Implementation

3.1 GUI

As our multiplayer implementation is turn based (as part of system requirement 18d), only minimal changes were made to the single player GUI. To access the multiplayer gamemode, a simple button was added to main menu with consistent styling. When clicked the first player chooses their detective in the same way as in solo gameplay (as part of system requirement 4a), followed by the second player. The second player must select a different detective from the first player.

During gameplay, both players use the same GUI. The only differences are the change in detective sprite displayed, the display of both player scores instead of just one, and the players' remaining action points.

3.2 Scoring

One of the major modifications to the inherited project is the change in scoring system as described in our Final Architecture and Traceability Report [1] in section 1.3.3. Instead of giving a score based on the number of items collected multiplied by a factor of time taken, the score is now mostly based on time, with additional points given and deducted for finding clues and successful/unsuccessful accusations with new methods `GameMaster.ClueCollected()`, `GameMaster.GivePoints(Float:points)`, `GameMaster.TakePoints(Float:points)`, and updated method `GameMaster.Update()`. We chose to make this change in accordance with the additional requirements introduced in the scenario brief (in particular, requirement 19a) in our new Requirements Table [2]. Similarly, the leaderboard system was overhauled as the inherited version only supported 3 players, so it was replaced with an arcade-style name/score sorted list (as part of functional requirement 14a). This is handled by a new method in `Leaderboard.GetScores()` which reads the new format of the leaderboard text file.

3.3 Multiplayer

Each player is given a number of 'Action Points' they can do each turn (as part of system requirement 18d), denoted as the private constant `GameMaster.TURNS_PER_GO`, displayed in the top left with a clear label, 'Actions Remaining'. An action is defined as talking to (or accusing) a character, moving between rooms or collecting a clue, according to our User Manual [3]. An alternative option to this implementation would be to base the turns around time; each player gets a set amount of time per turn, and when that time elapsed, the next player would take their turn. We chose not to do this as it would not be compatible with either the new or the old scoring methods as each player would have the same scores and take the same amount of time. An action point is deducted each time the new method `GameMaster.UseTurn()` is called, which returns False if the current player has no remaining turns, prompting a call to new method `GameMaster.SwitchPlayers()` to alert the user to switch players. Other methods and structures are used to handle using Action points and switching detectives, as described in our Architecture and Traceability Report section 1.3.1.

In a multiplayer game, the players share an inventory of the clues collected (as part of requirement 18e). However, only the player who collects the clue receives points. We decided this would be the most enjoyable method of clue collection because otherwise both users would be collecting the same clues, which would lead to playing two single player games simultaneously.

3.4 Riddles

Our new requirement 20 suggests a locked room is needed in our game, which can be opened by passing a challenge. The previous group (Team Watson) already implemented a basic version of this idea, however we decided to remove it entirely and implement our own system described in detail in our Architecture and Traceability Report section 1.3.2. [1] We chose to implement a riddle as the puzzle to unlock the the locked room instead of using the previous system of finding a key, as this felt too easy. The riddle adds more depth to the game and makes it more interesting and engaging (as part of non-functional requirement 25). It is also different to the rest of the gameplay, whereas a key is collected just like another clue. Also, as the inventory is shared in our multiplayer game mode, one player collecting the key would grant access for both of the detectives. Another change we made was making the allocation of the locked room random (as part of system requirement 1). The previous system always had one room (the *Control Room*) as the locked room, while our locked room (as `GameMaster.lockedRoomIndex`) is chosen at random when a new game begins in `GameMaster.Start().`

For extensibility, the riddles are stored in a JSON file. A JSON Handler has been created to deal with loading in data from this file, while a `Puzzle` class was added to store details of a riddle. We added a new Unity Scene *Puzzle* as a transition between the room the character was previously in and the locked room, which loads a new instance of the `Puzzle` class consisting of the riddle text, a correct answer and a set of wrong answers that can be as long as desired (as part of system requirement 20a). The riddle, along with the answers, is displayed in our scene for the player whose input is manages by Unity Button components and the new `PuzzleScript.IsCorrect(int)` method which individually grants players access to the locked room if they answer successfully.

3.5 Game Over

The game ends once a player accuses the correct character of the murder using the correct evidence (as part of system requirement 12a). In single player, the score is shown in the centre of the screen, with a box below it to allow the player to enter their name (as part of system requirement 13b). In the new multiplayer mode, both scores are displayed along with two boxes to for the players to enter their names (as part of system requirement 13a). The technical details and changes to architecture can be viewed in our Architecture and Traceability Report section 1.3.3.

It should be noted that it is no longer possible for a player to lose the game (in the inherited project, accusing the wrong character, or with an incorrect set of clues would cause the player to lose the game). We decided to get rid of this condition (user requirement 17) for the sake of multiplayer. If one player were to lose the game, the competitive element would be lost. Similarly, if both players were to lose the game, then the game would end in a draw, once again removing the competitive element. Instead, we replaced the ‘lose’ condition with the loss of points.

3.6 Sound and Vision

The rest of the changes made to the game can be regarded as ‘superficial’ or ‘polish’. This includes appropriate menu music, ambient room sounds as well as assorted sound effects (including for collection of clues or answering the riddle). These were added to make the game more enjoyable and immersive (as part of non-functional requirement 25). As many of these sounds required attribution, their creative commons details can be seen in a new panel by clicking the new ‘Credits’ button on the main menu. Background images have been also been updated to feature textures rather than block colours and other extra details, such as animated smoke, once again with the purpose of making the game more immersive.

Bibliography

[1] H. Cadogan, S. Davison, T. Fox, W. Hodkinson, C. Hughes and A. Pearcy, "Final Architecture and Traceability Report", *Wedunnit!*, 2017. [Online]. Available: http://wedunnit.me/webfiles/ass4/AT4.pdf . [Accessed: 02- May- 2017].
[2] H. Cadogan, S. Davison, T. Fox, W. Hodkinson, C. Hughes and A. Pearcy, "ReqTable.pdf", *Wedunnit!*, 2017. [Online]. Available: http://wedunnit.me/webfiles/ass4/ReqTable.pdf . [Accessed: 10- Mar- 2017].
[3] H. Cadogan, S. Davison, T. Fox, W. Hodkinson, C. Hughes and A. Pearcy, "ReqTable.pdf" *WeDunnit: Homicide in the Hub User Manual*, 1st ed. Wedunnit, 2017, p. 2. Available: http://wedunnit.me/webfiles/ass4/UM.pdf . [Accessed: 02- May- 2017].