Implementation

Change list & justification

Add new class ‘PlayerSwitchScreen’

- Our new requirement [2.1.5] states that the game must be turn based. This means that the players will have to switch between who is playing the game, as the game will only be running locally. During this time, and to prompt the players to switch over, this new screen is displayed.
- It includes a title saying: “Time to switch players!” and a button to switch to the next player.
- It includes a leaderboard that shows the top scoring players, so players can see who is winning.
- This class extends ‘AbstractScreen’

Add new class ‘NumberOfPlayersSelectionScreen’

- The new requirement [2.1.5] states that the game must be turn based. To complete this, there needs to be two or more players.
- This screen allows the user to set how many people will be playing. It includes a slider from 2 to the maximum amount of players, which can be set by changing a constant. It also contains 2 buttons, one to start the game and one to return to the main menu.
- The initial requirement was for 2 players to be able to play the game, however the way we had implemented the code was dynamic enough to allow us to have as many players as we’d like.
- We set an arbitrary limit of 10 players and allowed the user to choose the number of players as this was minimal effort. We set the maximum to 10 as we felt that to be a suitable number based on the type of game. We felt this method was much better than just 2 player turn-based as the game would be much more competitive.
- A new button ‘multi-player’ was added to the main menu to navigate the player to this screen.
- It extends `AbstractScreen`

Add new class `ScenarioBuilder`

- The new requirement [2.1.5] states we need to add turn based multiplayer. This means the game scenario will need to be the same across all players, however one player's actions shouldn’t affect another player's game.
- This class generates the game state, including the clues, NPCs, and murderer/motives.
- This class is used to build the games for each player that is playing. It generates instances of the `GameSnapshot`, explained below.

Add new class `GameSnapshot`

- GameSnapshot is used to hold the instance information for each game. It was created to allow us to complete requirement [2.1.5].
- The snapshot stores information that will need to differ for each player. This includes Clues, NPCs, location etc, as these have player dependent properties.
- As each game needs to be the same (same killer/clues), the GameSnapshot instances start as duplicates - with the same values, but in different memory locations.

GUI changes

- Unlike the project we picked up in assessment 3, this one had the required dependencies for the use of TrueType Fonts. This allowed us to have far more interesting fonts in the game, especially for our titles.
- We also implemented new skins for our buttons, labels, sliders and checkboxes to give the game a more refined look.
- No architectural changes were required for this as we simply edited the Assets and UIHelpers classes that already existed.

Music changes

- We disabled the music [1.3.3] by default, this means that the player has to enter the “Settings” menu and manually untick the “Muted” checkbox to enable the music.

Secret Room Implementation

- The new requirements [4.1.4] require the game to include a ‘secret room’
- The new room was made with Tiled, using the same tileset as existing rooms.
- The transition to the room takes place when the puzzle is solved.
- Logic was added to check for the rendering of the secret room. When this was detected, an image overlay was added on top of the room to create a ‘torch in a dark room’ effect.
- This image was created by us and was fully black aside from a transparent circle in the center. As the game is centered around the player, this allowed the player to only see the few tiles around their character thus giving the room a more mysterious feel, we determined that this was in line with the theme of the game.
- Logic was added to the random NPC distribution process to ensure that no NPC spawned in the secret room, as we wanted the room to be an extra feature and not something that was essential to the completion of the game.
- No significant architectural changes were required, as no methods or classes were required for the implementation of the secret room.

Puzzle implementation

- One of the new requirements [4.1.5] we were given in this assessment was the implementation of a puzzle that had to be solved before being able to enter a secret room. We decided on a puzzle in which the player must click three books in a book case in the correct order to unlock the room. The idea took inspiration from the classic false books often seen in films.
- Each player gets to solve the puzzle, however the puzzle differs per player.
- For this we implemented a “Puzzle” class which handles the logic for the puzzle. The bookshelf tiles were given a boolean “secretRoom” property, and one of the bookshelves in the game are randomly chosen from the possible values when the game is generated.
- When the player clicks on the correct bookshelf in the game, a transition screen is generated. Following which, a puzzle screen is generated containing 9 buttons. 3 of the buttons are randomly assigned as the correct ones and if all 3 are clicked in a row then the secret room is revealed, if any of the erroneous buttons are clicked then the puzzle resets itself.

Extra Score

- To make the secret room [4.1.4] more interesting, we placed an item in it that provided extra points to the player that found it. When the player interacts with the extra score item, they obtain a random score between 300 and 700 points.
- Although the extra score functionality wasn’t requested by the client, we felt that its addition was necessary, as it provided a purpose for players to visit the secret room.
- The implementation of this involved adding a property called “Extra Score” to the item tile to provides extra points. In this case the item was a tile that looked like a cash pile.
- Methods were added to the `Room` class to search through the tileset of the room each time a room was initialised. These searched for the “Extra Score” property and added the locations of any tiles with said property to a List.
- Logic was then added to check whether an item had this property every time an interaction was made, if it did then the extra score would be added to the player’s score and some dialogue would appear.
- Further logic was then added to disable the item after this interaction took place so that no further score could be obtained by any player. This disabling logic meant that in a multiplayer scenario, only the first player to find the cash pile would get the extra points.