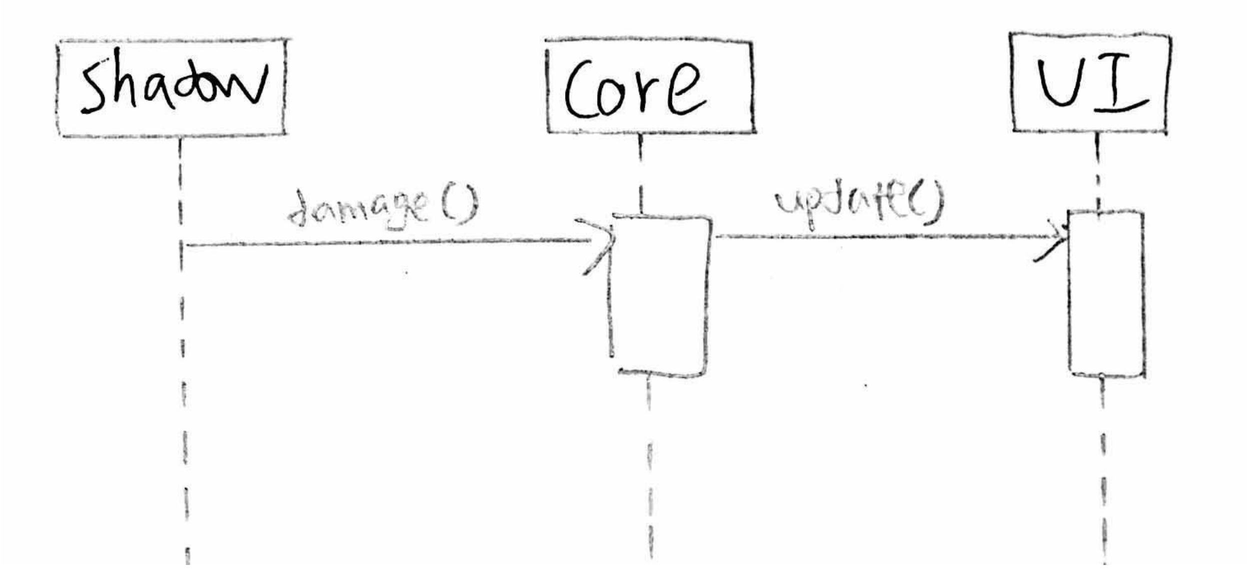
Homework Question 1: Use cases & Sequence diagrams

1) Use Case: Shadow damage

Name	**Shadow damage**
Objective	To update the health bar of the core accordingly when a shadow touches the core
Pre-conditions	1. A game must already be in session 2. There must still be sufficient health left for the core
Post-conditions	Success: 1. The health of the core is reduced 2. The shadow is removed from the game Failure: 1. The light orb does not disappear 2. The user is not carrying anything
Actors	1. Shadows (enemies) 2. Core 3. User interface
Trigger	When a shadow touches the core
Normal Flow	1. A shadow moves towards the core 2. The shadow touches the core 3. The health of the core is reduced
Alternative Flow	1. If the core's health is completely depleted, the game session ends and the shadow is removed
Interacts with	1. Shadow class 2. Core class 3. User Interface class
Open Issues	1. The disappearing of the shadow and updating of the core's health should be done in 0.5 seconds 2. During a game session, disappearing of shadow should only be performed when the shadow is touching core

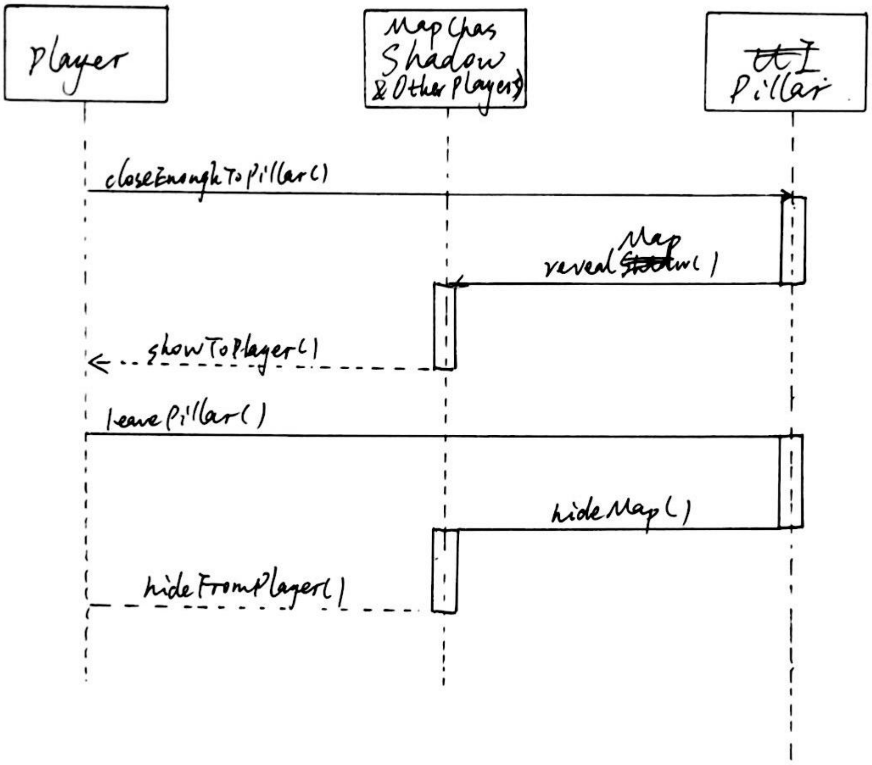
1) Sequence diagram: Shadow damage



2) Use Case: Gain full sight of the map

Name	**Gain full sight of the map**
Objective	To get all current movement information of shadow and other players on the map, so that players can plan their next movement to defend the core.
Pre-conditions	1. The player must be connected to the game room. 2. The player must stand next to one of the pillars closely enough. 3. All players must have stable internet connection.
Post-conditions	Success: 1. The map is no longer covered with darkness but every movement of shadow and other players becomes visible. Failure: 1. The player cannot see the fully visible map. 2. Some other players' information has not been successfully updated, so the map information is out-of-date.
Actors	1. Player (User) 2. Shadow (Enemies) 3. Pillar 4. Server
Trigger	To protect the core, player must observe the movement shadow to make defensive plan.
Normal Flow	1. The player moves next to a pillar. 2. The map visibility attribute is updated to "visible". 3. The user interface displayed on the phone is now providing full sight to this player.
Alternative Flow	1. Graphics rendering is not working properly and the player cannot see the full map. 2. Bad internet connection exists among the players so some players are shown not moving on the map.
Interacts with	1. Player class 2. User Interface class 3. Pillar class 4. Shadow class
Open Issues	1. How quickly the game engine can detect whether the player stands closely enough near the pillar. 2. How quickly the graphics rendering can respond to visibility changes.

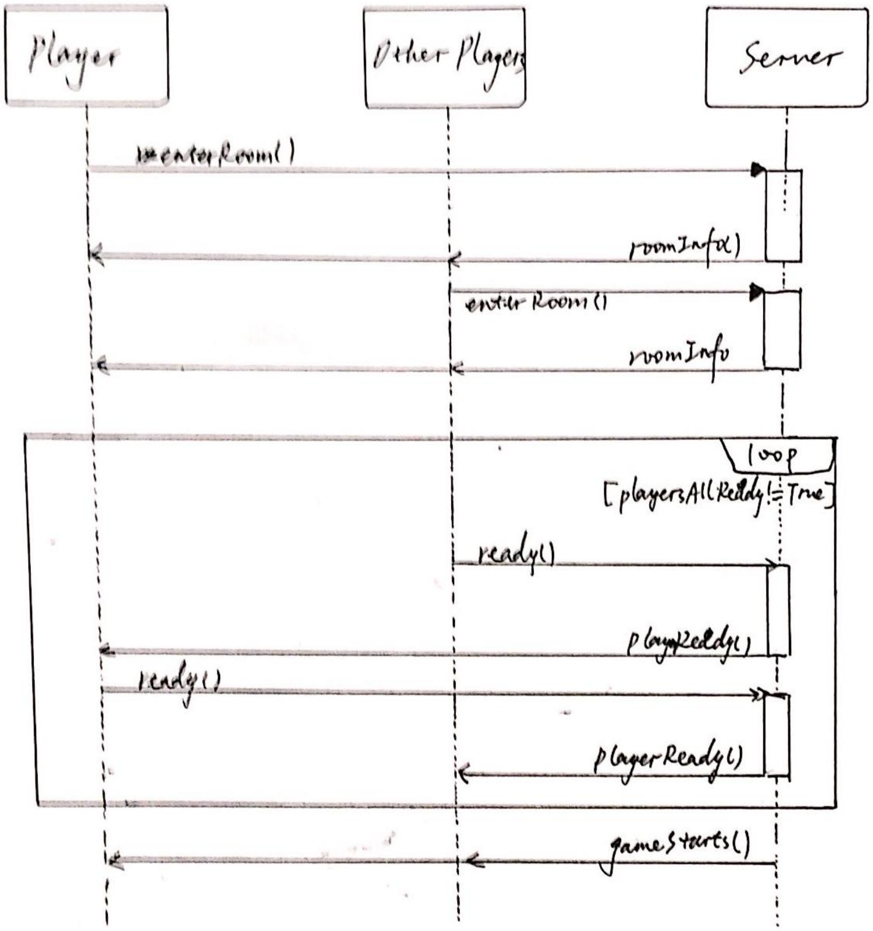
2) Sequence diagram: Gain full sight of the map



3) Use Case: Game starts

Name	**Game starts**
Objective	To initiate the cooperative task of purifying children’s mind for all connected players in the same game room.
Pre-conditions	1. All players must be successfully connected to the game room. 2. All players in the game room must press “ready” button. 3. All players must not have bad internet connection.
Post-conditions	Success: 1. Game is initialized that elements including map, shadow generator, pillars, orbs of light and player characters are created. Failure: 1. Game session is aborted. 2. Game fails to start.
Actors	1. Players (User) 2. Server
Trigger	Players want to play the game together with other players.
Normal Flow	1. Players are matched into the same room. 2. Players are all in “ready” mode. 3. Game world is initialized and synchronized to all players’ device.
Alternative Flow	1. A player loses connection and the game session is aborted. 2. Game world cannot be synchronized to all devices and the game fails to start.
Interacts with	1. Player class 2. User Interface class
Open Issues	1. How to quickly detect player’s network condition. 2. How quickly the graphics rendering can be initialized.

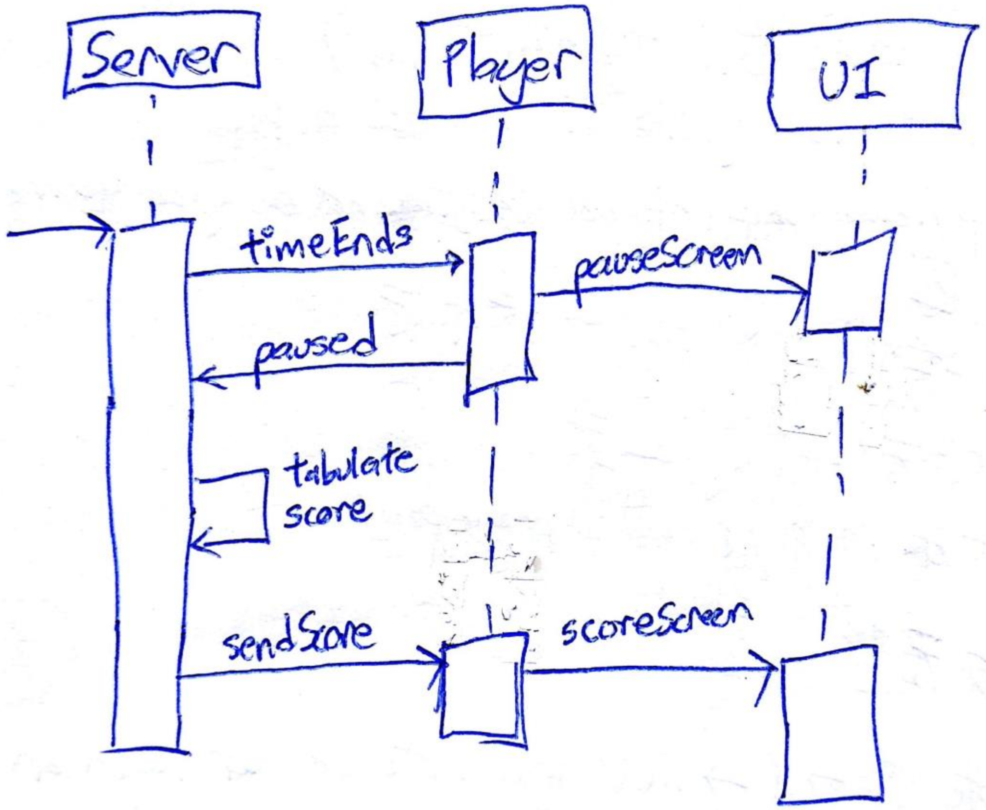
3) Sequence diagram: Game starts



4) Use Case: wave defending successful

Name	**Wave defending successful**
Objective	The main objective of defending the core has been met before the game time is up
Pre-conditions	1. The user must be connected to the game room 2. The health of the core is not used up 3. The game time has ended
Post-conditions	Success: 1. Players are all freezed and can no longer move around in the screen 2. The game ends and shows a pop-up screen showing that the players have won the game Failure: 1. The game continues to proceed even though time has ended 2. Shadows continue to appear to hit the core
Actors	1. Player (User) 2. Other players 3. Server
Trigger	The game time has ended and the health of the core has not been used up
Normal Flow	1. The game time ends 2. Server checks that the core health is not used up 3. Show all players the “they have won” screen
Alternative Flow	4. The game continues to proceed even though time has ended 5. Shadows continue to appear to hit the core
Interacts with	1. Player class 2. User Interface class
Open Issues	1. Checking whether a shadow hits a core right before the game time ends and causing the core health to be used up 2. How to ensure that all game status are always synchronized (player position, orb placement etc) with the server so that objective success/failure is correctly computed

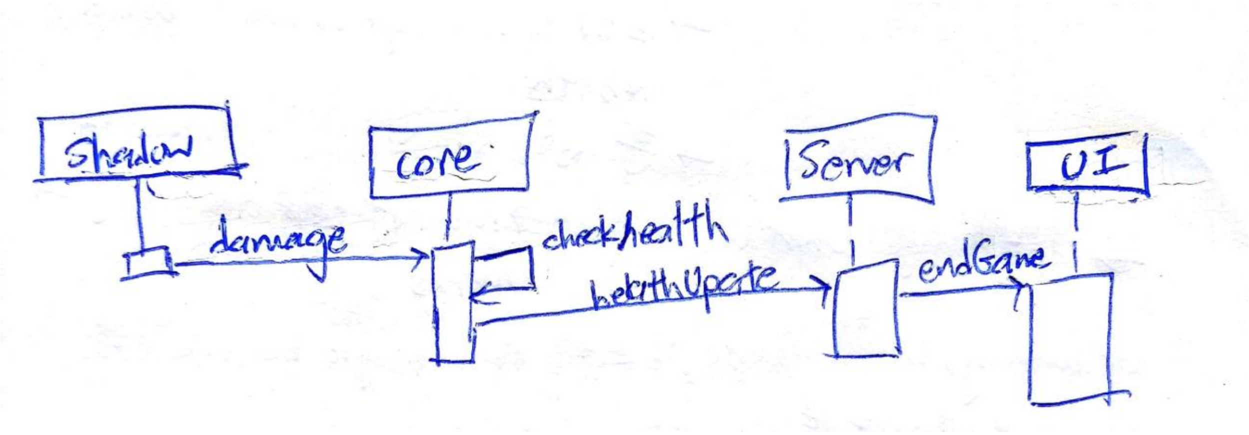
4) Sequence diagram: **wave defending successful**



5) Use Case: Objective lost

Name	**Main objective of defending the core is not met**
Objective	To end the game when the objective of defending the core is not met. i.e the health of the core is used up
Pre-conditions	1. The user must be connected to the game room 2. The health of the core is used up 3. The game time has not ended
Post-conditions	Success: 1. Players are all freezed and can no longer move around in the screen 2. The game ends and shows a pop-up screen showing that the players have lost the game Failure: 1. The game continues to proceed even though time has ended 2. Shadows continue to appear to hit the core
Actors	1. Player (User) 2. Other players 3. Server
Trigger	The health of the core has been used up before game time ends
Normal Flow	1. The game time ends 2. Server checks that the core health is not used up 3. Show all players the “they have won” screen
Alternative Flow	1. The game continues to proceed even though health of core is used up 2. Shadows continue to appear to hit the core
Interacts with	1. Player class 2. User Interface class
Open Issues	1. The updating of core’s health across all players should be instantaneous 2. All the decrease of core’s health should be recorded and tallied in real-time as the game proceeds

5) Sequence diagram: **Objective lost**


Shadow
core
Server
UI
damage
checkhealth
healthUpdate
endGame