# Report For Hunt the Wumpus:

## Summary and Program Functionality

The Hunt the Wumpus game is a text-based game created in the 1970s. This involves the player moving through a series of caves that are connected, in the shaped of a dodecahedron. The player’s aim to is hunt a monster called the “Wumpus”. In progression of the game, there are aspects that the player must attempt to avoid, such as “super bats” that may bring the player around the map, and bottomless pits. The player has “crooked arrows” which can be used to kill bats and the Wumpus. The game is won by the player shooting the Wumpus and is lost if the player is eaten by the Wumpus, falls into the bottomless pit, or has no more arrows.

To extend this base implementation of the game, further additional features were implemented. The most major of them being the implementation of a graphical user interface to interact with the program, so the game can be played as either a text-based version or using a Graphical User Interface.

Including this, the number of Wampuses have been doubled, and the Wumpus has the ability to move (When the player smells the Wumpus, but misses shot), both to add an extra element of difficulty to the game. There is also a feature where there are arrows scattered around the map, and the player can pick these arrows up if they come across a cave that contains these arrows. The game will therefore not end if the player runs out of arrows, while there are still uncollected arrows in the cave system.

## Major Design Decisions

### Script Design

For classes, it was decided that entities that was cause actions themselves, would be required to have individual classes that would interact with other classes.

The Main class is what holds the main method and has an instance of every other class to be used. This is where the game is run, and all the logic is called from other classes.

The Game class is used to hold the methods that affect the gameplay and the aspects of the game that are not actions by other entities within the game. This includes features such as checking whether the player is near any of the warnings and displaying the warnings on screen.

The Player class is designed to hold all of the methods that correlates to the actions of the player, that can influence the game, such the shootArrow method to contain the code logic for how the player shoots an arrow.

This is a similar way to how and why the Wumpus and the Bats classes have been implemented, as these are different entities, with their own actions that can influence the game. A Wumpus class was also necessary for the fact of having 2 Wumpuses, which would have its own individual location and death status.

///ADD OTHER CLASSES USED///

Below is the UML table created to represent the final program. One UML program is for the text-based implementation of the game, while the second is to represent the necessary classes for the Graphical User Interface for the game.

///ADD THE UML DIAGRAM WITH CAPTIONS///

### Game Logic

//WHY WAS JSON FILE USED FOR THE CAVE CONNEXTIONS?

//WHY DO WE USE -1 TO REPRESENT ENTITIES THAT DON’T EXIST

// Discussion on how the game loop and conditions contribute to the overall gameplay experience.

//WHY IS THERE A GAME OBJECT IN THE PLAYER CLASS

## Bugs or Defects

## Sample Gameplay Scenarios

## Version Control Explanation

The version control used to produce this project was a private repository on GitHub, where both members were added as contributors to the repository. The link to the repository is below:

<https://github.com/Dod900ls1/HuntTheWumpus>

The method to how the workflow was split is by classes and methods. The idea of functional abstraction was used, where it was confirmed what values and the datatypes of these were to be inputted and the return values were confirmed beforehand, while the actual logic behind how the method is implemented is not needed to be known by the other member. This allowed for seamless code development, where which part of the code to work on for each member was determined.

## Manual and Unit Tests

## Additional Features

## Readability and Documentation

## Individual Contribution

The classes that I was assigned to work on were the Player class, Wumpus Class, Bat class, the Main Class, some aspects of the Game class.

## Conclusion

Overall, the game produced is a successful implementation of the original “Hunt the Wumpus program”, with several aspects of additional functionality that has multiple different difficulty levels.

//MENTION ABOUT HOW GUI WAS THE MOST DIFFICULT ASPECT TO DEVELOP?

From this development process, the idea of functional and procedural abstraction was developed and understood, with the idea that two developers don’t need to know how the other developer’s program works if the return type and arguments passed in are explicitly stated and confirmed. This allows for a more seamless workflow, where both developers can work at their own pace and independently, while the development of code still continues to occur.

## Reference Table

## Note

## Appendix