The Game:

In a dark and desolate land, our hero finds themself lost in a strange and unfamiliar castle; but they are not alone. Following him is Count Dracula himself and he is hungry.

Our game is a turn-based, arcade-style 2D game where the player must navigate around multiple barriers and floor-traps (punishments) as an enemy entity follows. The player must not get touched by the enemy as they will die and the game will be over. To win, the player must locate and collect a certain amount of keys to unlock the exit door to escape. Collecting the keys will increase the player's score by one. Once the keys have been collected, the door will open and the player has the option to leave. There are bonus keys as well (colored golden) that when collected, increase the player's score by 2. These golden keys change position over a certain amount of time, so the player must act hastily when trying to collect them. Strictly collecting bonus keys will not open the door, only collecting the main silver keys will. There are multiple 'spiked' traps laid on the floor. If the player were to step onto one of those, their score would decrease by 1. If a player's score goes negative, the player loses. When the player reaches the door, an end screen will appear displaying the amount of time played and the player's score. A restart button is also displayed and will restart the game from the beginning. When the enemy catches the player, a death screen will appear and the player once again has the option to restart from the beginning.

For the class structure for our game, we were somewhat faithful to our original UML class diagram. The class hierarchy for the Character, Player and Enemy classes was kept along with many of its methods and fields. Some other methods were added to these classes during implementation, one of which was the tick method that plays a huge part of the game's functionality.

The major change to the game's class structure was the removal of all the 'Cell' classes and its hierarchy. To display the cells, we instead implemented a Background class that held the game's map within a 2D array of different types of Tiles (punishment, reward, bonus, barrier, walls, floor, etc.). We added another class called Board which was responsible for the interactions between the entity classes (Player/Enemy) and the Background, as well as listening for keyboard events. Another class called Game was added to run the Board class and display start/end screens. Other minor classes that were added included Sound which held the soundtracks of multiple audio files and communicated with the Board class and Tiles aka cells which held images.

For our statement PDF during the design phase, there were only a few minor changes which included the change in map size and the removal of some aspects such as defeating enemies, level changes, and powerups. For our Use Cases, we realized that they were done a bit incorrectly, and we relied mostly on the class diagrams and the statement to implement our game.

There were many things learned when creating this project. Firstly, the importance of having a good design before implementation was learned the hard way for us, as there were times where a few of us were working on some aspects of the game that couldn't function

properly with each other or some of us waiting for one aspect to be implemented first before starting work because they needed to know how the first part functioned first. Another aspect was the importance of communication, as some of the aforementioned problems could have been avoided if we had communicated just a little bit more effectively. The importance of using Git was also learned as well because we avoided problems like code being deleted by reverting the branch and allowed members of our group to work more freely by creating another branch and merging them afterwards. In the later parts of the project, we learned the importance of designing and writing the code with the thought of keeping a tight coupling and high cohesion, as it could make refactor changes easier and overall make a cleaner project. We've learned alot about what goes into testing software, as all of us had not learned about testing yet and we had to test this relatively big project. Lastly, we have gained much needed experience in not only coding and its principles but with working and collaborating as a team.

Video:
https://youtu.be/V1xb_FZ7JIg