International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

House of Fear

Aditya Sonwane & Rahul Naik
Department of Computer Science
S. S. & L. S. Patkar College of Arts & Science & V. P. Varde College

Abstract

Video games have developed into interactive digital experiences that combine exploration, decision-making, artificial intelligence, environmental design, sound, and real-time player interaction. Horror escape games use these elements to create suspense while encouraging players to observe their surroundings, solve problems, and make decisions under pressure. This paper presents the proposed design of House of Fear, a single-player 3D horror escape game in which the player is trapped inside a mysterious house and must find a way to escape.

The game focuses on exploring rooms and corridors, discovering clues, collecting useful items, solving puzzles, unlocking doors, and avoiding a hostile enemy. The proposed system brings together player movement, enemy AI, item collection, puzzle mechanics, house exploration, hiding, object interaction, audio and lighting, and game-state management. These modules are intended to function as one connected gameplay experience rather than as isolated features.

The development approach uses Unity as the game engine and C# for gameplay programming. Primary research may be conducted through questionnaires, observation, and playtesting to gather feedback about controls, puzzle clarity, enemy behaviour, atmosphere, and overall engagement. Secondary research considers game-design principles and player motivation, including exploration, achievement, competence, autonomy, and immersion. Actual evaluation results should be added only after playtesting has been completed.

The objective is to create a suspenseful and understandable escape experience while demonstrating practical concepts in game development, artificial intelligence, interactive 3D environments, and software-system integration. The proposed design can be expanded in the future with additional rooms, puzzles, enemy behaviours, difficulty settings, and story elements.

Keywords: Horror escape game, Unity, C#, enemy AI, puzzle system, 3D environment, item collection, player interaction, survival gameplay.

1. Introduction

Video games are interactive software systems in which players influence events through their actions and decisions. Modern games combine graphics, animation, artificial intelligence, physics, user interaction, sound, and gameplay rules. Horror games use these components to create tension, uncertainty, and a sense of danger. Escape games add problem-solving and exploration by asking players to discover information, collect objects, and complete objectives before they can progress.

House of Fear is a proposed single-player horror escape game. The player begins trapped inside a mysterious house and must explore its rooms, find clues and useful objects, solve puzzles, and unlock a route to freedom while avoiding a hostile enemy. Enemy behaviour, environmental lighting, sound effects, and interactive objects contribute to the suspense and challenge.

The project integrates several systems into one gameplay loop: Explore → Search for clues and items → Solve puzzles → Avoid the enemy → Unlock new areas → Escape. Exploration encourages observation, puzzles provide immediate objectives, enemy AI creates pressure, and game progression gives the player a clear goal. The project also provides an opportunity to apply Unity and C# programming to a practical interactive application.

2. Problem Statement

Horror escape games must balance suspense, clear objectives, responsive controls, and meaningful challenges. If players cannot understand what they need to do, puzzles can feel confusing rather than rewarding. If enemy behaviour is too predictable, the sense of danger may decrease; if it is too difficult, players may feel that success depends on chance rather than strategy. A disconnected environment, weak sound design, or limited object interaction can also reduce immersion.

House of Fear addresses these design challenges by proposing a connected escape experience in which exploration, item collection, puzzle-solving, hiding, enemy detection, and door unlocking support the same goal. The challenge is to make the house understandable enough to explore while preserving uncertainty and suspense. The project therefore focuses on integrating

responsive player controls, useful clues, clear interaction feedback, and enemy behaviour that creates pressure without making progress impossible.

3. Objectives

The primary objective is to design and develop a single-player 3D horror escape game that combines exploration, puzzle-solving, survival, and interactive enemy behaviour. The specific objectives are:

To create an explorable house environment with rooms, corridors, hiding places, and interactive objects.

To implement player movement and object-interaction controls using Unity and C#.

To develop enemy AI that can patrol, detect the player, and pursue the player according to the implemented rules.

To provide clues, keys, collectible items, puzzles, and locked doors that support game progression.

To use lighting and sound effects to build a suspenseful horror atmosphere.

To manage game states such as active play, pause, caught/game-over, and successful escape.

To evaluate controls, puzzle clarity, enemy difficulty, and player engagement through actual playtesting when available.

4. Primary and Secondary Research

4.1 Primary Research

Primary research for House of Fear focuses on obtaining direct feedback from potential players and observing how they interact with the game. It can be conducted through player questionnaires, interviews, gameplay observation, playtesting sessions, and feedback forms. This research should be performed after a playable prototype is available.

The research can examine whether players understand the controls, whether clues are noticeable, whether puzzles are fair, whether enemy behaviour feels challenging, whether hiding mechanics are understandable, and whether sound and lighting contribute to the horror atmosphere. Players can also be asked which parts of the game feel confusing, repetitive, or especially engaging.

Proposed questionnaire (use a 1–5 scale where 1 = Very Poor and 5 = Excellent):

1. How easy is it to understand the player controls?
2. How interesting is exploring the house?
3. How clear are the clues and puzzle objectives?
4. How challenging is the enemy AI?
5. How useful are hiding places and escape strategies?
6. How effective are the lighting and sound effects in creating suspense?
7. How enjoyable is collecting items and unlocking doors?
8. How fair is the overall difficulty?
9. How engaging is the gameplay?
10. Would you play the game again or recommend it? (Yes/No)

Note: Actual participant numbers, scores, charts, and conclusions must be added only after the survey or playtesting has been carried out. No research results are assumed in this paper.

4.2 Secondary Research

Secondary research examines existing horror and escape games, game-development documentation, and published work on player motivation and game design. Relevant design concepts include exploration, achievement, competence, autonomy, and immersion. These concepts help explain why players may enjoy discovering hidden areas, overcoming challenges, understanding game rules, and feeling that their actions affect progress.

For House of Fear, exploration is supported by a house containing rooms and objects to investigate; achievement is supported by completing puzzles and unlocking doors; competence is supported by learning enemy patterns and using items effectively; autonomy is supported by choosing where to search and when to hide; and immersion is supported by the environment, lighting, sound, and narrative context. These are design principles to guide development, not claims that player responses have already been measured.

5. Literature Review

Research on video-game motivation has considered how achievement, immersion, social interaction, and mastery influence player experience. Achievement-oriented play can involve completing objectives and overcoming obstacles. In an escape game, this can be represented through solving puzzles, finding keys, and opening new areas. Exploration-oriented play involves investigating unfamiliar environments and discovering useful information or hidden objects.

Self-Determination Theory has also been applied to understanding player experience. Autonomy concerns meaningful choices, while competence concerns the feeling of learning and becoming effective. In House of Fear, players can make decisions about which rooms to search, when to hide, and how to approach a challenge. They can develop competence by learning the environment, understanding clues, and responding to

enemy behaviour. Immersion can be supported by consistent environmental design and carefully used sound and lighting.

These concepts support the project's emphasis on connecting atmosphere, exploration, puzzle-solving, and enemy AI. The final design should be evaluated with players to determine whether these intended features are clear and engaging in practice.

6. Existing System

Existing horror and escape games commonly combine exploration, item collection, puzzle-solving, stealth, and enemy avoidance. However, games vary in how clearly they communicate objectives and how they balance enemy pressure with opportunities for careful exploration. Some designs may frustrate players when clues are difficult to notice, controls are unclear, or enemy encounters feel unfair.

For this project, the design challenges to consider include repetitive searching, unclear puzzle dependencies, unpredictable enemy difficulty, limited feedback when interacting with objects, and inconsistent use of sound or lighting. House of Fear aims to address these challenges through connected clues and items, readable interaction prompts, understandable enemy states, and an escape objective that links the gameplay systems. These are design goals and should be verified through testing rather than assumed to be proven advantages.

7. Proposed System

The proposed system is a single-player horror escape game in which the player is trapped inside a mysterious house and must find a way to escape. The player explores rooms, collects useful items, discovers clues, solves puzzles, and unlocks doors while avoiding a hostile enemy.

The major modules are:

11. Player Movement – controls movement, camera view, and basic navigation.
12. Enemy AI – manages patrolling, player detection, chasing, and enemy states.
13. Puzzle and Item System – supports clues, keys, collectible objects, and puzzle completion.
14. House Exploration and Interaction – allows players to investigate rooms and interact with doors, drawers, switches, and other objects.
15. Game Progression – manages objectives, pause and game-over states, and successful escape.

The modules form the gameplay loop: Explore → Collect items and clues → Solve puzzles → Avoid the enemy → Unlock doors → Escape. The exact features included in the final implementation should be updated to match the completed build.

8. Game Architecture

8.1 Player Layer

Player movement, camera control, interaction input, player status, and navigation through the house.

8.2 Gameplay Layer

Puzzle logic, item collection, locked doors, hiding mechanics, enemy AI, objectives, and game-state transitions.

8.3 World Layer

Rooms, corridors, furniture, interactive objects, clues, keys, hiding locations, lighting, and environmental audio.

8.4 Presentation Layer

Main menu, pause menu, interaction prompts, objective feedback, sound effects, visual effects, game-over screen, and escape screen.

8.5 Persistence Layer

Where implemented, stores selected settings and progress data using an appropriate local save format. If save/load is not implemented, this layer should be described as future work rather than an existing feature.

9. Game Modules

9.1 Player Movement

The Player Movement module controls how the player moves and looks around the house. It may include walking, running, camera control, and navigation through corridors and rooms. Responsive movement is important because players must investigate the environment and react when the enemy approaches.

9.2 Enemy AI

Enemy AI controls the hostile character's behaviour. A basic state model can include Idle or Patrol → Detect Player → Chase → Search or Return to Patrol. Detection can use distance and, if implemented, line-of-sight checks. The enemy's speed, detection range, and recovery behaviour should be balanced through testing so encounters remain tense but understandable.

9.3 Puzzle and Item System

The puzzle and item system allows players to discover clues, collect keys or tools, and use them to solve

challenges. A puzzle can define its required item or condition, completion status, and effect on the environment. For example, finding a key may allow the player to open a particular locked door. The dependency between clues, items, and objectives should be clear enough for players to make progress through observation.

9.4 House Exploration and Interaction

The house exploration module provides rooms, corridors, objects, and hidden locations to investigate. The interaction system enables the player to examine or use supported objects such as doors, drawers, switches, keys, and hiding places. Clear visual or text feedback can help players understand when an object is interactive.

9.5 Game Progression

The game progression module manages the player's objectives and the conditions for success or failure. Progress may involve collecting required items, completing puzzles, unlocking a route, and reaching an exit. The game can transition between active play, pause, game-over, and escape-complete states. Only the states implemented in the final build should be documented as completed features.

9.6 Audio and Lighting System

Audio and lighting contribute to the horror atmosphere and provide environmental feedback. Footsteps, door sounds, ambient audio, and enemy cues can help players interpret events, while limited lighting and shadows can increase uncertainty. These effects should support navigation and tension without obscuring essential clues or making the game unfair.

10. Puzzle System

A puzzle provides a structured obstacle that the player must resolve to open a route or obtain an item. Each puzzle can include an initial state, clue, required action or item, completion condition, and result. For example, a clue may direct the player to locate a key or activate a switch. Puzzle dependencies should be tested to ensure that the player can complete them in the intended order and that required items are not inaccessible.

11. Enemy Behaviour System

Enemy behaviour can be organised as a state machine so that the character changes behaviour in response to game conditions. In a basic implementation, the enemy patrols an area, detects the player when a detection condition is met, pursues the player, and returns to searching or patrolling when the player is no longer detected. Optional behaviours such as investigating sounds or checking hiding places should be described only if they are implemented.

Enemy difficulty can be adjusted through detection range, movement speed, chase duration, and the distance at which the enemy gives up pursuit. Testing should verify that the enemy can navigate the environment, does not become stuck in geometry, and gives the player a reasonable opportunity to respond.

12. Item Collection and Inventory

The item collection system allows players to pick up useful objects such as keys, clues, tools, or puzzle components. The inventory, if included, tracks which items have been collected and whether they can be used. Item interaction should provide feedback when an object is collected and when an item is required but unavailable. This module connects exploration with puzzle-solving and progression.

13. Hiding System

The hiding system, if implemented, gives the player a way to reduce the chance of being detected during an enemy encounter. Hiding locations may include wardrobes, cupboards, or other suitable objects in the environment. The system should define when the player can enter and leave a hiding place, how the camera behaves, and whether the enemy can detect a player who is hiding. Its rules should be consistent and communicated through gameplay feedback.

14. Interaction System

The interaction system handles player actions involving supported objects. It can use a raycast or trigger area to identify an object within reach, display an interaction prompt, and invoke the corresponding action when the player presses the interaction key. Actions may include opening a door, collecting an item, examining a clue, operating a switch, or entering a hiding place. Interaction logic should be modular so that new objects can be added without rewriting the entire player controller.

15. Game-State and Save System

The game-state system controls transitions such as starting the game, pausing, resuming, being caught, restarting, and escaping. A save system may store progress or settings if persistence is part of the implementation. For a small Unity project, local JSON

files or PlayerPrefs may be suitable depending on the type of data. PlayerPrefs is generally more appropriate for simple settings than for complex structured game progress. If save/load functionality is not present, it should be listed as a possible extension rather than claimed as complete.

16. Methodology

The development methodology consists of the following stages:

Stage 1 – Requirement Analysis: identify the target player experience, essential gameplay systems, controls, environment needs, and expected outcomes.

Stage 2 – Game Design: plan the house layout, gameplay loop, puzzles, clues, items, enemy states, and escape condition.

Stage 3 – Prototype Development: implement basic movement, camera control, object interaction, and a small test environment.

Stage 4 – System Development and Integration: add the enemy AI, puzzle and item logic, doors, audio, lighting, and game-state management.

Stage 5 – Testing: check controls, collision, puzzle dependencies, enemy navigation, game states, and performance.

Stage 6 – Refinement: correct bugs, improve interaction feedback, balance enemy difficulty, and adjust the environment based on observed issues.

Overall flow: Research → Requirements → Game Design → Prototype → Implementation → Integration → Playtesting → Refinement.

17. Research Methodology

Primary research can be conducted through player questionnaires, observation, and playtesting once a playable build is available. Feedback should focus on control usability, clarity of clues, puzzle difficulty, enemy behaviour, atmosphere, and overall engagement. The sample size, participant profile, and actual responses should be recorded accurately.

Secondary research can include published work on player motivation, game design, horror-game conventions, Unity documentation, and relevant technical resources. Secondary sources inform the design choices, while primary feedback can be used to assess how well the implemented game meets its goals. No conclusions about player satisfaction should be made before collecting and reviewing real feedback.

18. Algorithms and Processing Techniques

18.1 Enemy Detection

A simple distance-based detection condition can be expressed as: Distance = $\sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)}$. If the distance between the enemy and player is less than or equal to the configured detection range, the enemy may enter a chase state. A line-of-sight check can be added if it is supported by the implementation.

18.2 Item Interaction

If an interactive object is within the player's interaction range and the player presses the interaction key, the game checks the object's type and performs the associated action. For a locked door, the system checks whether the required key or puzzle condition has been satisfied before changing the door state.

18.3 Puzzle Completion

IF all required puzzle conditions are satisfied → Set puzzle status to Completed → Trigger the result (for example, reveal an item or unlock a door) → Update the current objective. The actual conditions depend on the puzzle designed for the game.

18.4 Enemy State Machine

Patrol → Detect Player → Chase → Search/Return to Patrol. The precise transitions depend on detection, visibility, distance, and the enemy logic implemented in Unity.

18.5 Game Progression

Collect Required Items → Complete Puzzle Conditions → Unlock Route → Reach Exit → Set Game Status to Escaped. If the enemy catches the player, the game may instead transition to the game-over state.

19. System Implementation

The game can be implemented as a collection of interconnected Unity components. Potential components include Player Controller, Camera Controller, Enemy Controller, Enemy State Manager, Interaction Manager, Item or Inventory Manager, Puzzle Manager, Door Controller, Audio Manager, UI Manager, and Game Manager. The final component list should reflect the actual scripts and systems used in the project.

A modular approach makes it easier to test and maintain individual systems. For example, a new puzzle can be added through the puzzle and interaction logic without redesigning player movement. Similarly, enemy parameters can be adjusted separately from the UI. Unity scenes, prefabs, colliders, triggers, and C# scripts can be

used to organise the environment and gameplay behaviour.

20. Primary Testing and Evaluation

The game should be evaluated through structured testing. Technical tests should check player controls, collision, object interaction, item collection, puzzle completion, door states, enemy navigation, and game-state transitions. Playtesting should examine whether players understand the objectives and whether the horror atmosphere supports rather than blocks gameplay.

Suggested evaluation parameters:

Player movement – responsiveness and ease of use.

Exploration – whether rooms and important objects are understandable.

Puzzle system – clarity, logical consistency, and completion reliability.

Enemy AI – detection, pursuit, navigation, and perceived difficulty.

Audio and lighting – contribution to atmosphere and useful feedback.

Overall engagement – player interest and willingness to continue.

A five-point scale may be used: 1 = Very Poor, 2 = Poor, 3 = Average, 4 = Good, and 5 = Excellent. Actual scores should be collected from participants and inserted into the report; blank or unmeasured values should not be presented as results.

21. Results and Discussion

The intended outcome is a playable horror escape experience in which exploration, item collection, puzzles, enemy AI, and escape progression work together. The evaluation should determine whether players can understand the controls, identify useful clues, complete puzzles, and respond to enemy encounters. Technical testing should also confirm that interactions and game-state changes work reliably.

At the time this document is prepared, no player-test scores or measured results have been provided. Therefore, quantitative results cannot be reported here. After testing, this section should include the number of participants, test procedure, measured feedback, observed problems, and changes made as a result. The discussion should distinguish observed findings from intended design goals.

22. Advantages of the Proposed Game

Integrated Gameplay: exploration, puzzles, items, enemy behaviour, and escape progression support a shared objective.

Problem-Solving: players must examine clues and use items to progress.

Suspense and Survival: enemy detection and pursuit create pressure and encourage planning.

Interactive Environment: doors, drawers, switches, and other objects can make the house feel responsive.

Expandable Design: additional rooms, puzzles, items, enemy behaviours, and story elements can be added later.

Practical Learning: the project demonstrates Unity, C# programming, AI logic, 3D environment design, and system integration.

23. Limitations

Developing multiple connected gameplay systems requires time for implementation, debugging, and balancing. Enemy AI may initially have a limited set of behaviours, and puzzle difficulty can be hard to balance without player testing. Large or detailed environments can affect performance, particularly if they contain many objects, lights, and sound sources.

The quality of the final experience depends on the implemented feature set and the amount of testing performed. If save/load, advanced sound detection, multiple endings, or complex enemy behaviours are not implemented, they should not be presented as completed functionality. Primary research findings also remain unavailable until actual players participate in testing.

24. Future Scope

Additional Rooms and Areas: new rooms, hidden passages, basements, and alternative escape routes could expand exploration.

Advanced Enemy AI: enemies could investigate sounds, search hiding places, or use more varied search patterns if supported by the design.

Additional Puzzles: multi-step puzzles and interconnected clues could increase variety.

Difficulty Settings: detection range, enemy speed, clue availability, and puzzle complexity could be adjusted for different players.

Expanded Story: notes, environmental storytelling, and additional narrative events could explain the house's history.

Improved Persistence: checkpoints or a structured save system could preserve progress across sessions.

Performance and Accessibility: graphics options, control remapping, subtitle options, and performance optimisation could improve usability.

25. Conclusion

This paper presents the proposed design of House of Fear, a single-player 3D horror escape game focused on exploration, puzzle-solving, survival, and enemy AI. The player must investigate a mysterious house, collect useful items, discover clues, solve puzzles, unlock doors, and avoid a hostile enemy in order to escape.

The proposed system integrates player movement, enemy behaviour, puzzle and item management, house exploration, interaction, audio and lighting, and game progression into one connected gameplay loop. Unity and C# provide the development environment for implementing these systems. The project offers practical experience in interactive game design, programming, 3D environment construction, and system integration.

The quality of the completed game should be assessed through technical testing and actual player feedback. Future development may introduce additional environments, more advanced enemy behaviours, extra puzzles, and expanded story elements. By combining suspense with understandable objectives and responsive interactions, House of Fear aims to provide an engaging horror escape experience while demonstrating applied computer science and game-development concepts.

26. References

[1] Yee, N. “Motivations for Play in Online Games.” CyberPsychology & Behavior. Relevant to player motivation, achievement, social interaction, and immersion.

[2] Ryan, R. M., Rigby, C. S., & Przybylski, A. “The Motivational Pull of Video Games: A Self-Determination Theory Approach.” Motivation and Emotion, 2006.

[3] Mekler, E. D., Brühlmann, F., Tuch, A. N., & Opwis, K. Research on player motivation and game enjoyment.

[4] “Individual differences in motives, preferences, and pathology in video games: The Gaming Attitudes, Motives, and Experiences Scales (GAMES).” Frontiers in Psychology.

[5] Unity Technologies. Unity Manual and Scripting API documentation. https://docs.unity3d.com/ (accessed during project development as applicable).

Note: Verify the full bibliographic details and access dates for all references before final submission. Add any specific sources actually consulted during development.