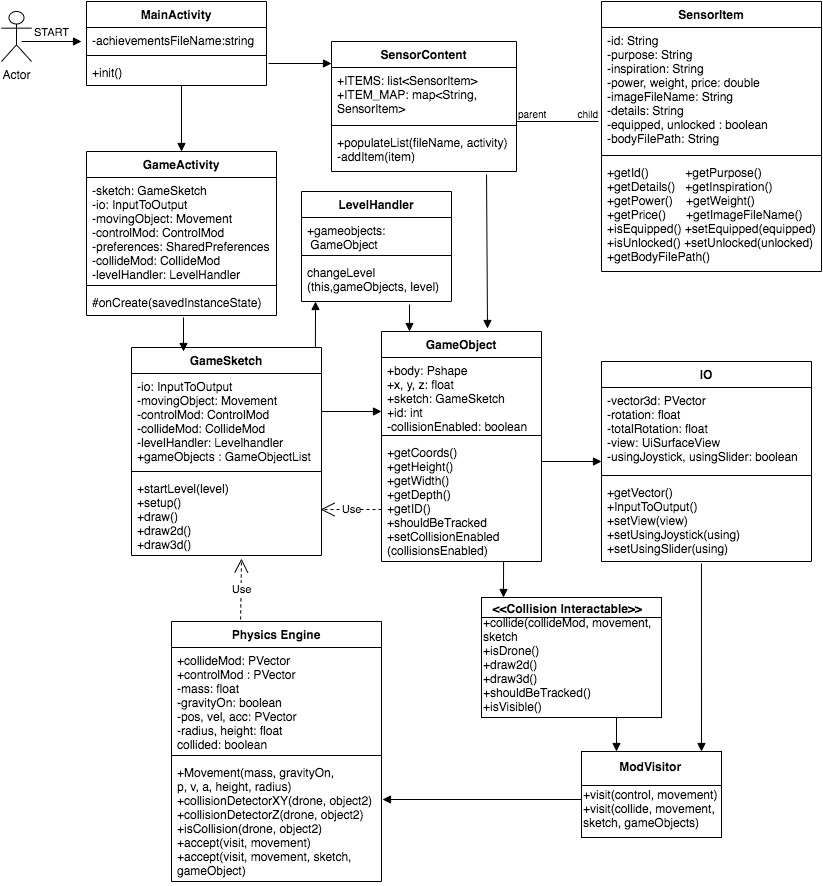


**Figure 1: UML sequence diagram of our system**



**Figure 2: UML class diagram of our system**

We chose to create a static UML diagram and a dynamic UML diagram which shows the fundamental aspects of our system, notably customization, collision detection and user control. We chose this representation for both diagrams because it gives a global overview of what happens when the game starts and because it shows the design choices we implemented in our system.

Both diagrams convey the main aspects of our system. At the start, when the game is first loaded, the system is in *MainActivity*. Once the player once has chosen all the customization options he wants on his drone and presses the Play button, the game moves on to *GameActivity*, which starts a new game sketch. In *GameSketch*, the CSV Parsers in *LevelHandler* will be called after the level has been chosen, and it will get a list of all the different in game objects that comes from the *GameObject* function, and then the different sensors will be equipped onto the drone specifically. After getting the drone from *GameObject* and having set the IO for the drone, *GameSketch* will check for collisions for the drone. Once all these steps have been achieved, the draw loop will start. Inside the draw loop, for every frame, for every in-game object, *GameSketch* will call *collideMod.accept* from *ModVisitor*, which will in turn visit the Physics engine to check any collision that is happening. Game sketch will also call *controlMod.acccept* from *ModVisitor*, who also receives the vector values from the IO at the same, and will visit the Physics Engine to update all the values related to movement such as position, velocity and acceleration. Game Sketch will also draw all game objects in 2D and 3D every frame. If the player pauses, the game will exit the draw loop and return to *GameActivity*, and if the player leaves the game from the pause menu the game will go back to *MainActivity*.

In general, these two UML diagrams convey the mentality we had throughout the project. While we had a complex task at hand, especially due to the limitations of the project, we still tried to make the code as short and easy as possible in order to implement new features or fix old features. With this project, we learned the importance of planning in advance the different modeling choices to make in a project as it allows the division of work to be more efficient compared to making the modeling choices as the project continues. Making diagrams of the project facilitates the comprehension of the project itself and makes it more structured as a whole. As a whole, this project made us understand how the development of software works in the industry.