McGill University

Comp 303: Software Design

Milestone 1: Project Proposal

Team Name: Parry Hotter

Allison Meikle

Filip Snitil

Murad Novruzov

Table of Contents

I. Main Concept and Theme

i. Theme Description

In our virtual world, we propose creating Dumbledore's Office, an interactive magical space that brings to life one of the most iconic locations from the Harry Potter universe. This space serves as both an educational and immersive environment where players can experience fundamental elements of the wizarding world through thoughtfully designed interactions and atmospheric details.

The office captures the essence of Hogwarts through its carefully crafted magical atmosphere, featuring stone floors adorned with a house-themed rug, dark wooden furniture with elegant gold accents, and ambient lighting provided by eternally floating candles. Central to the theme are AI-powered magical portraits of notable wizarding world characters that engage in dynamic conversations with players through a chat interface, Fawkes the phoenix who occasionally bursts into flames and regenerates, and the legendary Sorting Hat ready to engage with players in the time-honoured tradition of house sorting.

The consistency of the magical theme extends beyond aesthetics, as every interactive element serves to deepen the player's connection to the wizarding world. The room actively responds to player actions – interactive objects glow when approached, indicating for the player to explore their functionality like taking books from the shelves, initiating conversations with the portraits and pensieve, or taking part in the sorting ceremony which magically transforms the central rug to reflect the player's assigned house colours and emblem. These dynamic elements create a cohesive, responsive environment that maintains the immersive magical atmosphere while remaining true to both the source material and the fantasy framework of our virtual world.

ii. Room Layout

	W	W	P	W	W/t	W	P	W	W/t	W
W	.	.	./t	*	p	*	./t	.	F	W
P	./t	.	R	R	R	R	R	.	./t	P
W	*	.	R	R	R	R	R	.	*	W
W/t	B	.	R	d	d	d	R	.	.	W
W/t	B	.	R	d	d	d	R	.	*	W
W/t	B	.	R	R	R	R	R	.	.	W
W	*	.	R	R	R/t	R	R	.	*	W
W	.	.	.	*	S	*	.	.	.	W
W	.	.	.	.	.	.	.	.	.	W
W	W	W	W	W	D	W	W	W	W	W

Legend:
W - Wall/Window
B - Bookshelf
D – Door !
d – Desk
R – Rug !
P - Portrait
p - Pensieve
F - Phoenix (Fawkes)
S - Sorting Hat
. - Floor tile !
* - Floating Candle

Note: any tile that says /t will be where one of the text bubbles for the interactive objects pops up when the player is within the proximity range (specified above) telling the player what the object is and how to interact with it (bubble appears above the player if the player is on the same tile). Also note, the objects marked with a ! means the player can walk on the tile containing that object.

iii. Front-End Components & Player Interaction

When players enter Dumbledore's Office, an advisory will appear in the dialogue box as seen below:

“Welcome to Dumbledore’s Office! Please be aware that this room is modelled after the Harry Potter series and we wish to acknowledge the ongoing public discussions and controversy regarding statements made by its author, J.K. Rowling, particularly concerning gender identity. We encourage all players to approach this game with an informed understanding of these discussions and to engage with the content thoughtfully and inclusively.”

Players may then begin their journey with the mandatory Sorting Ceremony, stepping onto a designated tile beneath the Sorting Hat which rests prominently on an elevated pedestal near the room's entrance. When they step onto this tile, a text bubble will appear on the tile above the Sorting Hat, welcoming the player to the Sorting Ceremony and inviting them to begin by pressing the space bar. If they leave this tile before having activated the quiz (pressed the space bar), none of the other objects in the room will be able to be interacted with until the player returns to this tile and presses the space bar. Once they do so, it triggers a short behavioural quiz which asks them a series of character-defining questions through the menu function, designed to help them reflect on their values and personality traits. Based on their responses, the Sorting Hat assigns them to the most fitting house, bringing a personalized touch to their magical journey. Once sorted, they witness the room magically adapt to their house assignment through a change in the central rug's colour and emblem. The player can also come back at any point and take the sorting quiz again if they wish to see how different answers may affect their assigned house and see the different themes the room has to offer.

After sorting, players can move through the space to explore the other interactive elements. Walking down the left side of the room, they pass a wall adorned with a dark wooden bookshelf. As the player walks on the adjacent tiles (Manhattan distance of one tile), a text bubble appears on the three tiles to the left of the bookshelf, listing the names of available titles to choose from and prompting the user to enter the /takeBook command followed a title of one of the listed books (for example "/takeBook History of Magic") to select a magical text. Upon selection, the game will interface with the large language model (LLM) API to generate a personalized book description incorporating the player's username. Then, the selected book will gracefully float from the shelf, and the text bubble will then prompt the player to walk onto the tile containing the book to pick it up. This will reveal the personalized description of the book's contents in the dialogue box and return the book to the bookshelf, enhancing the magical atmosphere of exploration and discovery.

Then, as the player approaches the front of the office, they will pass by the four interactive magical portraits of Dumbledore, Hermione, Professor Snape, and Peeves the ghost. When the player stands on the tile directly in front of each portrait, another text bubble will appear in front of it telling the player which character is portrayed and inviting the player to talk to that portrait through the /chat command. For example, upon approaching Dumbledore's portrait, the user could enter "/chat How do I get into Hogwarts" into the chat to gain insight from the wise headmaster. These portraits then interface with the LLM API to display dynamic and engaging responses to the user's inputs in the dialogue box. These dialogues blend educational content about the wizarding world with entertaining interactions unique to the character portrayed in each.

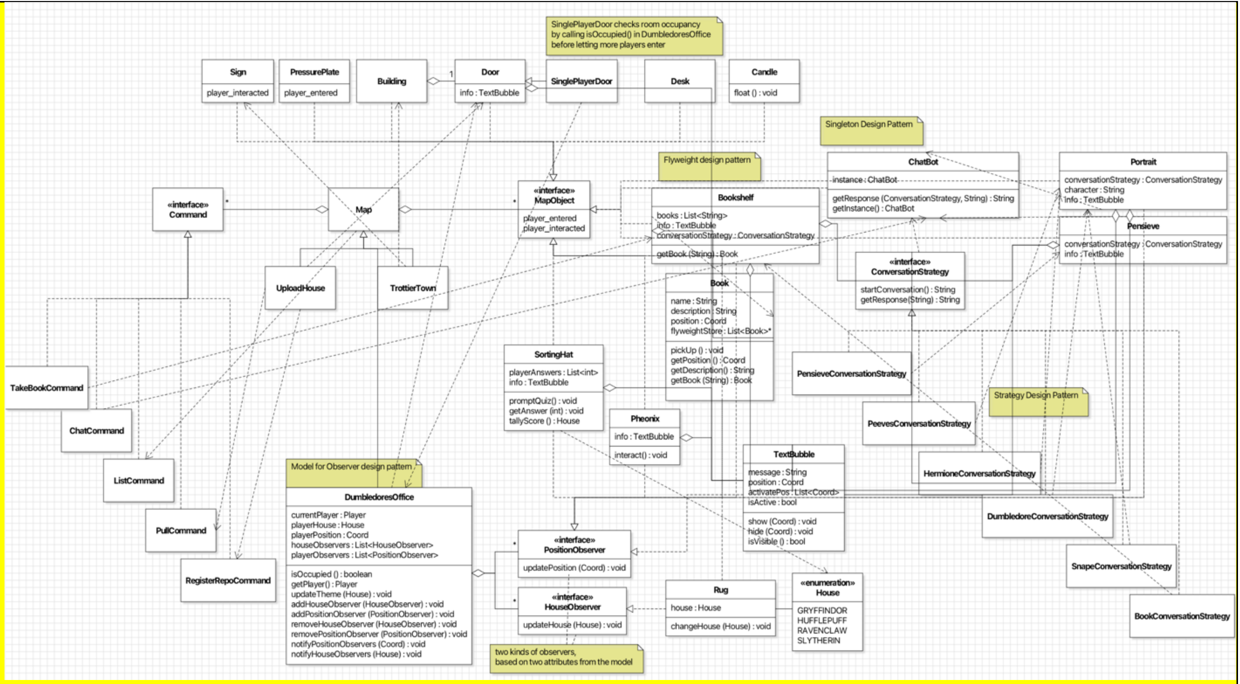
Similarly, Dumbledore's Pensieve resides on its own decorated stand centred in the front of the room. When the player stands on the tile in front of the Pensieve, another text bubble appears above the Pensieve, explaining that the Pensieve offers opportunities for memory exploration and prompting the player to ask it a question. By interacting with the Pensieve – again through the /chat command – players are presented with vivid descriptions of iconic moments from the wizarding world, such as Harry casting his first Patronus or Dumbledore’s duel with Voldemort. Each memory is brought to life through a description generated by the LLM, allowing players to engage further by asking follow-up questions, such as, “Why is this memory significant?” or “When did this happen?”. The Pensieve provides thoughtful, lore-rich responses, encouraging exploration of the emotional and narrative depth of these key events.

Finally, Dumbledore’s prized phoenix, Fawkes, majestically perches near Dumbledore's desk in the top-right corner of the office. When players approach him and are standing on any of the three adjacent tiles to Fawkes, a text bubble appears below him, inviting the player to interact with Fawkes by pressing the space bar. This triggers the phoenix's fiery and dramatic rebirth in flames, a wonderous visual for the player.

Other non-interactive elements enhance the room's atmosphere, such as an ambient lighting system of perpetually floating candles casting a warm glow across the space. At the center of the room sits Dumbledore's iconic desk, anchoring the space with its timeless presence. The room's design encourages exploration and discovery, with each interactive object providing clear interaction instructions through the player’s proximity-activated text bubbles. The space adapts to each player's house assignment, ensuring unique experiences while maintaining the cohesive magical atmosphere of Dumbledore's Office. Players can also revisit different aspects of the room as many times as they desire, with each interaction potentially revealing new dialogue or information.

II. Technical Implementation

i. Back-End Architecture


SinglePlayerDoor checks room occupancy by calling isOccupied() in DumbledoresOffice before letting more players enter
Sign
player_interacted
PressurePlate
player_entered
Building
1
Door
info : TextBubble
SinglePlayerDoor
Desk
Candle
float () : void
Singleton Design Pattern
Flyweight design pattern
«interface» Command
*
Map
*
«interface» MapObject
player_entered
player_interacted
Bookshelf
books : List<String>
info : TextBubble
conversationStrategy : ConversationStrategy
getBook (String) : Book
ChatBot
instance : ChatBot
getResponse (ConversationStrategy, String) : String
getInstance() : ChatBot
Portrait
conversationStrategy : ConversationStrategy
character : String
info : TextBubble
Pensieve
conversationStrategy : ConversationStrategy
info : TextBubble
UploadHouse
TrottierTown
«interface» ConversationStrategy
startConversation() : String
getResponse(String) : String
Book
name : String
description : String
position : Coord
flyweightStore : List<Book>*
pickUp () : void
getPosition () : Coord
getDescription() : String
getBook (String) : Book
SortingHat
playerAnswers : List<int>
info : TextBubble
promptQuiz() : void
getAnswer (int) : void
tallyScore () : House
TakeBookCommand
ChatCommand
ListCommand
PullCommand
RegisterRepoCommand
PensieveConversationStrategy
Strategy Design Pattern
PeevesConversationStrategy
Pheonix
info : TextBubble
interact() : void
TextBubble
message : String
position : Coord
activatePos : List<Coord>
isActive : bool
show (Coord) : void
hide (Coord) : void
isVisible () : bool
HermioneConversationStrategy
Model for Observer design pattern
DumbledoresOffice
currentPlayer : Player
playerHouse : House
playerPosition : Coord
houseObservers : List<HouseObserver>
playerObservers : List<PositionObserver>
isOccupied () : boolean
getPlayer() : Player
updateTheme (House) : void
addHouseObserver (HouseObserver) : void
addPositionObserver (PositionObserver) : void
removeHouseObserver (HouseObserver) : void
removePositionObserver (PositionObserver) : void
notifyPositionObservers (Coord) : void
notifyHouseObservers (House) : void
DumbledoreConversationStrategy
*
«interface» PositionObserver
updatePosition (Coord) : void
*
«interface» HouseObserver
updateHouse (House) : void
Rug
house : House
changeHouse (House) : void
«enumeration» House
GRYFFINDOR
HUFFLEPUFF
RAVENCLAW
SLYTHERIN
SnapeConversationStrategy
BookConversationStrategy
two kinds of observers, based on two attributes from the model

Note: This diagram is quite small and hard to read, thus we have included a separate pdf of the diagram in our submission.

The back-end architecture centers on a central *DumbledoresOffice* class that manages the core room attributes and maintains consistency for a single player's experience. This class tracks essential room-wide properties such as the current player, their house (originally None), their position, and lists of the interactive objects in the room that observe subsequent changes to the game state. To ensure a more focused and personalized magical experience, the room is designed to accommodate one player at a time. This is accomplished through *SinglePlayerDoor,* a subclass of *Door,* which communicates with *DumbledoresOffice* to check if the room is occupied before calling the method in its parent class to allow another player to enter.

The *SortingHat* class handles the house sorting ceremony, implementing a quiz system that is activated by pressing the space bar when the player is standing on the tile in front of the sorting hat. This displays the quiz to the player inside the menu. Once the quiz is complete, it tallies the player's answers into a final score and returns one of the options from the *House* enumeration, which it then communicates to *DumbledoresOffice* to update the room theme. This then notifies all implementing classes of the *HouseObserver* interface, namely the room's *Rug* class, to change its colour and emblem to reflect the newly sorted house whenever the player chooses to take the sorting quiz.

A core *PositionObserver* interface serves as the foundation for all other interactive elements in the room, including the *Pensieve, Portrait, Sorting Hat, Phoenix, TextBubble,* and *Bookshelf* which implement this interface. Each of these classes stores a *TextBubble* field, and when a player is within the specified range (stored in the *TextBubble*) the bubble appears, instructing the player on how to interact with the object. When the *TextBubble* for an object is visible, this means the specified interaction described is available for the player to take. The *Pensieve, Portrait,* and *Bookshelf* classes also contain a *ConversationStrategy* field specific to their character/function. The *ConversationStrategy* interface is what communicates with the LLM's API to respond to players' input from the /chat command. This approach allows for efficient management of the four distinct portrait personalities while maintaining consistent interaction mechanics.

The other decorative elements, the *Desk* and *Candle,* simply implement the *MapObject* interface and offer no other functionality other than the candle's continual floating appearance, achieved by moving the candle's image slightly back and forth.

ii. Primary Design Patterns

The **Strategy** pattern will be used for implementing the diverse interaction behaviours needed in Dumbledore's Office. Through this pattern, we can define a family of dialogue algorithms for our *Portrait* and *Pensieve* classes, which we will pass to the *getResponse()* method of the *ChatBot* to maintain unique conversation strategies while sharing common interaction mechanics. When a player approaches a portrait or the Pensieve and initiates dialogue through the /chat command, the appropriate dialogue strategy is retrieved through the attribute stored in that object which implements the *ConversationStrategy* interface. Then the /chat command passes the player's input string and the object's *ConversationStrategy* to the *ChatBot* which interfaces with the LLM API and returns the corresponding response. This approach provides the flexibility to easily add new character personalities or modify existing dialogue patterns without altering the core *Portrait*, *Pensieve,* and *ChatBot* class implementations.

The **Observer pattern** creates the foundation for our room's dynamic response system through a carefully structured relationship between the model class *DumbledoresOffice* and our observer interfaces *PositionObserver* and *HouseObserver*. All interactive elements in the room, from portraits to magical artifacts, extend from either of these two interfaces depending on which aspect of the game state is relevant to them. The *Rug*, for example, is the only interactive object affected by changes to the current player's *House*. Thus, it is notified by *DumbledoresOffice* when this attribute changes (every time the player chooses to take the sorting quiz and is assigned a different house), allowing it to adapt its appearance. Otherwise, the remaining interactive objects are player proximity-based, so as the player moves through the room, *DumbledoresOffice* notifies all instances of implementing classes of the *PositionObserver* interface to show their TextBubble if the player is within active proximity range to describe the objects to the user and instruct them on how to proceed with the interactions. For instance, the *Portrait* class, which manages our four magical portraits through a single class implementation with unique character configurations, observes these changes to adjust which (if any) character's portrait *TextBubble* is showing, telling the player which character is portrayed and prompting them to interact with it through the /chat command. This pattern proves especially powerful for maintaining synchronization between house-themed elements, portrait behaviours, and ambient magical effects, ensuring that all room components respond cohesively to state changes while preserving the distinct personalities and interactions of each portrait through our API integration.

iii. Supporting Design Patterns

The **Singleton pattern** is used to manage the Chatbot LLM API**,** ensuring that all objects interacting with the API access a single shared instance rather than creating multiple instances. This centralization improves efficiency and maintains consistency across all AI-driven dialogues in the game. The Chatbot class serves as the global access point for all objects that require responses from the LLM, such as Portrait, Pensieve, and Bookshelf. To personalize interactions, the chatbot's getResponse method utilizes a ConversationStrategy, which is an attribute stored in each object that interacts with the LLM. When an object calls getResponse, it passes both the ConversationStrategy and the user's input. The chatbot then formats the prompt using the strategy's character-specific preface to ensure the response matches the personality of the speaking character. For example, if a player interacts with Dumbledore's portrait, the chatbot constructs the prompt as: *"You are Albus Dumbledore, the wise and enigmatic headmaster of Hogwarts. Respond to this in a wise and slightly cryptic manner: {user_input}."* The chatbot then sends this customized prompt to the LLM API and returns the generated response to the original object, which displays it in the dialogue box. By implementing the Singleton pattern, all objects that require AI-generated dialogue access the same chatbot instance, reducing redundant API calls and improving performance. This approach ensures that responses are generated efficiently, maintains a consistent conversational style across all characters, and enhances the overall immersive experience of the game.

The **Flyweight** pattern is used to implement the *Bookshelf* and *Book* classes, optimizing memory usage and reducing redundant LLM API calls. The *Book* class maintains a static flyweight store that maps book titles (String) to their corresponding *Book* objects. When a player uses the /takeBook command, the *Bookshelf* calls its getBook(title) method, which in turn invokes Book.getBook(title). The *Book* class then checks its flyweight store to determine if an instance for the requested title already exists. If the book exists: The existing *Book* object is returned, and the *Bookshelf* animates it floating onto a tile in front of the player for interaction. If the book does not exist: The *Book* class invokes its private constructor, which generates a customized book description using the singleton *ChatBot* instance, incorporating both the book title and the player's username. The new *Book* object is then stored in the flyweight store for future retrieval. By applying the Flyweight pattern, we ensure that each book title is stored only once, preventing redundant API calls. This significantly improves performance by eliminating unnecessary requests to the LLM, reducing response times while maintaining an immersive and dynamic game experience.

III. Development Timeline

Note: This timeline is only tentative — the tasks and individual responsibilities may change because of bugs and problems we may encounter during the coding part.

Week 1 (Feb 17 – Feb 23):

- **Allison:** Implement DumbledoresOffice Singleton (magical effects, house updates).
- **Murad:** Set up Player movement logic with collision detection tied to MapObject.
- **Filip:** Develop Command interface and implement TakeBookCommand and ChatCommand (proximity detection framework).

Week 2 (Feb 24 – Mar 2):

- **Allison:** Design UI components for SortingHat interactions and house display.
- **Murad:** Implement PositionObserver and HouseObserver for dynamic updates.
- **Filip:** Begin SortCommand and integrate with SortingHat quiz logic.

Week 3 (Mar 3 – Mar 9):

- **Allison:** Add visual feedback (lighting effects using Bookshelf.lightUp() and stopLight()).
- **Murad:** Integrate player movement with magical object triggers (e.g., Bookshelf, Phoenix).
- **Filip:** Develop Portrait dialogue system using the Strategy pattern (implement PeevesConversationStrategy, HermioneConversationStrategy).

Week 4 (Mar 10 – Mar 16):

- **All: Integration Sprint #1** – Combine UI, player mechanics, magical interactions, and command executions.

Project Check-in with TAs: March 17, 2025 (Monday)

- Demonstrate core features: Singleton, Observer updates, command executions.
- Review design patterns and receive feedback.

Week 5 (Mar 18 – Mar 24):

- **Allison:** Refine ChatBot UI for magical interactions with portraits and Pensieve.
- **Murad:** Implement Phoenix animations (burnAndRegenerate() lifecycle).
- **Filip:** Finalize PetPhoenixCommand and integrate with Phoenix interactions.

Week 6 (Mar 25 – Mar 31):

- **Allison:** Polish SortingHat quiz UI, integrate with UploadHouse command.
- **Murad:** Optimize Observer pattern performance for dynamic house updates.
- **Filip:** Add advanced dialogue features with branching logic in DumbledoreConversationStrategy and SnapeConversationStrategy.

Week 7 (Apr 1 – Apr 6):

- **All: Integration Sprint #2** – Finalize system integration, conduct alpha testing, and debug.

Final Demo & Submission (April 7 – April 11, 2025)

- **Allison:** Ensure UI responsiveness, polish magical effects.
- **Murad:** Debug game logic, ensuring stable observer notifications.
- **Filip:** Final testing of command executions and dialogue interactions.
- **All:** Write final documentation and prepare for the demo.

Demo: Scheduled during this week (TBD by course).

Final Submission Deadline: April 11, 2025 (Friday).