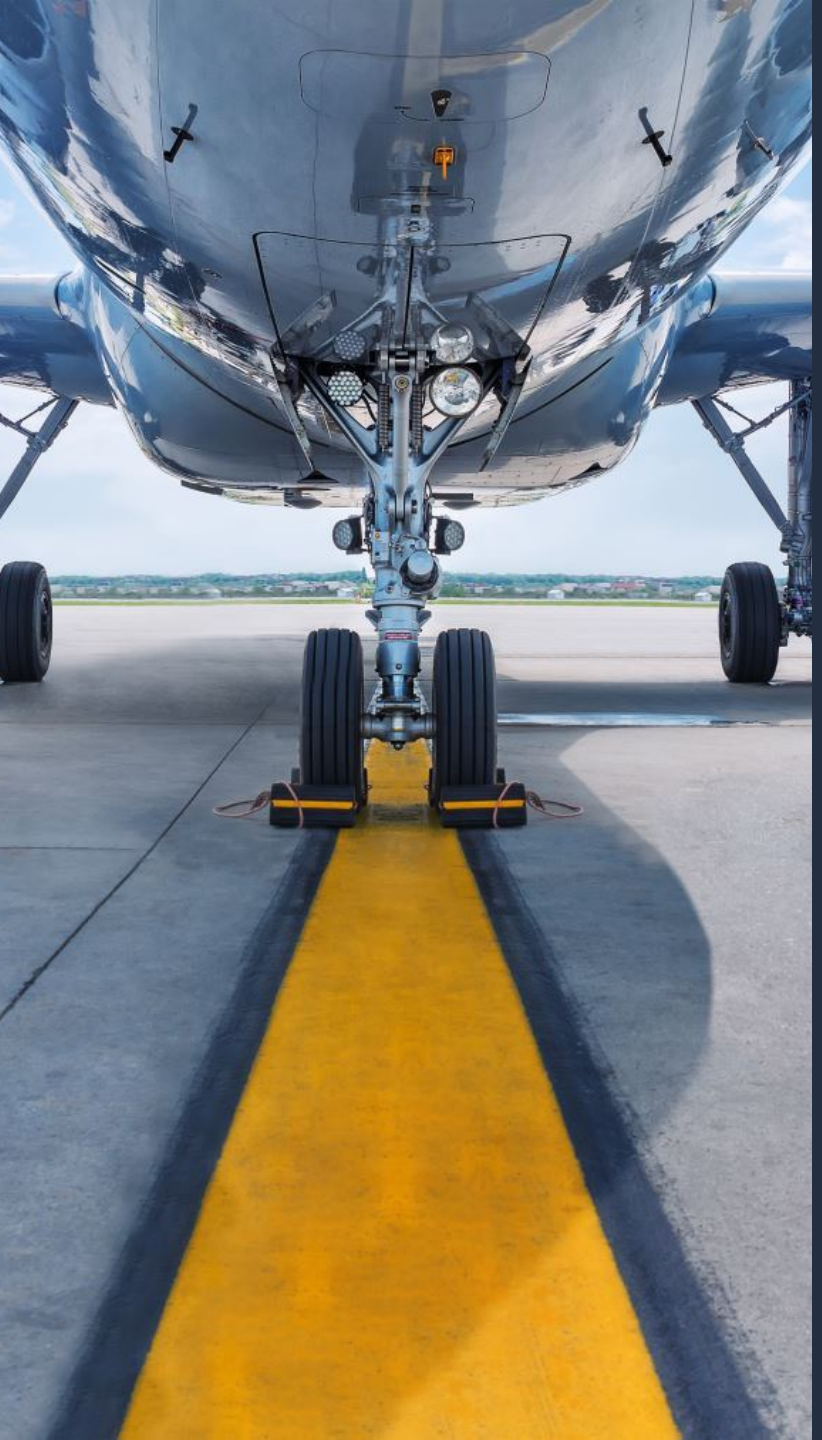
PRESENTATION

- https://www.youtube.com/watch?v=ApPPQpkoe-4



CMPE322 A1: CONCEPTUAL ARCHITECTURE

Group 20:

Group Leader: Jordan Herzstein

Christopher Seguin Bianchi, Laurie Yuzichuk, Cameron Bennett

Presenters:

Jasper Lim, Campbell Love

FLIGHTGEAR

- Open-source, multiplatform flight simulator
- Used by NASA and the FAA for standards and in training modules
- Started in 1997
- Client-Server Architecture

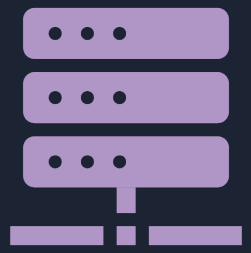
OVERVIEW

System Functionality	Concurrency
System Evolution	Division of Responsibilities
Data Flow and Control	Lessons learned
Use Cases	Conclusion

DERIVATION PROCESS

Consulted wiki, documentation, official website, general publications on flight simulators. Concluded FlightGear has an architecture that is a combination of:

Model-View-Controller Architecture,



High-Level Architecture,

and Client-Server Architecture.

SYSTEM FUNCTIONALITY

Was redesigned to better use parallelism

Now features a Model-View-Controller Architecture

Split into 2 main components: the FDM Server, and the client

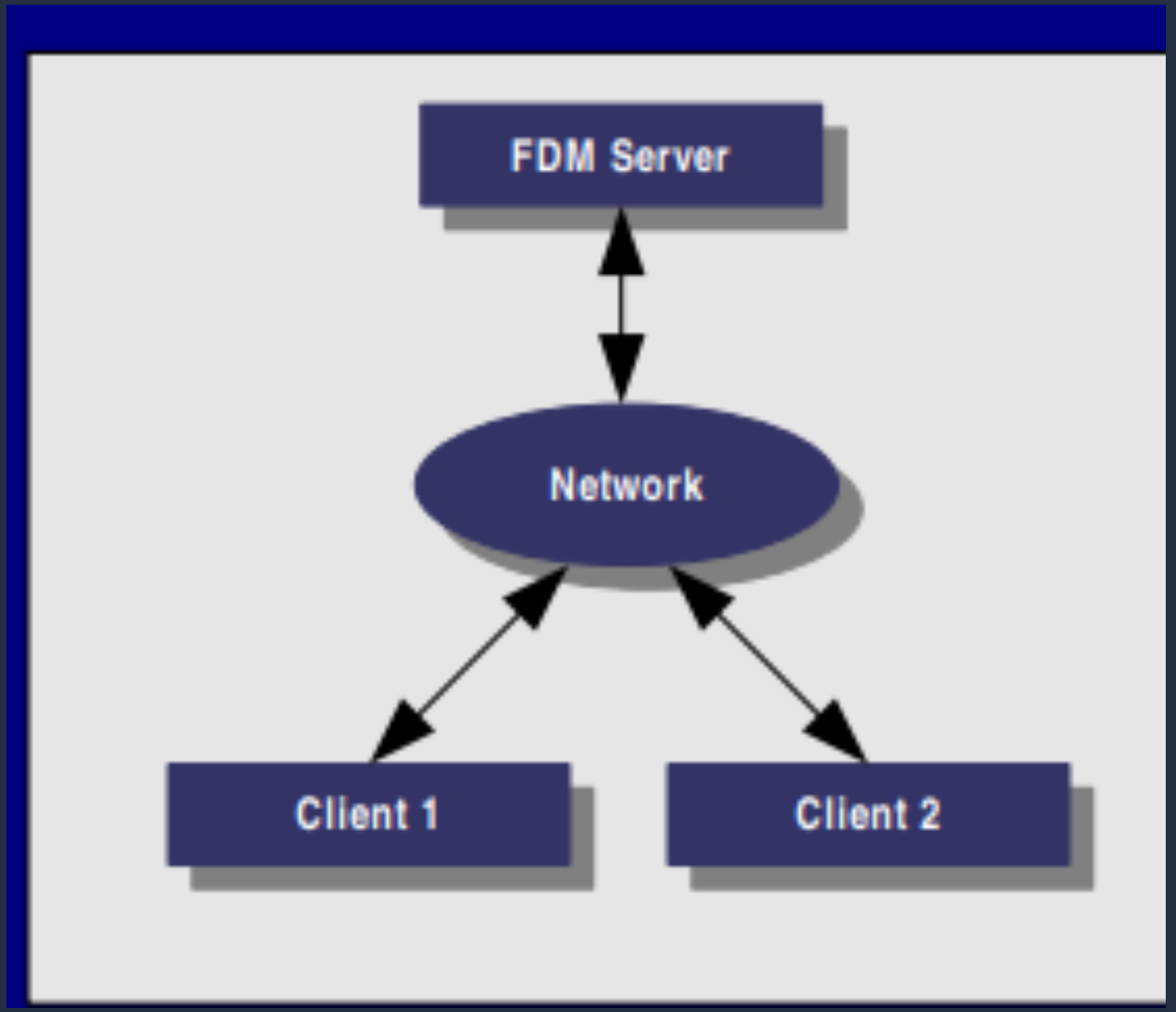


Figure 1-An example of a setup where multiple users can inhabit a single aircraft and operate its controls.

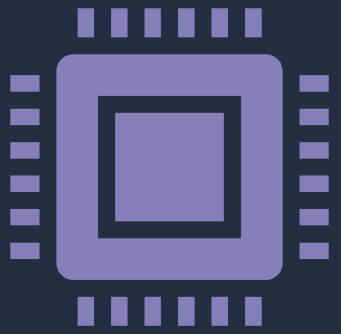
FDM SERVER

- Main calculations of the flight models and core simulation tasks, including the computation of flight dynamics and the management of environmental variables.
- Multiplayer settings as well as shared AI traffic data across sessions.

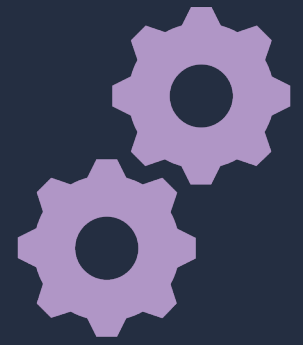
FDM CLIENT

- Interface between the user and the simulation by handling input/output operations as well as rendering the simulation's visual and audio components.
- It handles communications with the FDM server to reflect real-time changes in the simulation state, ensuring that user inputs directly influence the flight dynamics.

SYSTEM EVOLUTION



There are two forms of developers, core developers and open or ‘normal’ developers.



System architecture is built upon a modular design. Modules include aircraft models, scenery databases, user interface components, etc.

DATA FLOW AND CONTROL

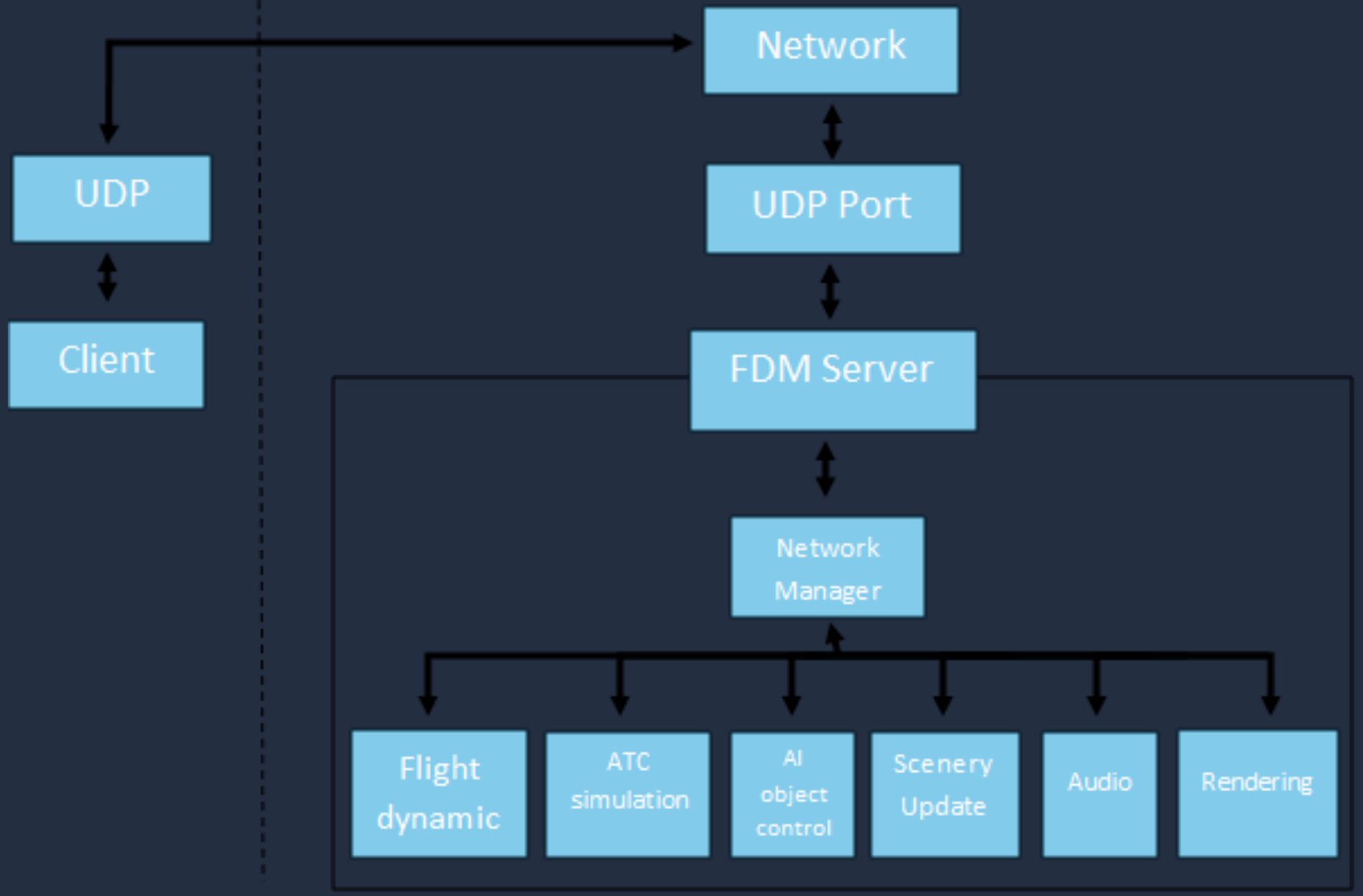


Figure 2: Diagram of FlightGear's Data Flow and Control Flow

DATA FLOW AND CONTROL

Through UDP Ports, the client/user and FDM server connect to a network. There, changes can be uploaded to and from one another and allow for either side to react accordingly.

DATA FLOW AND CONTROL

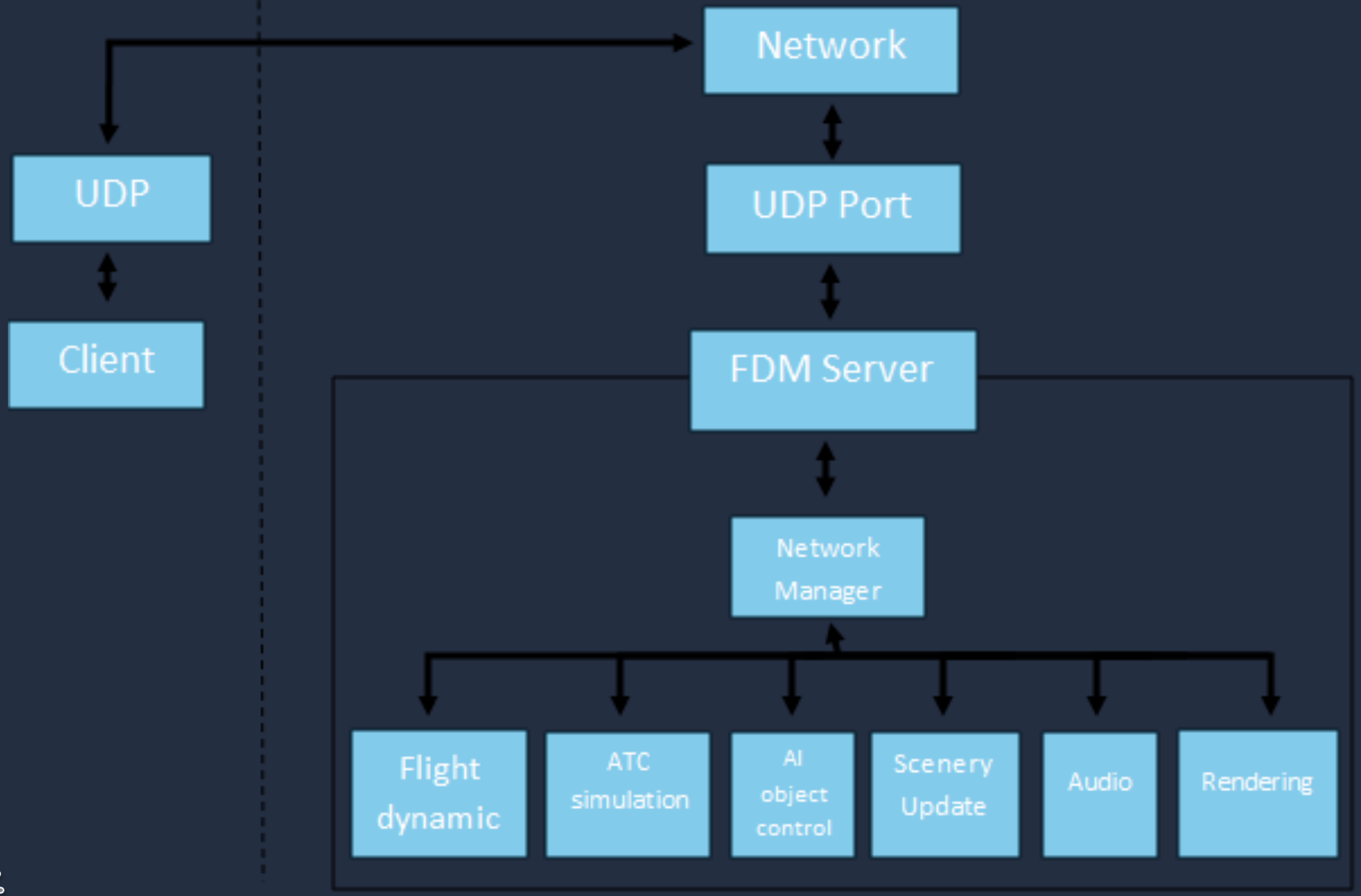
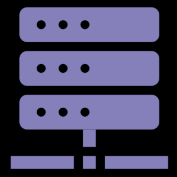
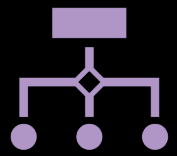


Figure 2: Diagram of FlightGear's Data Flow and Control Flow

DATA FLOW AND CONTROL



Through UDP Ports, the client/user and FDM server connect to a network. There, changes can be uploaded to and from one another and allow for either side to react accordingly.



The input the user gives using the UI, such is communicated through the network to the FDM server. Upon receiving the data, the network manager will call relevant components to make the necessary updates.

DATA FLOW AND CONTROL

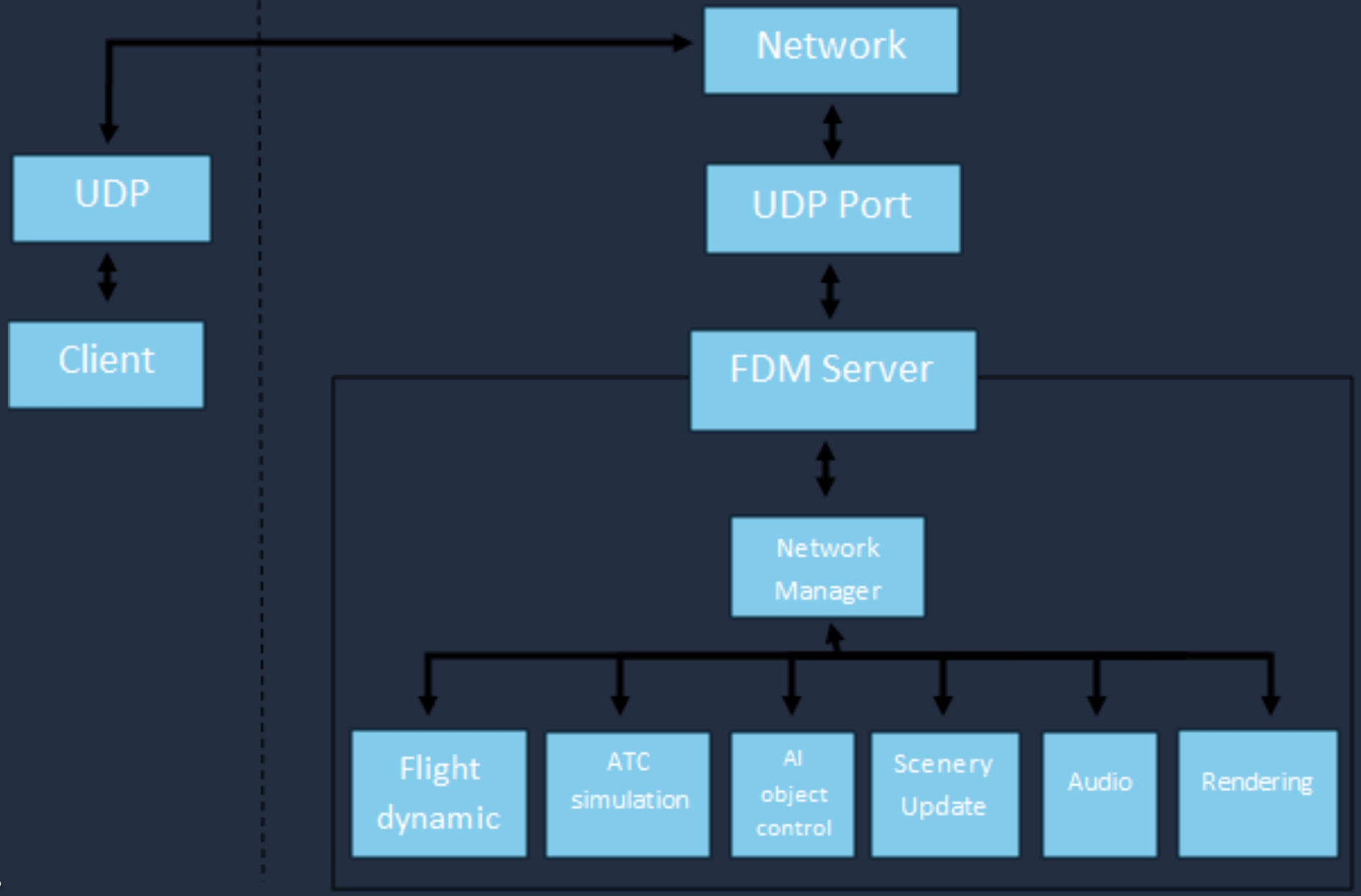
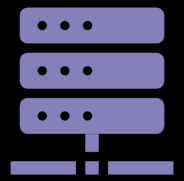
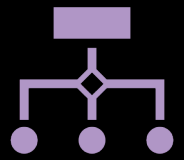


Figure 2: Diagram of FlightGear's Data Flow and Control Flow

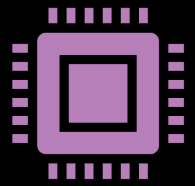
DATA FLOW AND CONTROL



Through UDP Ports, the client/user and FDM server connect to a network. There, changes can be uploaded to and from one another and allow for either side to react accordingly.



The input the user gives using the UI, such is communicated through the network to the FDM server. Upon receiving the data, the network manager will call relevant components to make the necessary updates.



Calculations will be made, and audio and visual rendering as well as updating the scenery outside the user's aircraft allow for the user to receive confirmation that their inputs have been registered by FlightGear's system. These updates will be output back to the network manager and from there, the FDM server will update the network and thus the client.

USE CASE #1

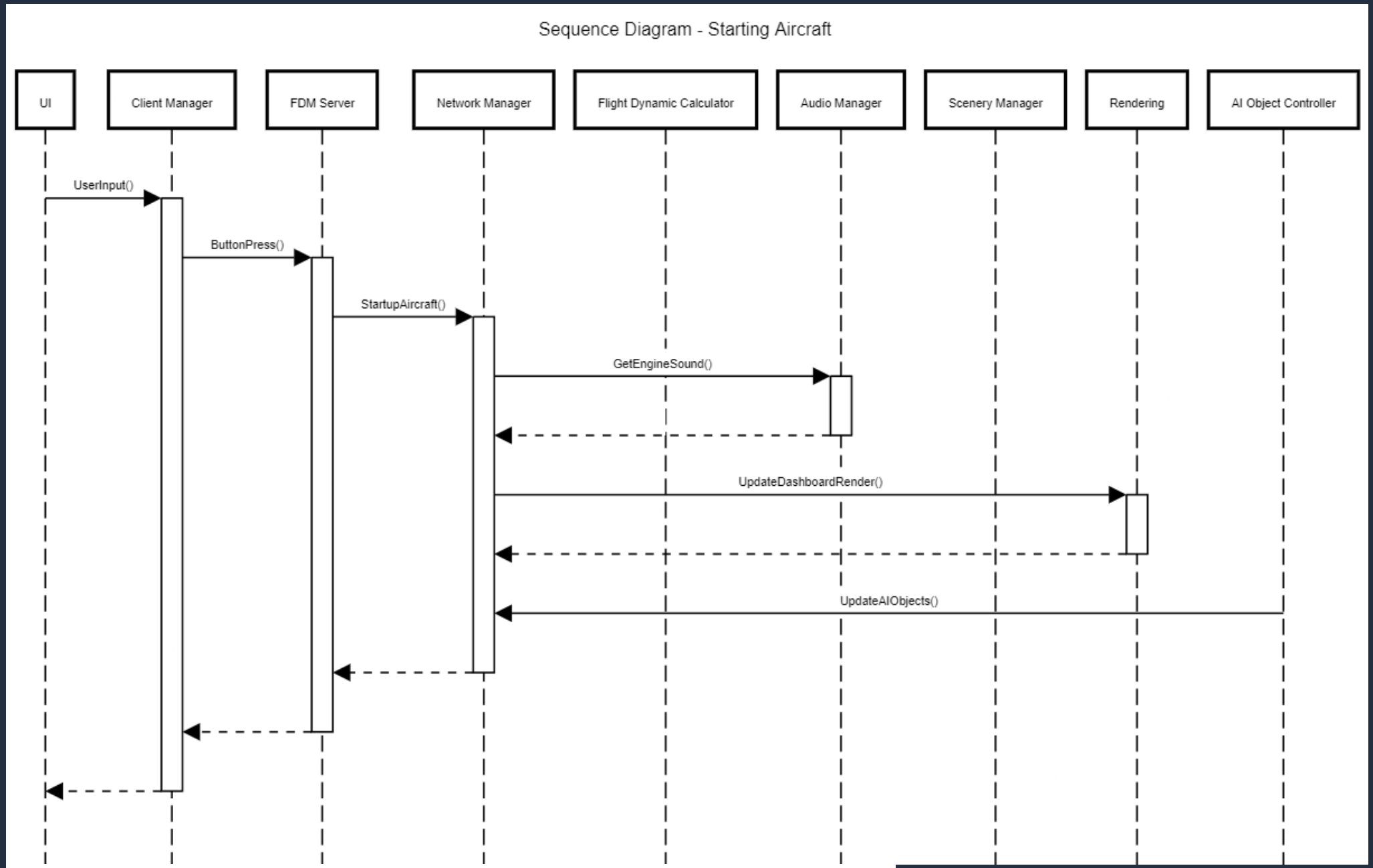


Figure 3: Sequence Diagram for Starting Aircraft

USE CASE #2

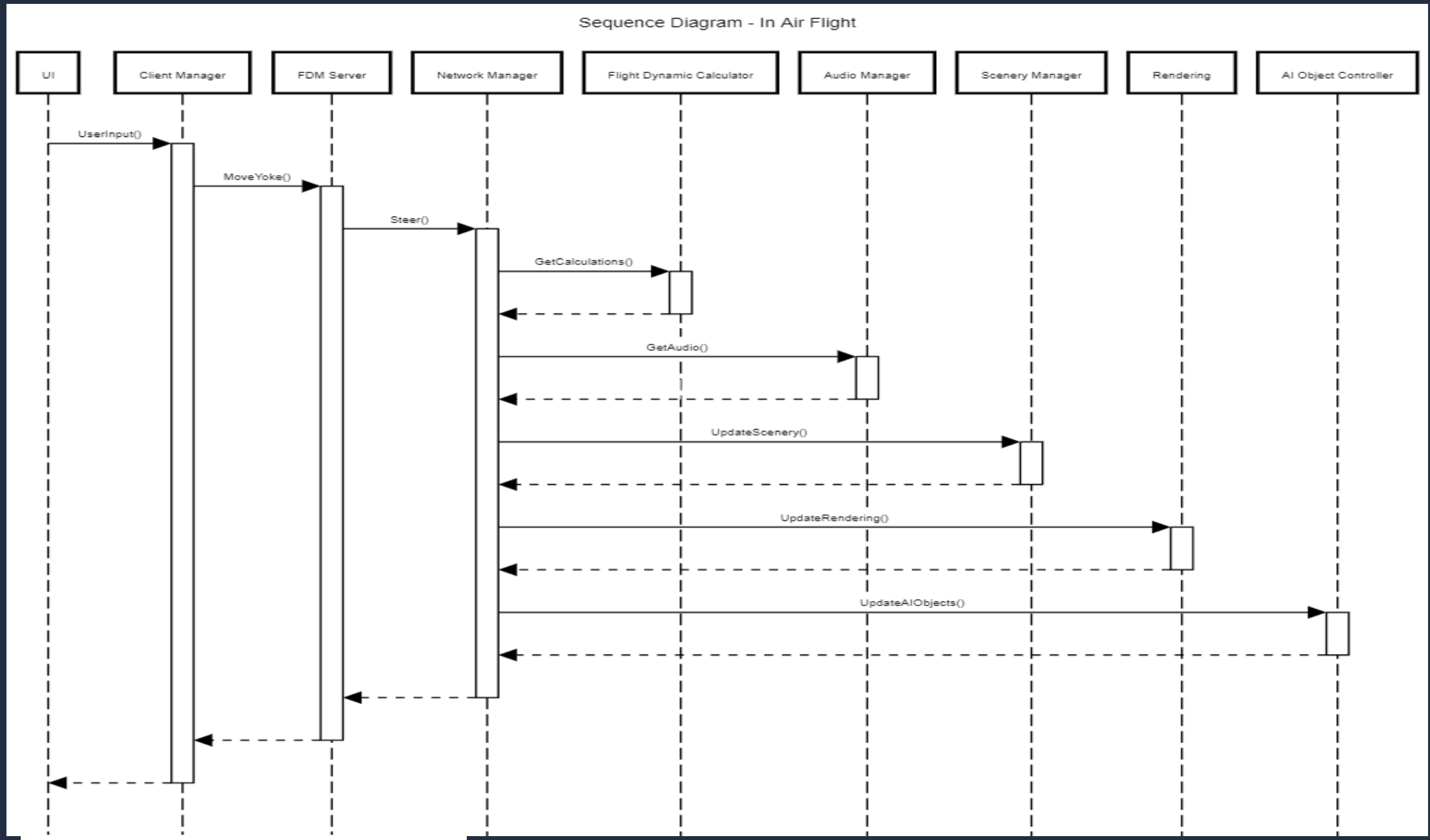
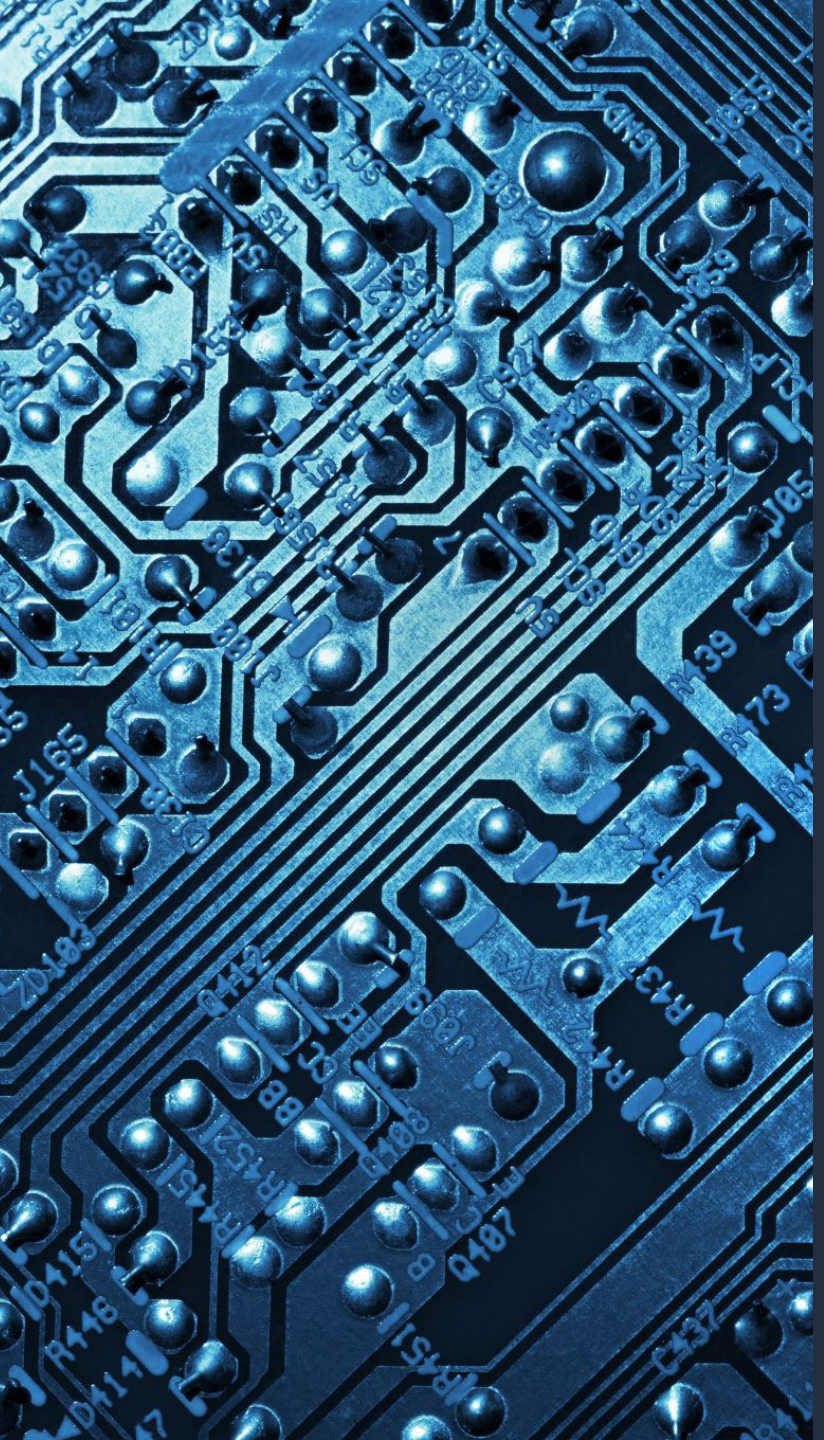


Figure 4: Sequence Diagram of When Flying Aircraft

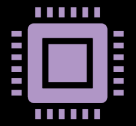


CONCURRENCY

- Evolutions in technology including multi-core processors has improved parallel processing capabilities
- Distributed computing potentially improves processing speed as more computers used, however, this adds challenges
- Threads allow for parallel processing on one machine, semaphores used to sync threads

DIVISION OF RESPONSIBILITIES

Core developers make direct contributions to the source code.



Normal developers can make improvements to the source code, but they need to be reviewed.

Employs High-Level Architecture (HLA), which splits the simulation into different components

This allows anyone to create components in FlightGear that are language flexible.

LESSONS LEARNED

- Most research is from FlightGear Wiki
- Concurrency research from D. Allerton's "Flight Simulation Software", not specific to FlightGear
- Did not compare with other projects such as Microsoft Flight Simulator
- Learned about the benefits and drawbacks of HLA and other practices of FlightGear

CONCLUSION

- The purpose of the system is to accurately simulate aircraft flight in a way that is open, accessible, and easily modifiable for developers and non-developers alike
- It is in part able to accomplish this by creating an environment for threading resource intensive tasks. For example, it uses the FDM server for computing intensive core simulation tasks
- Uses a mixture of MVC, HLA, and Client-Server Architecture