Horror light

1. Project Overview

It's a game where players find a way to escape the ghost while collecting treasure at the same time. And there will be a flashlight to shine into the darkness to find treasure and defeat ghosts.

2. Project Review

https://youtu.be/8OMghdHP-zs?si=odzQLsTQ5cbD5roa I will use "vampire survivor" to be an inspiration and change the tone of screen , treasure collecting , level changing and other features(flashlight, (maybe)set the time for each round).

3. Programming Development

3.1 Game Concept

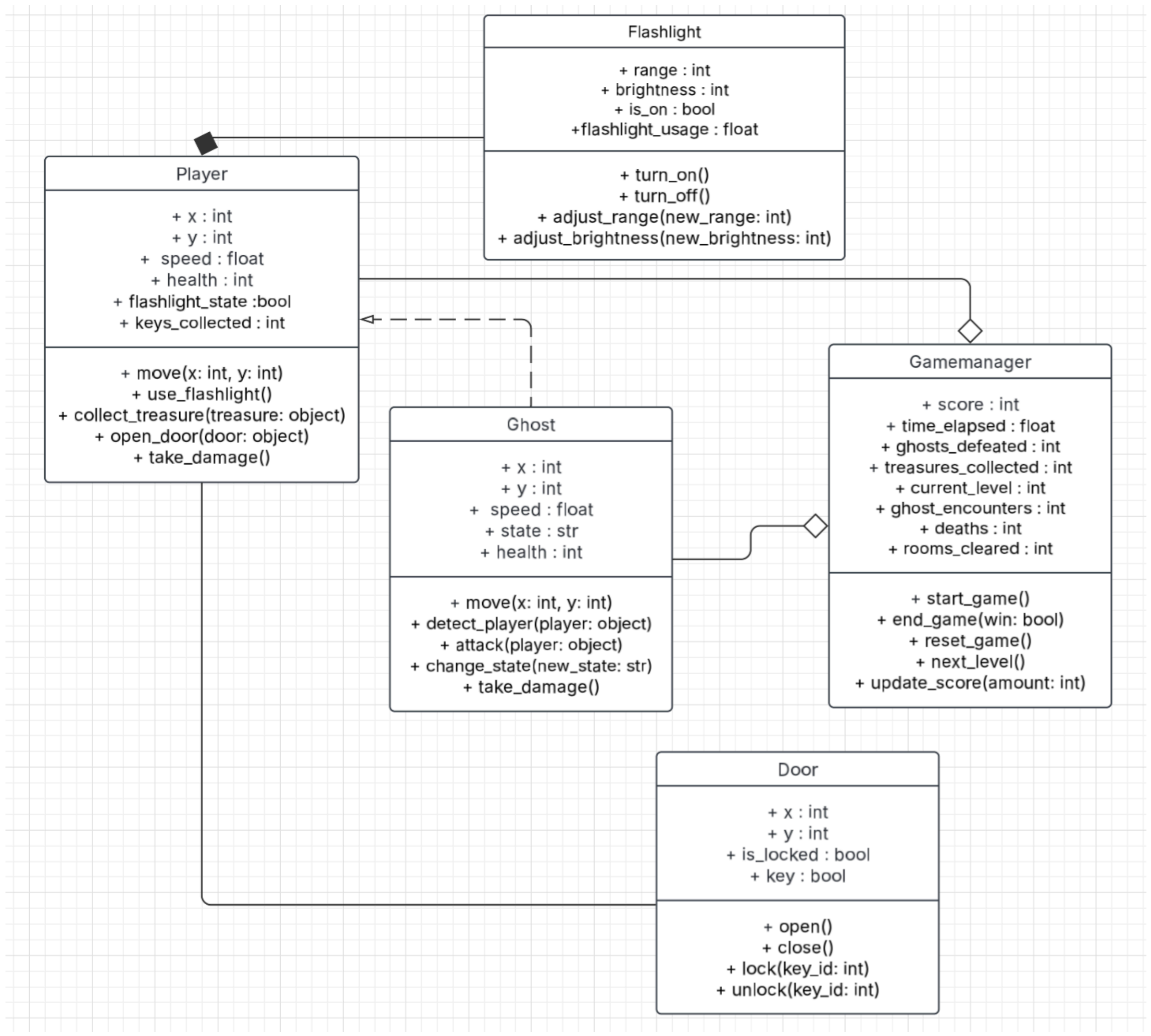
The entire game scene is dim. The character has a flashlight(it has no limited battery but has a limited brightness range and the player can use it anytime) to walk around looking for treasures in different rooms. and find the exit door to go to the next room. There will be a ghost in the room waiting to follow you, so you must use a flashlight to shine a light on the ghost in order to get rid of it. And when the ghost touches us, the game will end.

3.2 Object-Oriented Programming Implementation

(draf)

- Player : create player
- Ghost : control enemy
- Flashlight : Controls the light source and visibility
- Gamemanager : Controls the game state and logic

- Door : doors that can be locked/unlocked



Class	**Role**	**Attributes**	**Methods**
Player	Represents the player character	**x,y:**Position of the player on the screen.**,** **Speed:**Speed at the player moves.**,** **Health:**Number of times the player can survive	**Move()**:Moves the player based on the input direction (dx, dy) and ensures the player stays within the screen boundary. **UseFlashlight():**Toggles the flashlight state (on/off)., **CollectTreasure():**Call

		damage.**,** **FlashlightState:**Indicates if the flashlight is on or off.	ed when the player collides with a treasure. Increases score and updates treasure count., **OpenDoor():**If the player has the required key, the door will be unlocked and opened., **take_damage():**Called when the player is caught by a ghost. Reduces health or ends the game if health is zero.
Ghost	Represents the enemy in the game	**x,y:**Position of the ghost on the screen**,** **Speed:**Speed at which the ghost moves.**,** **State:**Current state of the ghost (Idle, chase, search).**,** **Health:**Number of hits required to defeat the ghost.	**Move():**Uses AI pathfinding to move toward the target position (player position).**,** **DetectPlayer():**Checks if the player is within a certain distance or visible through line of sight.**,** **Attack():**If the ghost collides with the player, calls player.take_damage()**.,** **ChangeState():**Changes the ghost state based on player proximity or visibility., **Take_damage():**Reduces health when the player shines the flashlight on the ghost. If health reaches zero, the ghost is destroyed.
Flashlight	Controls the light source and visibility	**Range:**Radius of the flashlight beam., **Brightness:**Brightness level of	**TurnOn():**Sets `is_on` to `True`, allowing the player to see objects within the beam.**,** **TurnOff():**Sets `is_on`

		the flashlight beam. **is_on:**True if the flashlight is currently on.	to `False`, making the screen dark.**,** **AdjustRange():**Changes the flashlight's range (e.g., upgrade or environmental effect).
Gamemanager	Controls the game state and logic	**Score:**Current player score based on treasures collected.**,** **time_elapsed:**Total time spent in the current session.**,** **ghosts_defeated:**Number of ghosts the player has defeated.**,** **treasures_collected:**Number of treasures collected, **current_level:**Current level of the game.	**StartGame():**Initializes the game state, spawns player and ghosts, and starts the timer.**,** **EndGame():**Ends the game and records the player's performance (win or loss).**,** **ResetGame():**Resets all variables and starts a new session.**,** **NextLevel():**Increases the level and spawns new ghosts and treasures. **Update_score(amount):**Updates the player's score when treasure is collected or a ghost is defeated.
Door	Represents doors that can be locked/unlocked	Position, **x,y:**Position of the door on the screen, **is_locked**:If `True`, the door requires a key to open.**,** **key:**the key that is used for opening the door (if applicable).	**Open():**Opens the door if it's unlocked or the player has the key.**,** **Close():**Closes the door. Useful for treasure or gameplay mechanics.**,** **Lock():**Locks the door with a key.**,** **Unlock():**Unlocks the door if the player has the key.

3.3 Algorithms Involved

AI Pathfinding : The ghost will use a vector-based pathfinding algorithm to chase the player.

Collision Detection : For detecting when the player or ghost collides with walls or objects.

Line of Sight : The flashlight will create a cone of visibility using a mask and alpha blending.

Randomization : Key and treasure locations will be randomized each playthrough for replayability.

4. Statistical Data (Prop Stats)

4.1 Data Features

feature	**Why it is good to have this data? What can it be used for.**	**What will you obtain 50 values of this feature data?**	**Which variable (and which class will you collect this from?)**	**How will you display this feature data (via summarization statistics or via graph)?**
Time Played	Helps to understand how long players are engaged in the game. Can be used to improve game pacing and balancing.	Record the total time for each game session (50 rounds). (playing manually)	time_elapsed in GameManager class	Line graph : to show the trend of time played across multiple sessions.
Ghost Encounters	Measures how often players encounter ghosts. Can be used to balance difficulty.	Increment count every time the ghost detects the player (50+ times). (playing manually)	ghost_encounters in GameManager class	Bar chart : to compare frequency of ghost encounters between levels.
Treasures	Indicates how	Increment	treasures_colle	Pie chart : to

Collected	successful players are at collecting treasures. Can be used to adjust treasure placement.	count each time a treasure is collected (50+ times). (playing manually)	cted in Player class	show the percentage of treasures collected vs total available.
Flashlight Usage	Helps to analyze how important the flashlight is for player success. Can be used to adjust flashlight mechanics.	Record how long the flashlight is on during each session (50+ rounds). (playing manually)	flashlight_usage in Flashlight class	Line graph : to show how flashlight usage changes over time.
Deaths	Measures how often players fail. Can be used to adjust difficulty or AI behavior.	Increment every time the player is caught by a ghost (50+ events). (playing manually)	deaths in GameManager class	Bar chart : to compare death rates in different levels.
Rooms Cleared	Measures player progression. Can be used to track game completion rate.	Increment every time the player clears a room. (playing manually)	rooms_cleared in GameManager class	Bar chart : to show the number of rooms cleared per session.
Ghosts Defeated	Measures player performance and combat success. Can be used to balance AI behavior.	Increment every time a ghost is destroyed. (playing manually)	ghosts_defeated in GameManager class	Pie chart : to show percentage of ghosts defeated vs total encountered.
Keys Collected	Helps to see how players explore and interact with game	Increment every time a key is collected (50+ events). (playing	keys_collected in Player class	Bar chart : to compare key collections across different sessions.

	mechanics.	manually)		

4.2 Data Recording Method

Json File

4.3 Data Analysis Report

- Mean and median for time played, ghost encounters, and treasure completion.
- Standard deviation for flashlight usage and death rate.
- Completion rate for each and total game completion rate.

Present : Pie chart for game over (win/loss).

Bar chart for ghost encounters and treasure completion.

4.4 How will display each feature

4.4.1 Statistical Summary Table Plan

Feature	**Statistical Value**	**Graph Type**	**X - axis**	**Y - axis**
Time Played	Average, Min, Max	Line Graph	Game Sessions	Time (minutes)
Ghost Encounters	Average, Standard Deviation	Bar Chart	Game Sessions	Number of Encounters
Treasures Collected	Average, Max, Completion Rate (%)	Pie Chart	Percentage	
Flashlight Usage	Average	Bar Chart	Game Sessions	Usage Time (minutes)
Deaths	Average, Max	Bar Chart	Game Sessions	Number of Deaths
Rooms	Average,	Bar Chart	Game Sessions	Number of

Cleared	Completion Rate			Rooms Cleared
Ghosts Defeated	Average, Max, Defeat Rate	Pie Chart	Percentage	
Keys Collected	Average, Max	Pie Chart	Percentage	

4. Project Timeline

Week	**Task**
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission - Develop core player movement and flashlight system
3 (24 March)	- Implement AI pathfinding and ghost behavior
4 (31 March)	- Add treasure, key, and puzzle system
5 (7 April)	- Integrate sound and environmental effects
6 (14 April)	- Test and optimize the game - Submission week (Draft)

Weekly Plan

Week	**Plan**
26 March – 2 April	- Plan data features and game structure - Set up project folder and build player movement prototype

3 April – 9 April	- Develop Player, Ghost, and Flashlight classes - Implement flashlight logic and ghost pathfinding - Add collision detection and test basic gameplay
10 April – 16 April	- Build room/level layout and transitions - Implement treasure collection and doors - Start tracking gameplay data (JSON file)
17 April – 23 April	- Finish all data tracking and saving - Build graphs (line, bar, pie) and stat summaries
24 April – 11 May	- Debugging - Create full report with graphs and stats

50% Tasks (by 16 April)

- Player movement, flashlight, and ghost AI working
- Room transitions and map layout completed
- Basic treasure/door interaction
- Start logging key data points (time, flashlight, deaths)

75% Tasks (by 23 April)

- Full data tracking for all features (JSON file)
- Graphs for time, encounters, treasures built and working
- Table with statistical summary (average, SD, etc.) generated

Final 25% Tasks (by 11 May)

- Fine-tune AI, flashlight, etc.
- Bug fixing and usability improvements
- Polish visual layout and UI
- Finalize report with exportable charts and summary

5. Document version

Version: *4.0*
Date: *31 April 2025*

Date	Name	Description of Revision, Feedback, Comments
14/3	Pattapon	Don’t forget to change the title to your project name! Some details are still unclear, as mentioned in my comments. Also, you can remove the questions from the template. :)
16/3	Phiranath	Don’t forget to remove the templates, reformat the section number, add a time table, and change the document version after this review. Class Attributes and Methods are needed in section 3.2.
29/3	Pattapon	Good job. There are some suggestions in my comments.