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. Player Experience and Interaction

When players enter Dumbledore's Office, they begin their journey with the Sorting Ceremony, stepping onto a designated tile beneath the Sorting Hat. After completing an interactive quiz that determines their Hogwarts house, they witness the room magically adapt to their assignment through a change in the central rug's appearance.

After sorting, players can move through the space to explore the other interactive elements. As they approach the front of the room, they pass by dark wooden bookshelves, filled with magical texts that respond to player proximity by emitting a soft glow. Players can select a book from the bookshelf, and it will float out onto a nearby tile from which the player can then pick it up and see a short description of their selection. They also pass by magical portraits which glow when a player moves to a position within a Manhattan distance of two tiles, prompting the player to initiate a conversation with them. These dialogues blend educational content about the wizarding world with entertaining interactions unique to the character portrayed in each. As players move through the space, they encounter various other magical objects that respond to their presence: Fawkes the phoenix bursts into flames and regenerates when the player interacts with him and the Pensieve offers opportunities for memory exploration.

The room's design encourages exploration and discovery, with each interactive object providing clear feedback through an illuminating glow. The space adapts to each player's house assignment, ensuring unique experiences while maintaining the cohesive magical atmosphere of Dumbledore's Office. Players can revisit different aspects of the room multiple times, with each interaction potentially revealing new dialogue or information.

II. Technical Implementation

i. Front-End Components

The front-end design of Dumbledore's Office creates an immersive magical environment through carefully placed visual elements and interactive components. The foundation of the space consists of stone floor tiles, creating a classic Hogwarts aesthetic. At the center lies an intricately decorated rug that dynamically changes its design and colours to reflect the player's assigned house after the sorting ceremony, serving as a focal point of the room.

The office walls feature four interactive magical portraits of Dumbledore, Hermione, Professor Snape, and Peeves the ghost, each emitting a subtle glow when players move within their two-tile activation range (measured in Manhattan distance). This visual feedback system employs subtle tile highlighting, creating an intuitive interface that maintains immersion while clearly communicating available interactions to players. Once in range, players can initiate a personalized conversation with the characters by using the /chat command, which interfaces with a large language model (LLM) API to create dynamic and engaging interactions.

Similarly, the Pensieve resides on its own decorated stand, featuring an ethereal glow as the player moves into its two-tile activation range. 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 an 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.

The office walls are adorned with dark wooden bookshelves that glow faintly as players walk on the adjacent tiles. When standing in front of a glowing bookshelf, players can use the /takeBook command to select a magical text, which gracefully floats from the shelf. Walking onto the tile containing the book allows the player to pick it up, revealing a brief and immersive description of the book's contents, enhancing the magical atmosphere of exploration and discovery.

The Sorting Hat rests prominently on an elevated pedestal near the room's entrance, with a specially marked floor tile beneath it to signify the sorting ceremony location. Players can initiate the ceremony by using the /sort command, which triggers a short behavioural quiz 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.

Fawkes's majestic perch stands proudly near Dumbledore's desk, crafted to highlight the phoenix's fiery and dramatic rebirth in flames. Players can interact with Fawkes by using the /petPhoenix command when positioned on any adjacent tile, triggering a mesmerizing display of the phoenix's magical abilities.

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.

ii. Room Layout

W	W	W	P	W	W	W	P	W	W	W
W	.	.	.	*	p	*	.	.	F	W
P	.	.	R	R	R	R	R	.	.	P
W	*	.	R	R	R	R	R	.	*	W
W	B	.	R	d	d	d	R	.	B	W
W	B	.	R	d	d	d	R	.	B	W
W	B	.	R	R	R	R	R	.	B	W
W	*	.	R	R	R	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

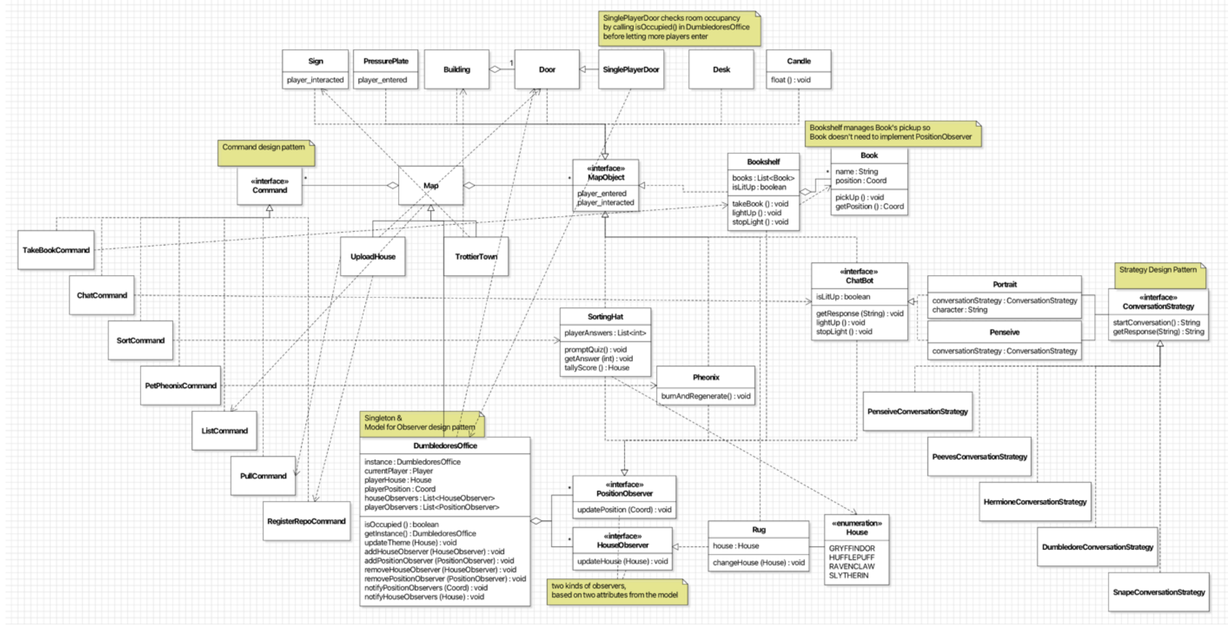
F - Phoenix (Fawkes)

S - Sorting Hat

. - Floor tile

* - Floating Candle

iii. 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
SinglePlayerDoor
Desk
Candle
float () : void
Bookshelf manages Book's pickup so Book doesn't need to implement PositionObserver
Command design pattern
«interface» Command
Map
«interface» MapObject
player_entered
player_interacted
Bookshelf
books : List<Book>
isLitUp : boolean
takeBook () : void
lightUp () : void
stopLight () : void
Book
name : String
position : Coord
pickUp () : void
getPosition () : Coord
TakeBookCommand
UploadHouse
TrottierTown
«interface» ChatBot
isLitUp : boolean
getResponse (String) : void
lightUp () : void
stopLight () : void
Portrait
conversationStrategy : ConversationStrategy
character : String
Penseive
conversationStrategy : ConversationStrategy
Strategy Design Pattern
«interface» ConversationStrategy
startConversation() : String
getResponse(String) : String
ChatCommand
SortCommand
SortingHat
playerAnswers : List<int>
promptQuiz() : void
getAnswer (int) : void
tallyScore () : House
PetPheonixCommand
Pheonix
burnAndRegenerate() : void
PenseiveConversationStrategy
ListCommand
Singleton & Model for Observer design pattern
DumbledoresOffice
instance : DumbledoresOffice
currentPlayer : Player
playerHouse : House
playerPosition : Coord
houseObservers : List<HouseObserver>
playerObservers : List<PositionObserver>
isOccupied () : boolean
getInstance() : DumbledoresOffice
updateTheme (House) : void
addHouseObserver (HouseObserver) : void
addPositionObserver (PositionObserver) : void
removeHouseObserver (HouseObserver) : void
removePositionObserver (PositionObserver) : void
notifyPositionObservers (Coord) : void
notifyHouseObservers (House) : void
PeevesConversationStrategy
PullCommand
«interface» PositionObserver
updatePosition (Coord) : void
HermioneConversationStrategy
RegisterRepoCommand
«interface» HouseObserver
updateHouse (House) : void
Rug
house : House
changeHouse (House) : void
«enumeration» House
GRYFFINDOR
HUFFLEPUFF
RAVENCLAW
SLYTHERIN
DumbledoreConversationStrategy
two kinds of observers, based on two attributes from the model
SnapeConversationStrategy

Note: This diagram is quite small and hard to read, thus we have included the actual JetUML file in our project repository and added 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 through the /sort command in the messages box. This displays questions to the player inside the dialogue box and receives player answers passed as an argument to the /sort command. 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.

A core *PositionObserver* interface serves as the foundation for all other interactive elements in the room. This class activates the glow of interactive objects through player proximity detection. From this base, specialized classes implement this functionality to create specific interactive elements. The *ChatBot* interface, a key extension, implements chatbot functionality such as the glow indicating which portrait the player can converse with. The *Pensieve* and *Portrait* classes implement this interface, each containing a certain *ConversationStrategy* attribute 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 *Pheonix* class also implements the *PositionObserver* interface, only offering the option to enter the command /petPheonix when the player is standing on one of the adjacent tiles. The *Bookshelf* also implements the *PositionObserver* interface, glowing when the player approaches and offering the /takeBook command. Upon this command, it removes a *Book* from its list and floats it out into the room for the player to pick it up and read.

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.

III. Design Patterns

i. Primary Design Patterns

The **Singleton pattern** is central to our implementation through the *DumbledoresOffice* class, which serves as the backbone of Dumbledore's Office's room management system. This class maintains critical room-wide properties such as the current house theme, the player's position, and the list of all interactive objects. By implementing the Singleton pattern, we ensure that the *DumbledoresOffice* class is only instantiated once, providing a single point of access to that instance through a global accessor method *getInstance()* which retrieves the single private instance also stored inside the class. This avoids creating multiple instances and maintains consistency across the application. The *DumbledoresOffice* singleton is particularly crucial for managing the room's appearance, as it maintains the visual theme based on the player's house assignment and handles the transition of themes when players enter or leave the space. This centralized control extends to all room attributes and interactive objects, ensuring a consistent magical experience for the player present.

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, 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 update appropriately. 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 is glowing and can be interacted with 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.

The **interconnection** between these patterns is evident in how the *DumbledoresOffice* singleton coordinates with our observer system to maintain room consistency. This architectural approach allows us to efficiently manage complex state changes while keeping our code modular and maintainable.

ii. Supporting 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, allowing each *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 delegates the player's input string to the *ConversationStrategy* which interfaces with the 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* and *Pensieve* class implementations.

The **Command** pattern offers a robust solution for managing the various player interactions within our magical space. By encapsulating player actions as command objects, we can create a consistent interface for handling diverse interactions such as initiating the sorting ceremony, engaging with chatbots, or manipulating magical objects like the floating books near our bookshelves. Each command implements the abstract *Command* interface to handle player interactions, providing specific implementations that work in conjunction with both the *DumbledoresOffice* singleton and relevant interactive objects. This pattern proves particularly useful for managing our sorting hat quiz system, where player responses must be tracked and processed systematically to determine house assignments. Furthermore, the Command pattern's structure allows us to potentially implement features like interaction history or action queuing, enhancing the room's interactive capabilities while maintaining clean, maintainable code.

These supporting patterns complement our primary **Singleton** and **Observer** implementations by providing additional layers of flexibility and control in our interactive magical environment. Their integration helps create a more sophisticated and extensible system while maintaining the coherent magical atmosphere that defines Dumbledore's Office.

IV. 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).