Phase 4 Report | CMPT 276 Project Group 5

Group members: Amar Mondair, Karanveer Singh, Kiran Khubbar, Prabhjot Singh

The Game:

Haunted Hotel is a top-down 2D action-adventure game in which the player takes on the role of a ghost hunter hired to clear a haunted hotel infested with dangerous ghosts. The player begins the game in the hotel lobby, where an NPC/ Hotel Manager explains the mission and the objective that is to collect 2 keys from 2 levels above the lobby and unlock the basement door to defeat the final lava monster to win the game. The levels are interconnected, meaning you can't exit a level without collecting a key from that level, and you need the money that ghosts drop when you kill them. Each level includes a unique enemy, punishments, and challenges such as projectiles fired by enemies.

The navigation controls are "WASD" and can engage with enemy either by a melee weapon by hitting "E" or shooting a counter projectile that is an electric ball by pressing "Space Bar". Enemy inflicts damage on contact and decreases the number of hearts players have left. Various environmental threats like Bomb Gifts on Evil Santa level and Fire punishment in the final level increase the difficulty.

Level 1 introduces Michael Myers that patrols the level throughout. It requires precise movement and strategy to collect the key. After completion of the task the player teleports back to lobby where then player proceeds to Level 2 where Evil Santa awaits with his menacing red eyes with a Christmas themed map, and this enemy also fires snowballs that cause damage. After collecting 3 gifts that contain money and the key while avoiding bomb gifts, the player exits Level 2 and now the Player is ready to face the Final boss in the basement that is twice as powerful as other enemies and shoots fireballs. And sets fire punishments throughout the map. The winning condition is to beat this boss.

The game combines combat, item collection, and strategic movements. Different theme-based enemies offer variety in gameplay. Haunted Hotel gives a challenging experience to the player, and we hope that you enjoy the game.

Comparison between Ideation and Final Product:

The final version of the game maintains our core vision from the Phase 1 Plan, but to create the game that we wanted, several major changes were made during the coding/ Phase 2 process.

The original UML diagram didn't include separate NPC or Projectile classes. These were later added as independent systems to support proper dialogue interactions and ranged combat. NPCs are now managed as their own entities for cleaner interaction handling.

The decision to add the A* search algorithm with Manhattan Distance Heuristic was one of the best for us. Before the enemies were using random and it was easy to defeat them.

Originally, we decided to include a separate Punishment class to handle traps and punishments. This idea was scrapped and instead in game punishments were added such as Bomb gifts and fire tiles which function as environmental hazards through collision checker.

There was also a Rewards directory in our UML diagram which was also replaced by the Objects directory, which contains all collectables and projectiles.

The Tile directory had Board and Barrier classes which were removed and now it only contains Tile.java, that is a default tile object which is used by TileManager.java that contains an array of Tile objects with different tile sprites in them.

Finally, the Main directory, originally, we only added GamePanel, Main and KeyHandler. During development, we realized that several additional classes were required. A few of them are mentioned below:

- **AssetSetter**: Sets assets such as Maps, NPC and monsters.
- **EventHandler:** Checks events like punishments or teleportation.
- **CollisionChecker:** Handles collision detection.
- **UI:** Manages all the in-game displays, pause screens and dialogues.

Lessons Learned:

This project helped our team better understand the importance of modular design, testing, and team coordination. One of our biggest challenges was managing merge conflicts when integrating multiple versions of core files. We assigned a person to integrate all the changes made by everyone. By assigning a main integrator, we learned the value of having a clear version-control workflow.

We also learned that automated testing is essential, especially for systems like AI, collision detection, and event handling. Many bugs that were difficult to detect during gameplay were easily identified through unit and integration tests.

Supporting Files:

Demo Video: https://www.youtube.com/watch?v=XzkpSPnscG4

JAR File: ProjectTeam5/HauntedHotel/HauntedHotel-1.0-SNAPSHOT.jar

Javadocs: ProjectTeam5/HauntedHotel/HauntedHotel-1.0-SNAPSHOT-javadoc.jar