3. Explain how our OOD will work - Pick two interesting (non-trivial) actions/steps that the game must support and explain how your OOD supports this. - For example, explain how the board will be drawn, or how the user's move is handled. - Imagine that you are presenting your design to your team and persuading them it's the best design. Your explanation here could be part of how you'd show that. -

1. One of the first non-trivial action steps that the game must implement is a maze generation algorithm, to make sure every cheese position is reachable and that walls are placed strategically. My OOD supports this by having a separate class to store the maze and another class to actually generate the maze. Therefore, the class which stores the maze can remain high level and the generation class uses more theory and a more difficult class. The UI will then interact with the simpler maze class. The maze generation class can use a cell[][] array such that each cell contains methods to setWalls(), isCat(), isMouse(), and isVisited() ETC.

2. The second non-trivial action within this game could be the movement of the cats throughout the maze, such that a cat only backtracks to previous positions if no new positions are found. My OOD supports this by keeping track of current and past positions in a Cat object. Doing this will allow for easier checks and backtracks whenever needed in another class such as the maze class. The maze class then uses the previous and current positions to find a new cell on the board reachable by this cat.