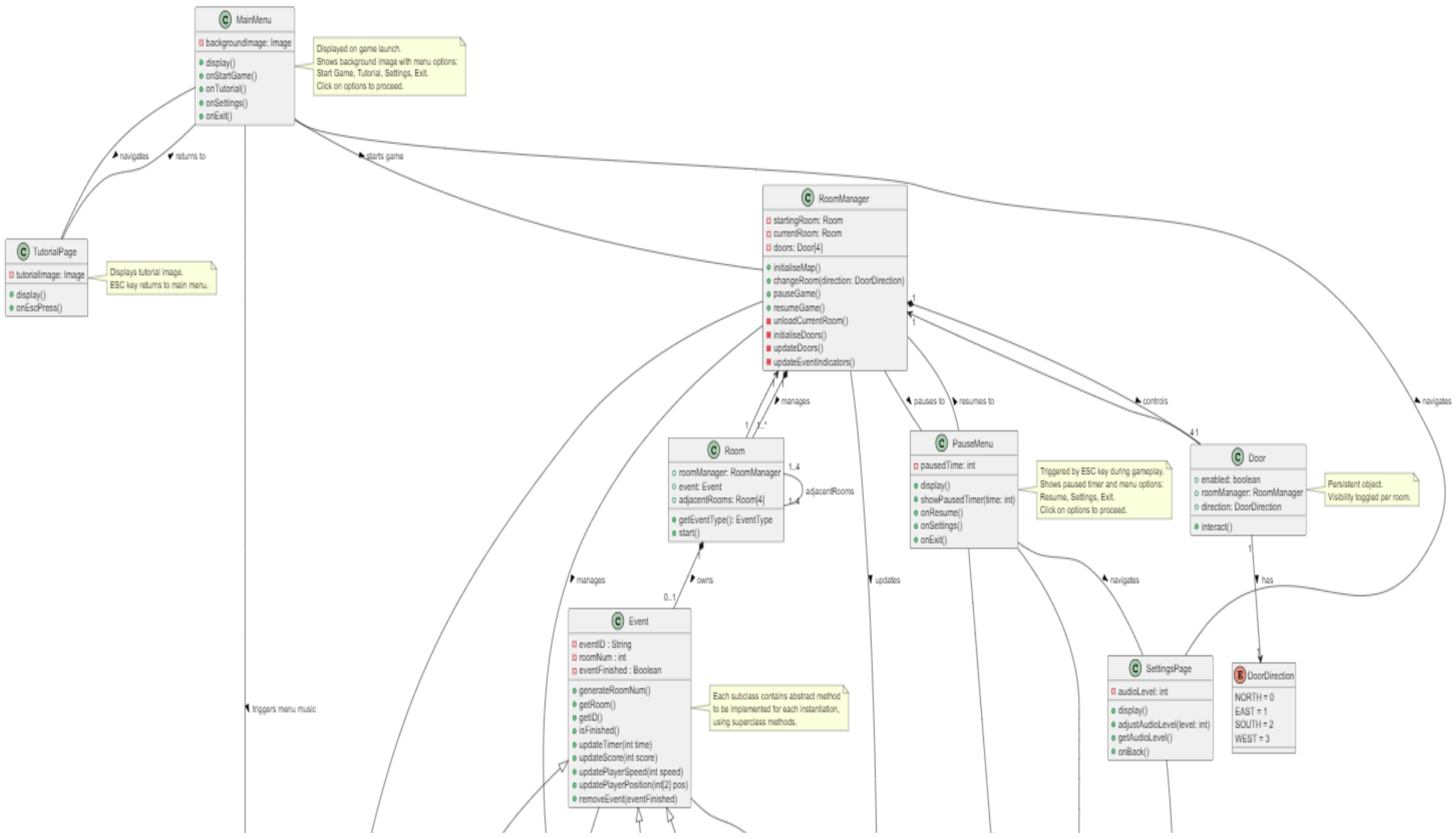
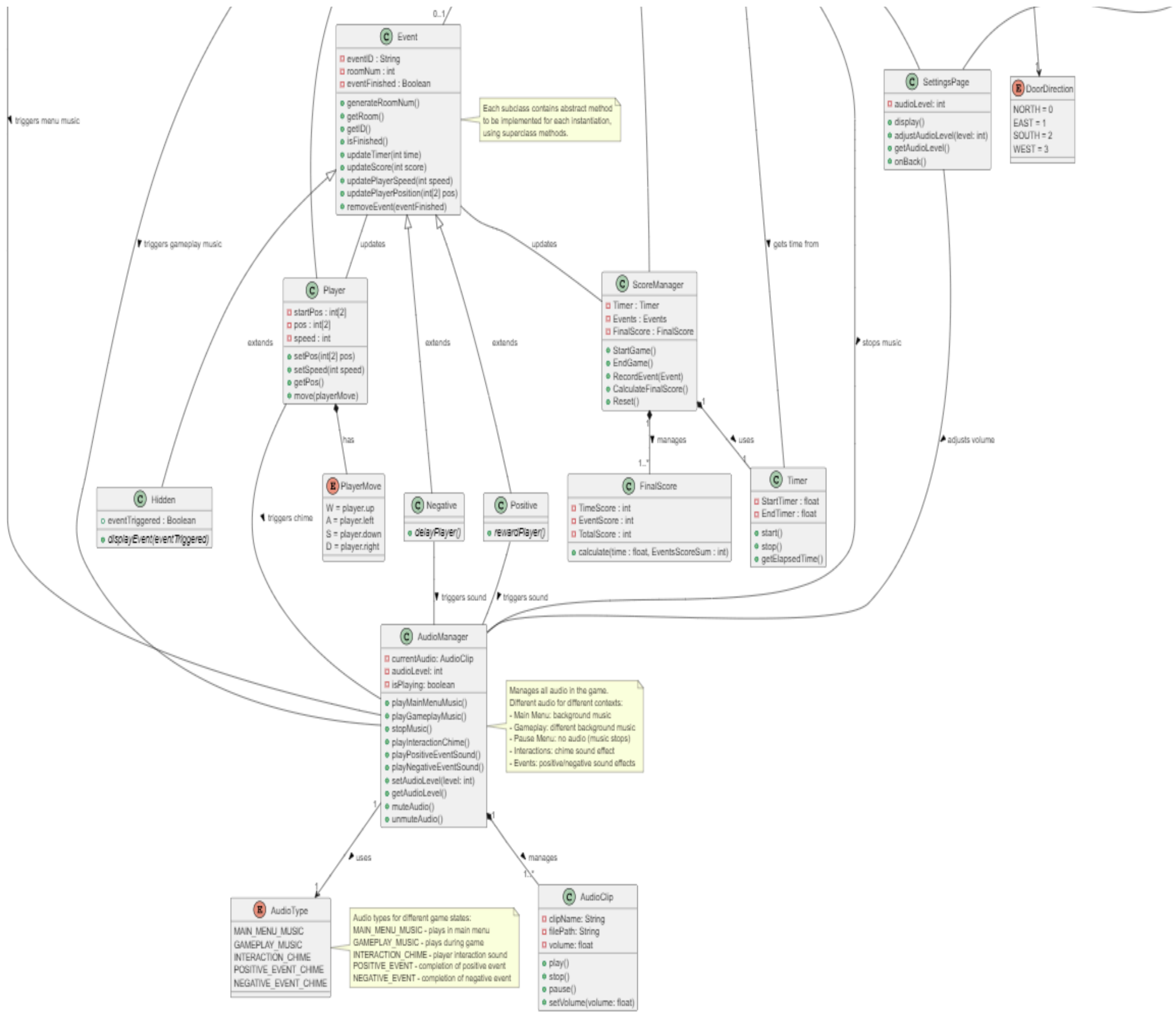
Architecture

During the architectural design process of our game we have used various UML diagrams (class, sequence, state and component) in order to give the most concise and accurate portrayal of the games general design. We have used plantUML in order to generate these diagrams using various IDE's and websites. To begin our design process we began by developing CRC cards, this gave us an opportunity to brainstorm the key components of the game and how it would all fit together, these cards can be found on our team website. This made designing UML diagrams much easier as we had a strong idea of the core structure of the game.

Class Diagram:
We began with class diagrams as these would give a great description of how all the components of the game would interact with each other and their individual attributes and features. The process began with different team members designing various separate class diagrams that encapsulated a particular structure, once this was completed we could combine these into one complete class diagram that describes the structure in a high level of detail. (Split into two cropped images to increase visibility, as it is a large diagram).

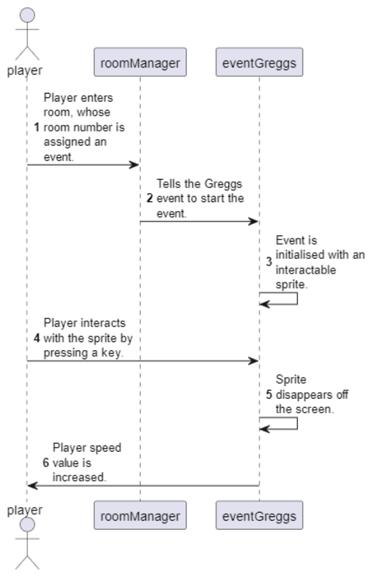




Sequence diagrams:

We have used sequence diagrams in order to describe various interactions between the player and the game, these diagrams were incredibly useful to allow us to show how requirements from the previous section would be met. The first three diagrams describe an interaction between the player and each various event, and how the game reacts to this interaction. The fourth diagram is a portrayal of how the different components work together once the player has successfully escaped. Finally we have a sequence diagram to describe an interaction where the player first pauses the game and then goes on to adjust the volume of the game.

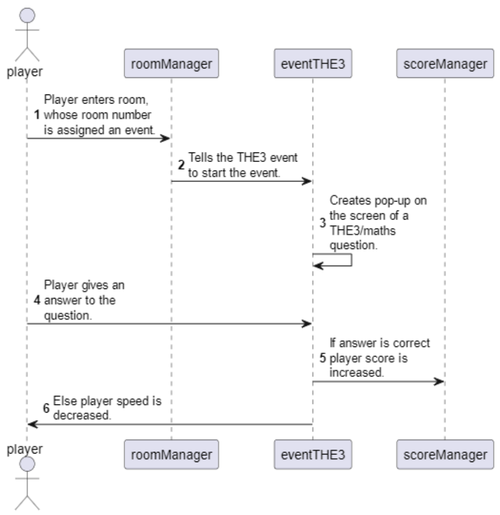
Positive Event Sequence Diagram:


player
roomManager
eventGreggs
1 Player enters room, whose room number is assigned an event.
2 Tells the Greggs event to start the event.
3 Event is initialised with an interactable sprite.
4 Player interacts with the sprite by pressing a key.
5 Sprite disappears off the screen.
6 Player speed value is increased.
player
roomManager
eventGreggs

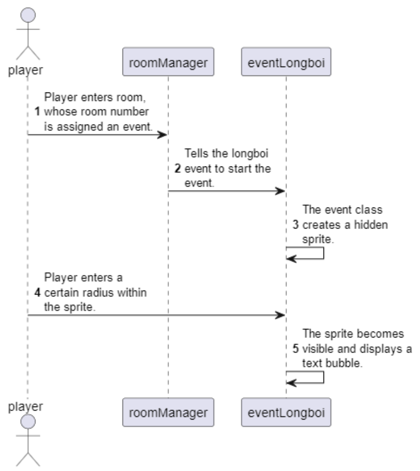
- UR_Positive - describes an implementation of a positive event subclass.
- UR_Fast - highlights a way that the player can escape quicker and obtain a better score.
- UR_University - “Greggs” is a typical student location for food, playing into what students may relate to.

Negative Event Sequence Diagram:

- UR_Negative - describes an implementation of a negative event subclass.
- UR_University - “THE3” is an aspect of University life that has strong links to University life.

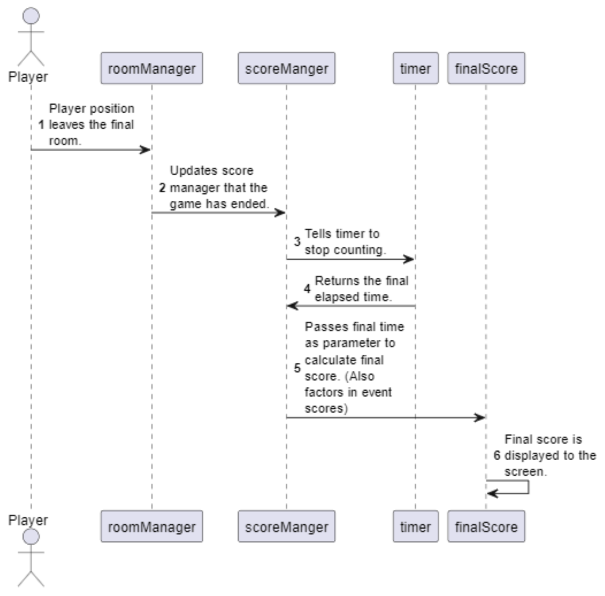

player
roomManager
eventTHE3
scoreManager
1 Player enters room, whose room number is assigned an event.
2 Tells the THE3 event to start the event.
3 Creates pop-up on the screen of a THE3/maths question.
4 Player gives an answer to the question.
5 If answer is correct player score is increased.
6 Else player speed is decreased.
player
roomManager
eventTHE3
scoreManager

Hidden Event Sequence Diagram:


player
roomManager
eventLongboi
1 Player enters room, whose room number is assigned an event.
2 Tells the longboi event to start the event.
3 The event class creates a hidden sprite.
4 Player enters a certain radius within the sprite.
5 The sprite becomes visible and displays a text bubble.
player
roomManager
eventLongboi

- UR_Hidden - describes an implementation of a hidden event subclass.
- UR_University - “longboi” is again another aspect of York University that students will enjoy, reinforcing the style of game we have chosen.

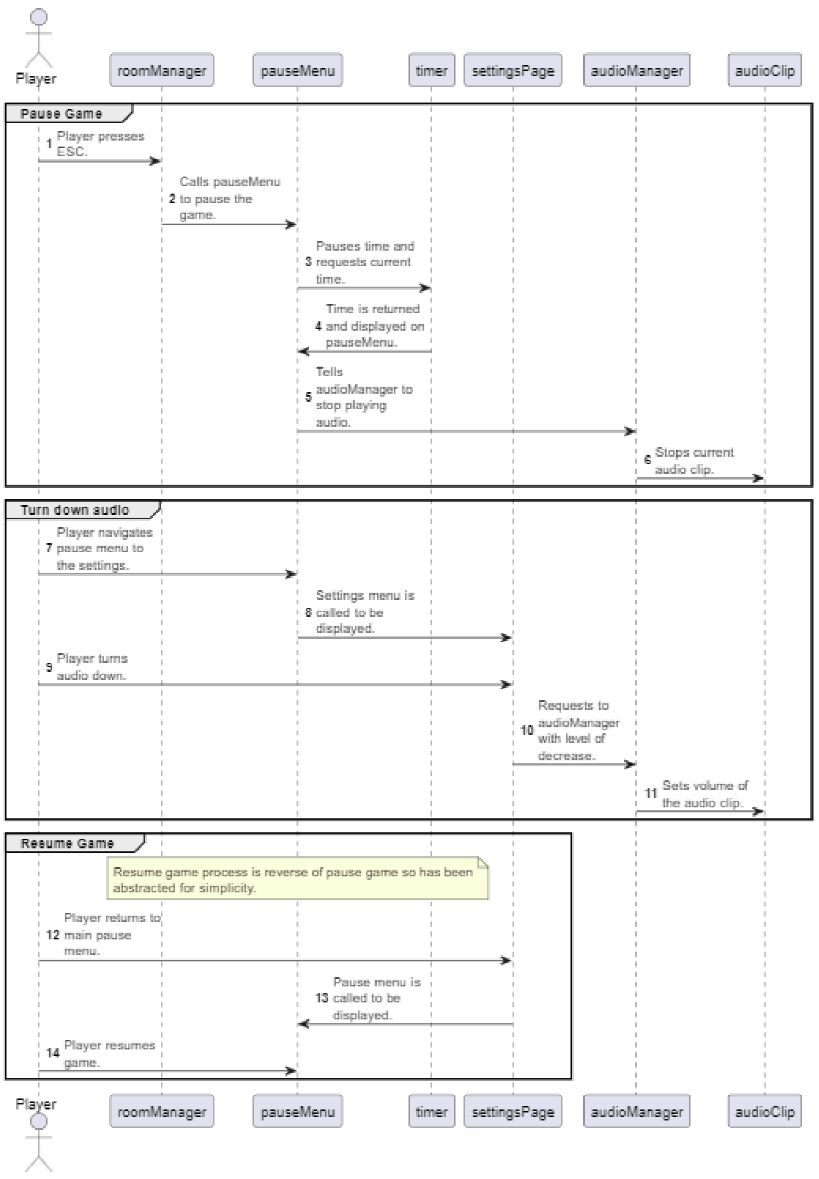
Game Ends Sequence Diagram:


Player
roomManager
scoreManger
timer
finalScore
1 Player position leaves the final room.
2 Updates score manager that the game has ended.
3 Tells timer to stop counting.
4 Returns the final elapsed time.
5 Passes final time as parameter to calculate final score. (Also factors in event scores)
6 Final score is displayed to the screen.
Player
roomManager
scoreManger
timer
finalScore

- UR_Lose_Time - highlights the outcome of what happens if the player escapes before the timer has reached 0.
- UR_Score - describes the final score being shown to the player once the game has ended.

Pause Menu Sequence Diagram:

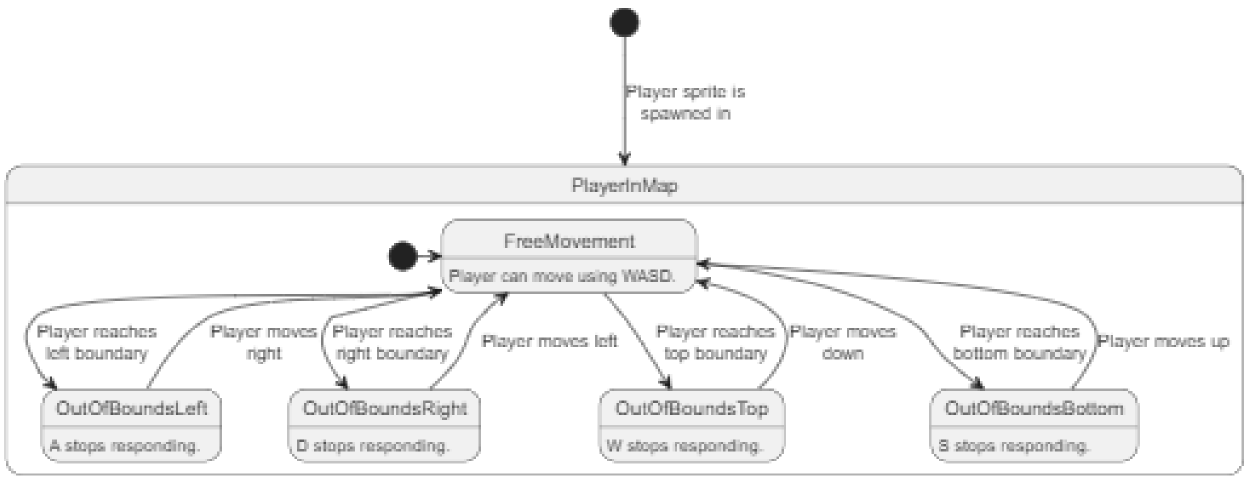
- UR_Pause - provides a description of how the player can pause the game, and the logic that follows (pausing timer and audio playing).
- UR_Settings - shows how the player can navigate the pause menu to reach the settings and the audio options that follow.
- UR_Audio - describes how the audio is adjusted by the player from the settings page and what needs to be updated.
- UR_Time_Track - gives an example of how the current time can be requested and describes the time remaining on the pause screen.



Player
roomManager
pauseMenu
timer
settingsPage
audioManager
audioClip
Pause Game
1 Player presses ESC.
2 Calls pauseMenu to pause the game.
3 Pauses time and requests current time.
4 Time is returned and displayed on pauseMenu.
5 Tells audioManager to stop playing audio.
6 Stops current audio clip.
Turn down audio
7 Player navigates pause menu to the settings.
8 Settings menu is called to be displayed.
9 Player turns audio down.
10 Requests to audioManager with level of decrease.
11 Sets volume of the audio clip.
Resume Game
Resume game process is reverse of pause game so has been abstracted for simplicity.
12 Player returns to main pause menu.
13 Pause menu is called to be displayed.
14 Player resumes game.

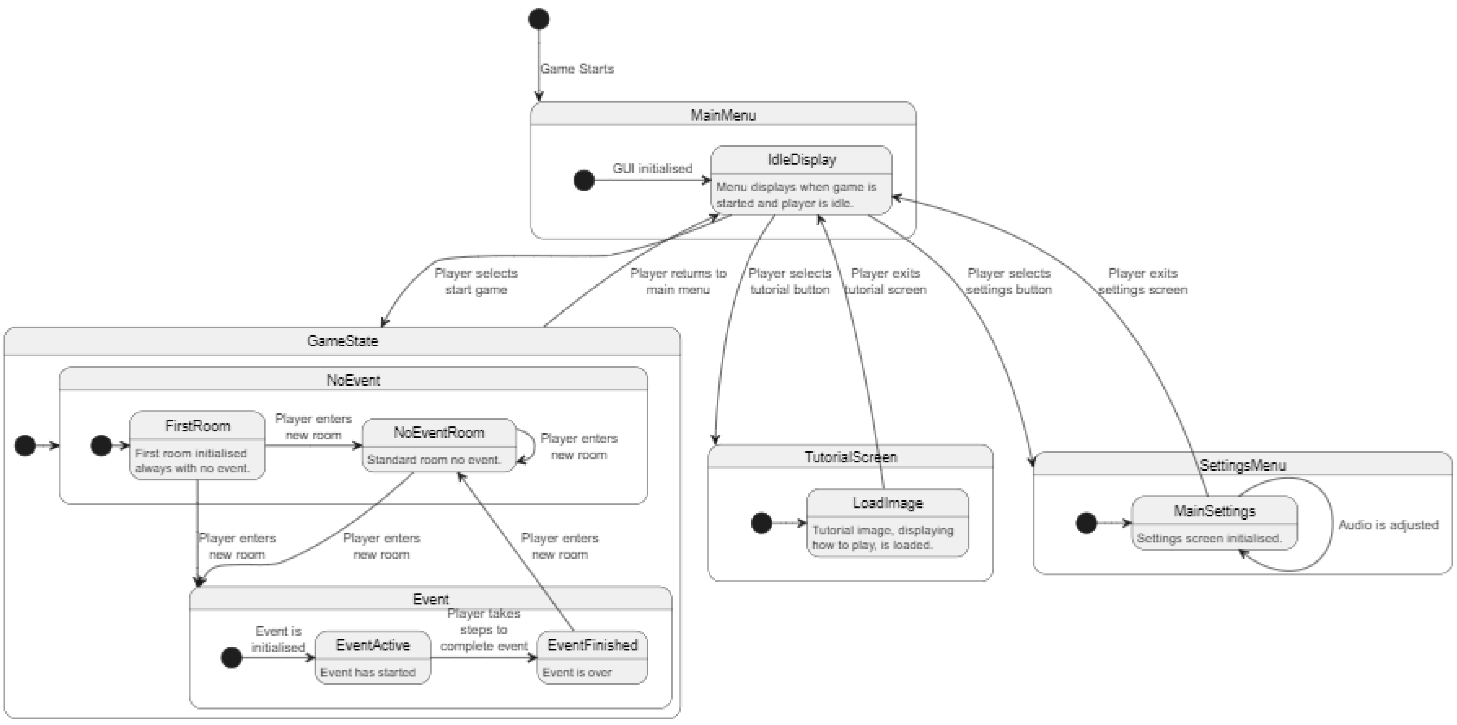
State Diagrams:
Our state diagrams have been key in portraying the different states displayed when the game is first loaded and what happens when the player interacts with this system. We have also used it to highlight how the boundaries of the map are enforced and how this links to the controller binds. In the final diagram we show a simple diagram to describe how the game checks to see if the timer has run out and to end the game once this has happened.

Map Boundary State Diagram:



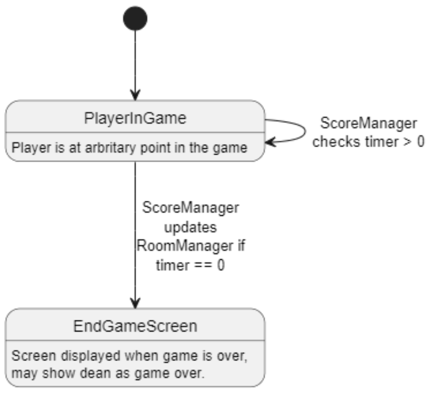
- UR_Player - describes without detail a player being spawned into the map as a sprite that has the ability to move around the map.
- UR_Map_Limits - clear boundaries are shown and each one is represented as a different state within the overarching state. When a player tries to exit a boundary the corresponding directional input is temporarily disabled.
- UR_Desktop_Inputs - shows the mapping between the input keys and the movement it elicits by the respective boundaries.

Game Start State Diagram:


Game Starts
MainMenu
GUI initialised
IdleDisplay
Menu displays when game is started and player is idle.
Player selects start game
Player returns to main menu
Player selects tutorial button
Player exits tutorial screen
Player selects settings button
Player exits settings screen
GameState
NoEvent
FirstRoom
First room initialised always with no event.
Player enters new room
NoEventRoom
Standard room no event.
Player enters new room
Player enters new room
Player enters new room
Player enters new room
Event
Event is initialised
EventActive
Event has started
Player takes steps to complete event
EventFinished
Event is over
TutorialScreen
LoadImage
Tutorial image, displaying how to play, is loaded.
SettingsMenu
MainSettings
Settings screen initialised.
Audio is adjusted

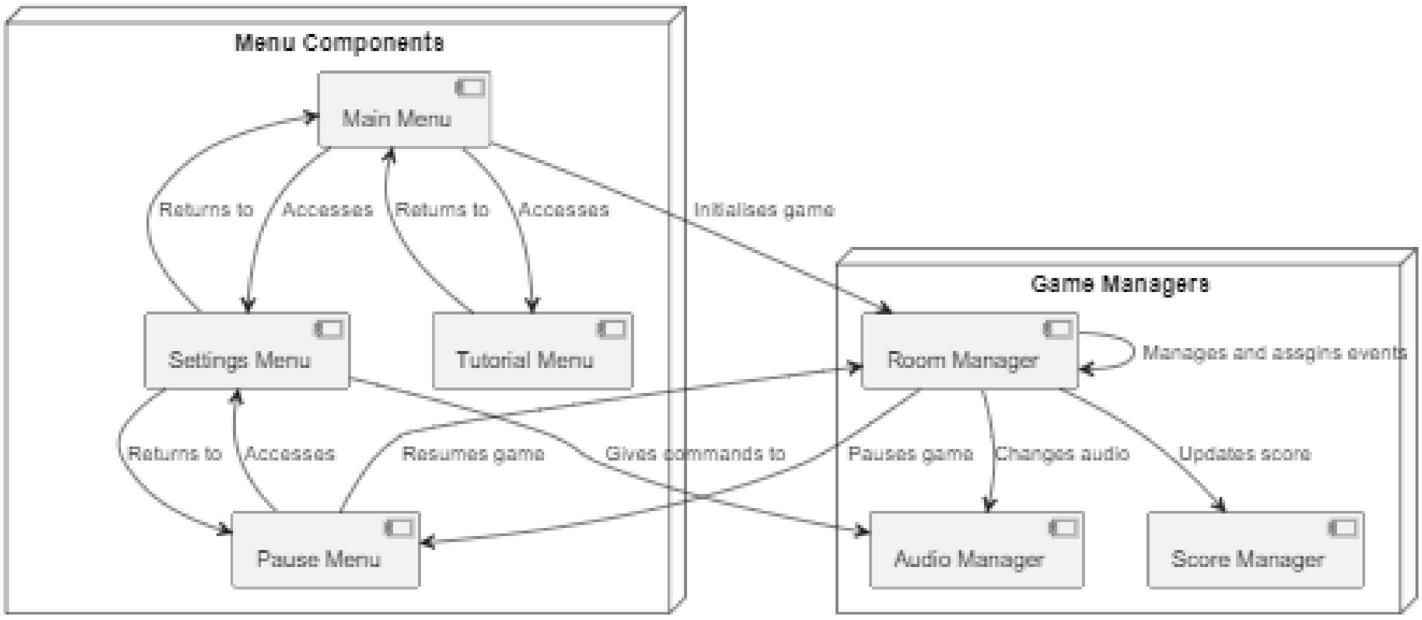
- UR_Main_Menu - shows how once the game is loaded up, the main menu is immediately displayed, displaying the ability to change to various other game states.
- UR_Tutorial - this diagram shows how the state of the game changed when the player selects the tutorial from the main screen, displaying the image.
- UR_Settings - highlights how the settings menu is accessed from the main menu and gives a brief portrayal of how audio can be adjusted.
- UR_Positive/Negative/Hidden - gives a general description of how the logic works for each event type within the game state (partitioned into having an event and not having an event).

Player Loses State Diagram:


PlayerInGame
Player is at arbitrary point in the game
ScoreManager checks timer > 0
ScoreManager updates RoomManager if timer == 0
EndGameScreen
Screen displayed when game is over, may show dean as game over.

- UR_Lose_Time - this diagram primarily describes how the game checks to see if the timer has reached 0 and what happens if this does occur.
- UR_Time_Track - briefly shows how the game is always checking and “listening” to see what the value of timer is.

Component Diagram:
We created a component diagram that describes the logic and access level, each component of the game uses to interact with one another, split up into menu components and game managers. This helped to create a more abstracted version of our class diagram, with the aim of describing the logic between menus and controllers with less focus on detailed mechanics.



- UR_Main_Menu - highlights which parts of the game the main menu can access.
- UR_Pause - like earlier shows how the tutorial is accessed via the main menu.
- UR_Audio - describes how audio is accessed both via the game menus and how the room manager controls the audio playing.
- UR_Settings - the settings can be accessed via either the main menu upon booting up the game or via navigating the pause menu, during gameplay.