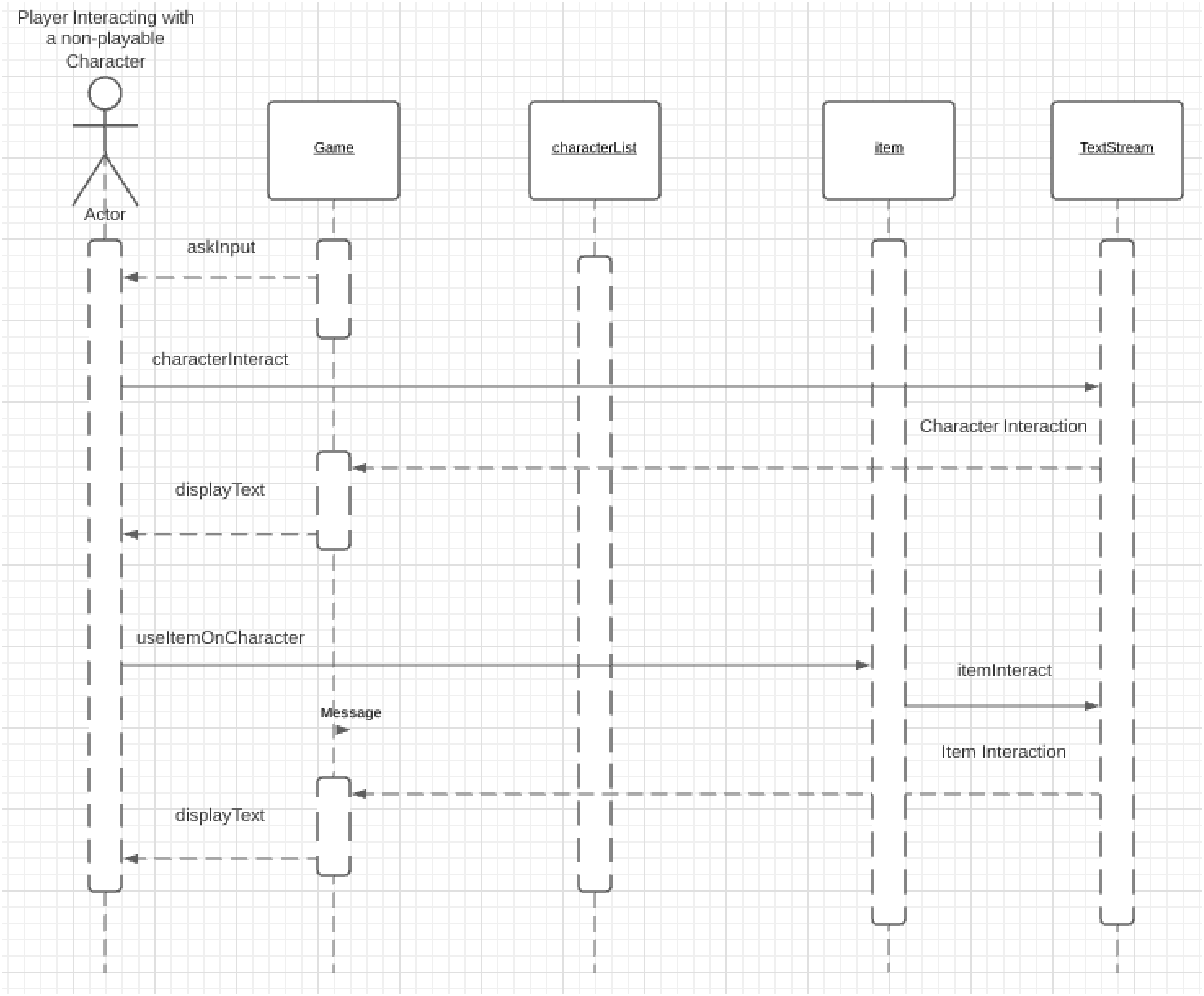

TextStream
+itemInteract(string,string): bool
+characterInteract(string): bool
+locationEnter(string): bool
+fixText(string): string
Player
-state: string
-inventory: vector<Item*>
-currentLocation: Location*
+getState(): string
+setState(string): bool
+getInventory(): vector<Item*>
+getLocation(): Location*
+setLocation(): bool
+useItemOnItem(Item*,Item*): bool
+useItemOnCharacter(Item*,string): bool
+addItem(Item*): bool
+removeItem(Item*): bool
<<Interface>>
Game
+playGame()
+askInput()
+help()
<<Interface>>
Game
+playGame()
+askInput()
+help()
LocationList
-locations: map(string, Location*)
+swapLocation(Location*): Location*
+getLocation(string): Location*
CharacterList
-characters: map(string, Item*)
+getCharacterItem(string): Item*
ItemList
-items: map(string, Item*)
+getItem(string): Item*
Location
-unlocked: bool
-name: string
-locItems: vector<Item*>
-character: string
+isUnlocked(): bool
+getLoc(): string
+addItem(Item*): bool
+removeItem(Item*): bool
+getItems(): vector<Item*>
+getCharacter(): string


Item
+<<virtual>> useOn(string): bool
+<<virtual>> useOn(Item*): bool
Add items here (abstracted from Item class)

Design – sequence diagrams




Player Interacting with an Object
Actor
Game
ItemList
Item
TextStream
askInput
Options
addItem
removeItem
useItemOnItem
useItemOnCharacter
Message
if Item is in usableOn
itemInteract is outputted
use
itemInteract
displayText



Class Descriptions

Classes

Player - The class of the player which records what room the player is in, and what items the player is holding. Players would have a way to retrieve their current inventory or location, or possible locations they can go to and output them to the screen. Item interactions will be done through useItemOn(item*, item*) or useItemOnCharacter(item*, string) and the getItem(item*) which will use and retrieve items for the player class.

Item - Abstract class that has a context attribute usableOn which is a vector of strings of things that it can interact with. When a player calls to use the item, it will check the individual item class useOn() and if the conditions are met it will return to the character the new item or open a new location based on the input parameter.

ItemList - Class that holds all the items in the game and a string name for each. Allows for easy translation of string to item class.

CharacterList - Class that has a map of each character’s name and relevant item for them.

Location - the class of the room the character is in. Has a bool to check if the room can be accessed, a vector of items that are present in the room, and a character name of who is in the room. The functions are able to addItem(), removeItem(), getItem, getCharacter to all return the contents of the room. The bool attribute unlocked (accessed through isUnlocked()) will determine if the location is available to be accessed.

LocationList - Class that holds all the locations in the game with functions that will retrieve a location and easy way of pulling a location from to update the player’s current location

TextStream - File streamer to read dialogue from text files. As well has the function which streams in the current input and translates it to upper case. Also checks with list classes to check if names are present in each list.

Game - Allows for the start and the interface input of the game. Includes the help function that players can use to see the possible ways you can interact with the game. Will be the main way for the game to ask the user for an input.