Adventure Game Report

This report will provide an overview of the adventure game, including its design, functionality, and testing.

Design

The adventure game is an interactive text-based game that allows a player to navigate through a series of rooms in an abandoned castle, collecting items and encountering obstacles along the way. The game is object-oriented, with three main classes: Game, Player, and Room.

The Game class manages the overall flow of the game, including the player's movements and the display of the current room. The Player class keeps track of the player's inventory and current location within the game. The Room class represents each location in the game, including its name, image, and connections to other rooms.

Functionality

The game begins in the entrance of the abandoned castle, where the player is presented with four options: north, south, east, and west. The player can choose which direction to go by typing the corresponding command.

As the player moves through the rooms, they may encounter obstacles or find items. For example, in the gallery room, the player must defeat a witch in order to obtain a golden key. In the scullery, the player can find a knife, which is necessary for defeating the witch.

If the player successfully navigates through the rooms and collects the necessary items, they can eventually rescue the lost puppy and win the game.

Testing

To ensure the game was functioning properly, we implemented automated testing using the unittest module in Python. This involved writing test cases for each of the game's functions, including the player's movements and item collection.

We also manually tested the game by playing through it several times, trying different combinations of actions to ensure that all possible paths were functioning correctly.

Conclusion

Overall, the adventure game is a fun and engaging experience for players. Its object-oriented design and automated testing allow for smooth gameplay and minimal bugs.

In terms of design, the use of separate classes for the game, player, and rooms helps to organize the code and make it more modular and scalable. For example, if we wanted to add additional rooms or items to the game in the future, it would be straightforward to do so by simply creating new instances of the Room class and modifying the connections between them.