FlightGear Conceptual Architectural Report

Group 1

https://www.youtube.com/watch?v=0jsESB3Lr_M

Group members:

Matthew - Group leader

Mark

Marcus

Armaan

Darcy - Slides/Presenter

Jacob - Slides/Presenter

Derivation Process

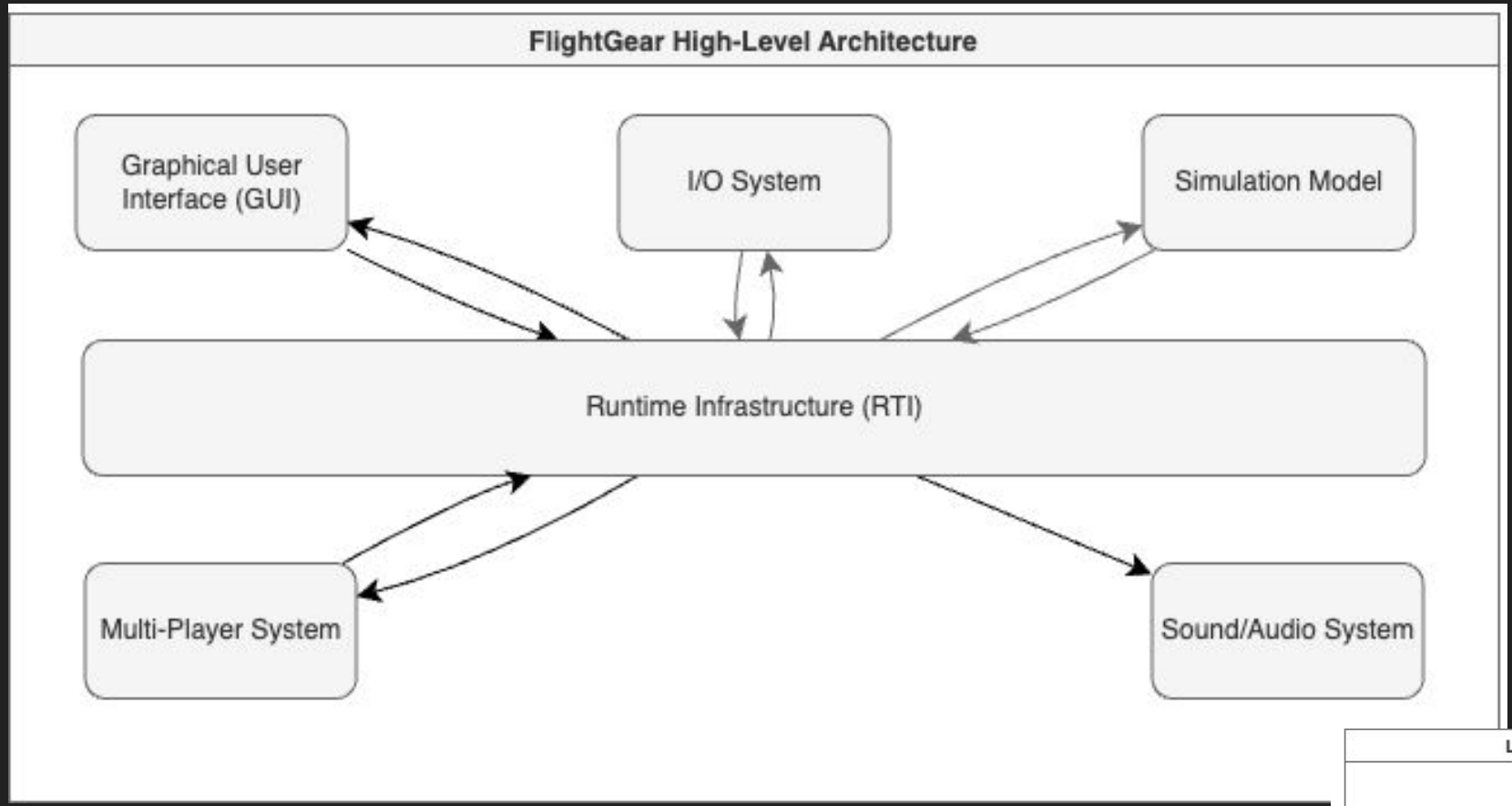
- Examining relevant **academic sources** covering the general development process of industry flight simulators as well as their general design and architecture
  - *Flight Simulator Architecture and Computer System Design and Research* (Guangda Liu, et al.)
  - *Architecture Development of Research Flight Simulator Based on COTS*, (Zhao Guozhu, et al.)

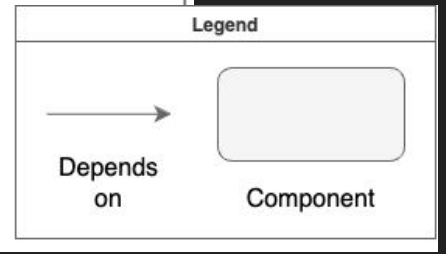
- **FlightGear wiki**
  - Detailed descriptions of components that make up the software.
  - Missing info due to its publicly maintained nature
  - Pieces of information put together to make a cohesive conceptual architecture of the software system.

Conceptual Architecture - Convention

- FlightGear is constructed using the **High-Level Architecture** convention
    - A general purpose architecture particularly used for distributed computer simulation systems.
    - Variant of Implicit-Invocation.
- Useful for applications that must be **reconfigured quickly and often**.

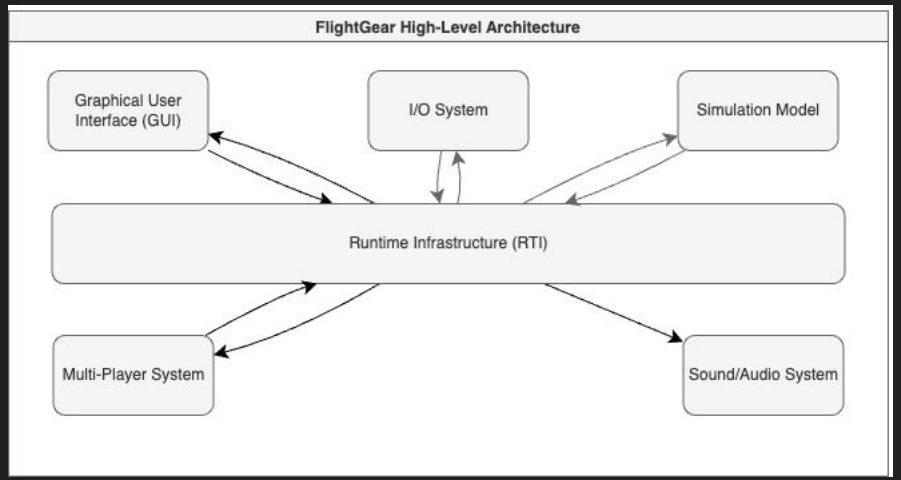
- 6 main components that create the system's main functionality
    1. Runtime Infrastructure (RTI)
    2. I/O System
    3. GUI
    4. Simulation Model
    5. Multi-player System
    6. Audio/sound System


FlightGear High-Level Architecture
Graphical User Interface (GUI)
I/O System
Simulation Model
Runtime Infrastructure (RTI)
Multi-Player System
Sound/Audio System


Legend
Depends on
Component

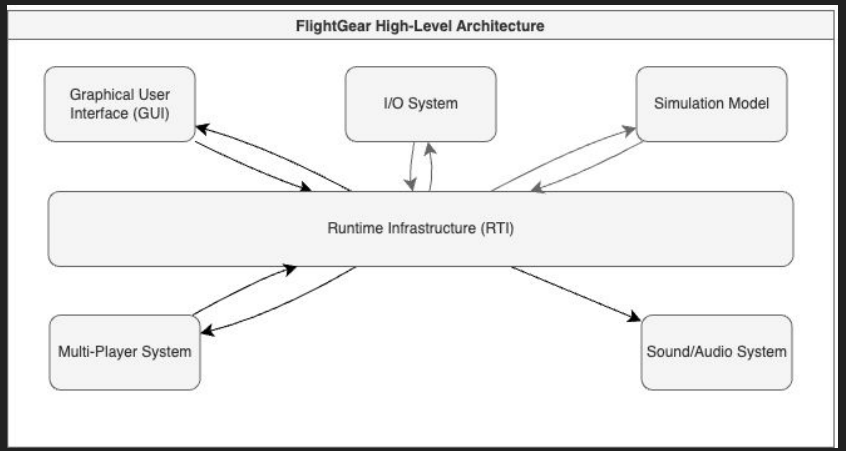
Conceptual Architecture - Runtime Infrastructure

- The pivotal component is the broadcasting system, or RTI.
  - Manages and coordinates/synchronizes distributed simulations.
  - Allows for the simulation to be split into different pieces, providing robust multi-threaded environments to take advantage of computers with multiple cores.
  - Can be run on different computers at once.



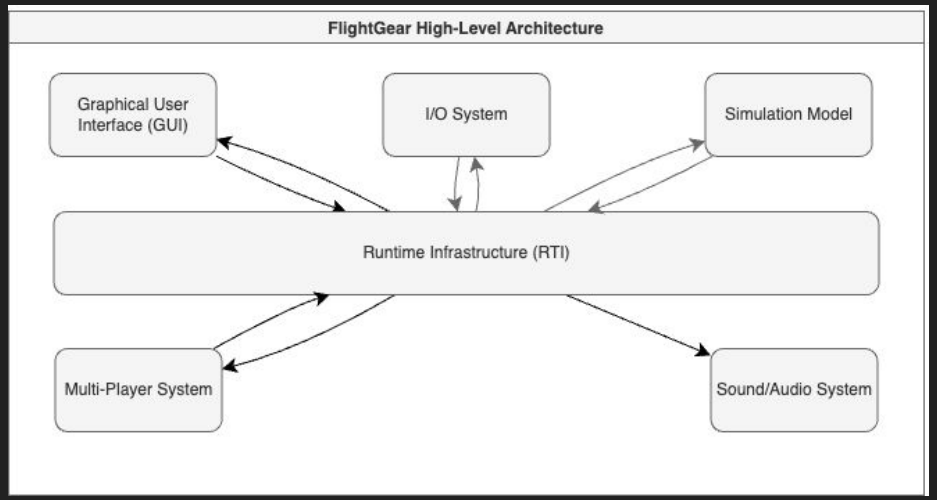
Conceptual Architecture - Federates

- Send and receive messages from the RTI in order to trigger and be triggered by certain events.
- Each has access to the full virtual process address space
- The Federates are designed using the Federation Object Model (FOM) which helps define the objects and their updates which get published by the RTI.



Subsystems - Runtime Infrastructure

- FlightGear uses OpenRTI which is an open source IEEE 1516 standard RTI (written by Mathias Fröhlich).
- This system also took advantage of an interface library (also written by Mathias Fröhlich) which sits on top of the RTI and hugely simplifies interfacing with the RTI and Federates.

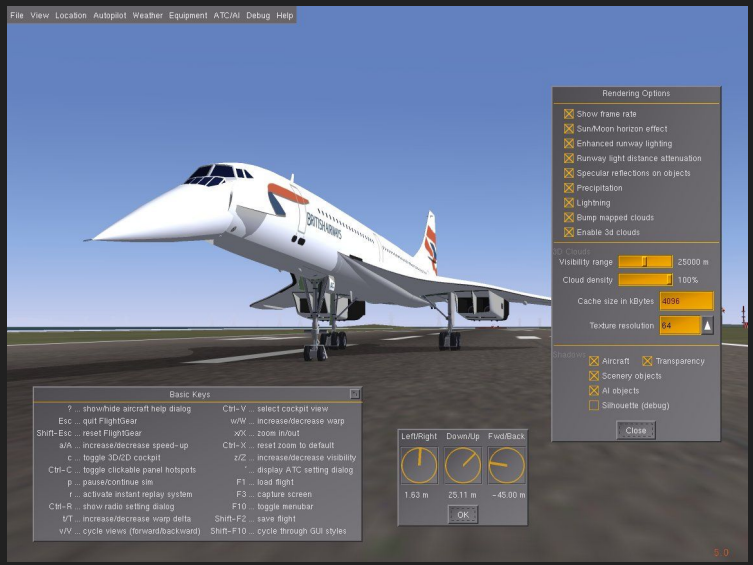


Subsystems - I/O

- Takes inputs from valid input devices and send messages through the RTI to carry out actions
- FlightGear allows users to use their own input devices (joysticks, yokes, rudder pedals, etc.) to control aircraft.
- Input device actions are recognized using an **xml file** which describes which buttons are to be used to control which functions.
- FlightGear comes with preset bindings for most standard input devices but allows users to create their own.

Subsystems - GUI

- Displays the actual simulation and what is functionally possible for the user to do within the simulator.
- This can be partitioned into two separate components:
  - The graphics engine
  - The user interface.

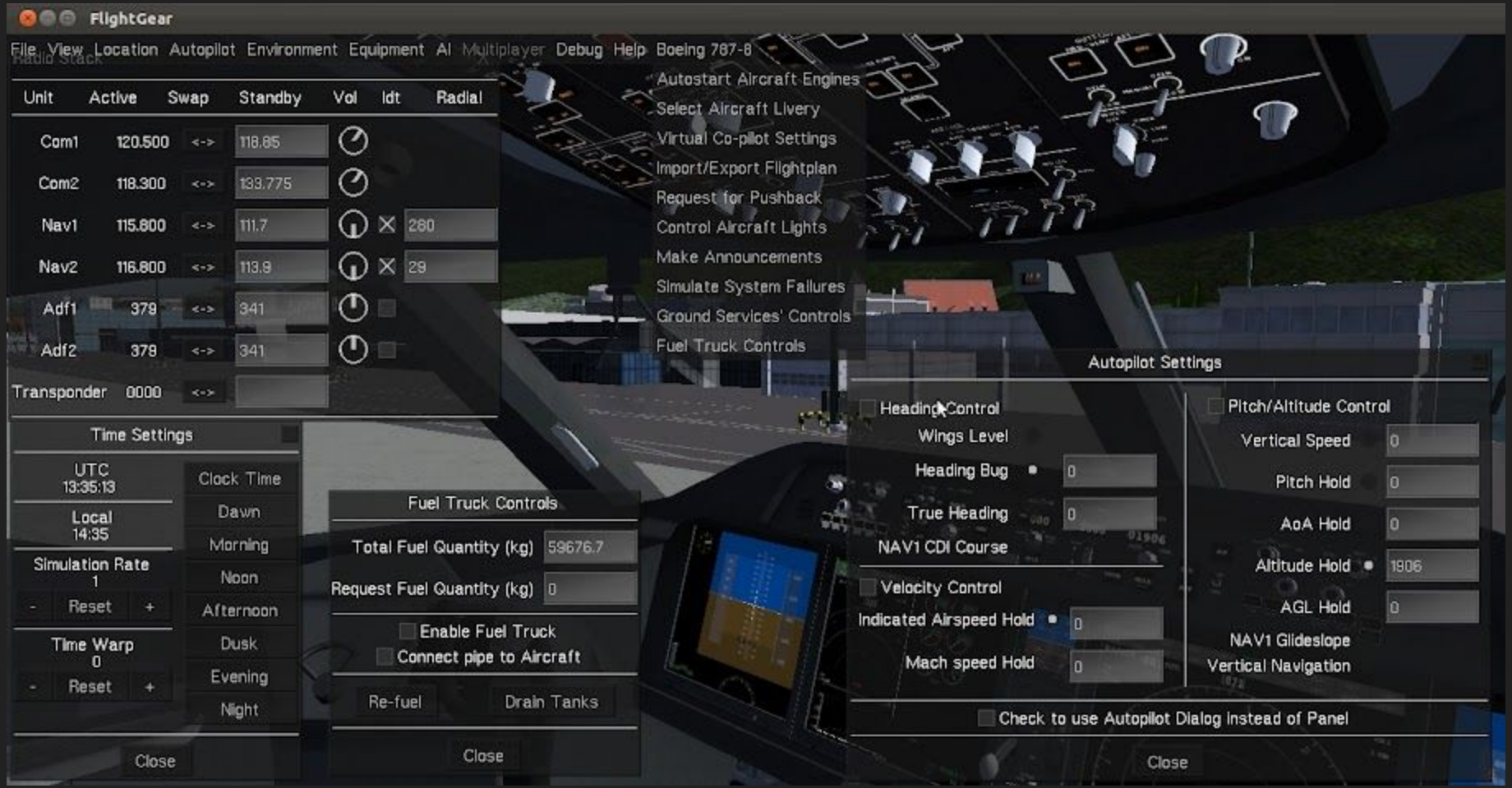


GUI - Graphics Engine

- Responsible for rendering the current state of the simulator
- The rendering system used is primarily “one pass”, where the entire visible scenery is rendered in one single pass.
- Uses 3D graphics toolkit called OpenSceneGraph during its rendering.
    - Benefits include models and scenery being able to load in separate threads, multi-monitor view, and improved precipitation modeling.

GUI - User Interface

- Allows user to customize configuration options and interact with the simulation
  - Menus
  - Dialogs
  - Keybindings
- The UI is strictly limited to elements which the user is able to interact with.
- Users can interact with the UI using a number of different methods
  - Mouse location, mouse clicks, mapped keystrokes, and/or a physical device which allows external physical actions to trigger events.
- The GUI receives messages from the RTI to update what needs to get rendered on screen given the changes in the simulation.
- Sends messages to the simulation to change performance.


FlightGear
File View Location Autopilot Environment Equipment AI Multiplayer Debug Help Boeing 787-8
Radio Stack
Unit Active Swap Standby Vol Idt Radial
Com1 120.500 <-> 118.65
Com2 118.300 <-> 133.775
Nav1 115.600 <-> 111.7 260
Nav2 116.600 <-> 113.9 29
Adf1 379 <-> 341
Adf2 379 <-> 341
Transponder 0000 <->
Autostart Aircraft Engines
Select Aircraft Livery
Virtual Co-pilot Settings
Import/Export Flightplan
Request for Pushback
Control Aircraft Lights
Make Announcements
Simulate System Failures
Ground Services' Controls
Fuel Truck Controls
Time Settings
UTC 13:35:13
Local 14:35
Simulation Rate 1
- Reset +
Time Warp 0
- Reset +
Clock Time
Dawn
Morning
Noon
Afternoon
Dusk
Evening
Night
Close
Fuel Truck Controls
Total Fuel Quantity (kg) 59676.7
Request Fuel Quantity (kg) 0
Enable Fuel Truck
Connect pipe to Aircraft
Re-fuel
Drain Tanks
Close
Autopilot Settings
Heading Control
Wings Level
Heading Bug 0
True Heading 0
NAV1 CDI Course
Velocity Control
Indicated Airspeed Hold 0
Mach speed Hold 0
Pitch/Altitude Control
Vertical Speed 0
Pitch Hold 0
AoA Hold 0
Altitude Hold 1906
AGL Hold 0
NAV1 Glideslope
Vertical Navigation
Check to use Autopilot Dialog instead of Panel
Close

Subsystems - Simulation Model

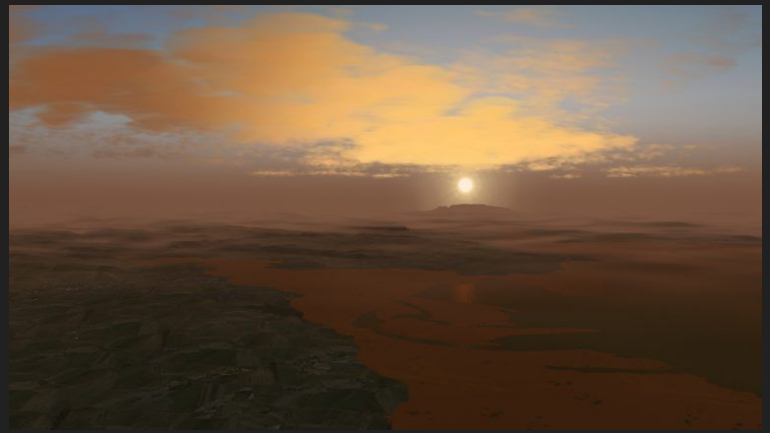
- Handling the actions of the objects within the simulated environment.
  - **Flight Dynamics Model** (FDM)
  - **Weather**
  - **AI system** for air traffic control, air traffic, ground traffic, etc.
- This component sends messages using RTI to receive updates from I/O and Multi-Player systems as well as publishes messages through the RTI to other components.

Simulation Model - Flight Dynamics Model

- The FDM calculates the physical forces acting on a simulated aircraft (e.g., thrust, drag, etc.).
- Every aircraft needs its own bespoke FDM to accurately simulate it
- FlightGear currently uses two different integrated FDMs: **JSBSim** and **YASim**.
    - **JSBSim** supports simulation of airplanes, rockets, helicopters, and some lighter than air aircrafts within the simulator.
    - **YASim** uses different calculation methods to additionally support airplanes and helicopters.

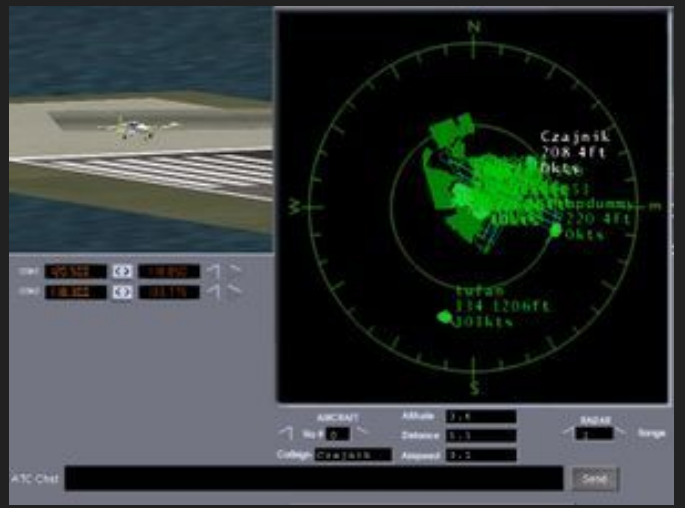
Simulation Model - Weather

- Weather is simulated in FlightGear using one of two weather engines:
  - **Basic Weather** (BW)
  - **Advanced Weather** (AW).
- Accurately simulate real weather fetch, predefined weather scenarios, 3D clouds as well as many other weather related interactions.
- **BW does not consider the effect of terrain on weather**
- **AW allows for the terrain to interact with the weather** and simulate realistic weather distribution.



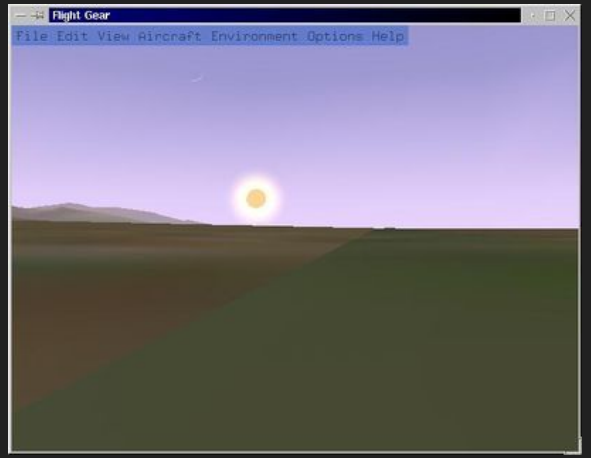
Simulation Model - AI System

- FlightGear operates a variety of semi-intelligent systems to simulate air traffic, ground traffic, etc.
- **Air Traffic Control** (ATC) is used to organize the flow of air traffic.
    - FlightGear's implementation of AI ATC attempts to simulate these tower communications with pilots in the simulation.
- AI air traffic simulates movement of aircraft between airports.
- AI ground traffic simulates cars, trains, and ships.



System Evolution - Initial Code Release

- Developed by Eric Korpela while he was working on his thesis.
- Poorly textured flat land with some mountains.
- After finishing his thesis, development largely halted due to Eric no longer working as the main developer.
- Development of a OpenGL version of the game begins



FlightGear 1.0

- A **multiplayer text and voice chat** system was implemented using **Asterisk**, an open source framework for building communication applications.
- **Improvements to FDMs**, specifically YASim, to allow for water planes, improved gliding, and overhaul of helicopters.
- **Improvements to AI systems**, with an emphasis on taxiing and ground operations of AI aircraft, as well as some smaller changes to ground networks at airports and boats.

FlightGear 1.9.0

- FlightGear was switched from **PLIB** (Portable Game Library) to **OSG** (OpenSceneGraph).
- These libraries are responsible for many systems in the game, such as GUI widgets, audio drivers, joystick support, networking, etc.

FlightGear 2.0.0

- **Major overhaul of the sound systems** of the game allowing for the Doppler effect, distance attenuation, etc.
- **Visual effect overhaul**
  - New 3D clouds, better lighting, dynamic water, etc.

FlightGear 3.0.0

- **New voice chat system** was added called FGCom.
- **New rendering system** to allow for real-time shadows and lighting.
- Built-in **on-the-fly terrain loading** allowed for loading of terrain that is close to the player, instead of needing to load the whole map before playing.
- **Advanced weather simulation** was added to create more realistic weather, such as weather fronts, and cloud formations based on mountain ridges.

FlightGear v2016.1 “San Francisco”

- Versioning scheme was changed from #.# to “year.# *update codename*”.
- **New launcher system** was added to allow for the ability to download certain aircraft, allowing for a reduction of overall package size.
- **Rendering system improvements** to allow for a more realistic cockpit, in-cockpit shadows, dynamic raindrops on the cockpit window, glass reflections, etc.[4]

FlightGear v2017.2 “Boston”

- **YASim FDM** received heavy development.
- **Multiplayer system improvements** to allow for bandwidth reduction

FlightGear v2020.3 “Keflavik”

- **Improvements to the JSBSim and YASim FDMs** to allow for easier development, better debugging, etc.

FlightGear v2020.3 to Present

- Only been small bug fixes since v2020.3
- Developers are in the **process of overhauling** a lot of the major systems such as moving to OpenGL core profile, a new world scenery system, a system to automate the population of the world map based on OpenStreetMap, a new rendering system, etc.

Data Flow

- **User Input**
  - Hardware Interfaces: Incorporates various input devices such as joysticks, yokes, and pedals to capture user actions.
  - I/O System: Translates physical inputs into understandable simulation commands for further processing.
- **Runtime Infrastructure (RTI)**
  - Command Processing: Converts user commands into structured messages and events suitable for simulation.
  - Data Coordination: Manages the exchange of information between different components, ensuring synchronized operation.
- **Simulation Model**
  - State Updates: Receives messages from RTI to update the current state of the aircraft and its surrounding environment.
  - Flight Dynamics Model (FDM): Calculates the aircraft's response to various physical forces like thrust, drag, and gravity.
  - Weather Simulation: Computes the evolving state of weather conditions affecting the flight experience.
  - AI Traffic Management: Uses AI traffic and ground traffic XML files to simulate the state of air traffic control, other aircraft, and ground vehicles.
- **Data Distribution**
  - Information Sharing: Sends the updated simulation state from the Simulation Model back to the RTI.
  - Distribution Network: RTI then disseminates this updated information to the Graphical User Interface (GUI) and the Audio/Sound System for rendering and playback.

Data Flow (Continued)

- **Graphical User Interface (GUI)**
  - Visual Rendering: Utilizes the latest simulation data to generate the next visual frame, displaying the aircraft and its environment.
  - Multiplayer Integration: In multiplayer mode, the GUI also visualizes other players' aircraft, ensuring a cohesive shared experience.
- **Audio/Sound System**
  - Sound Playback: Retrieves and plays various sounds triggered during the simulation, enhancing the realism of the flight experience.
  - Multiplayer Sounds: In multiplayer scenarios, the system also handles audio cues from other players' actions.
- **Multiplayer System**
  - Player Communication: Transmits details of each player's simulation state to the RTI, facilitating interaction between players.
  - Real-Time Synchronization: Ensures that all players experience a consistent and synchronized simulation environment.
- **Sequence Diagrams**
  - Detailed Visualization: Provides a comprehensive graphical representation of the sequential flow of data throughout the architecture, aiding in understanding and troubleshooting.

Concurrency

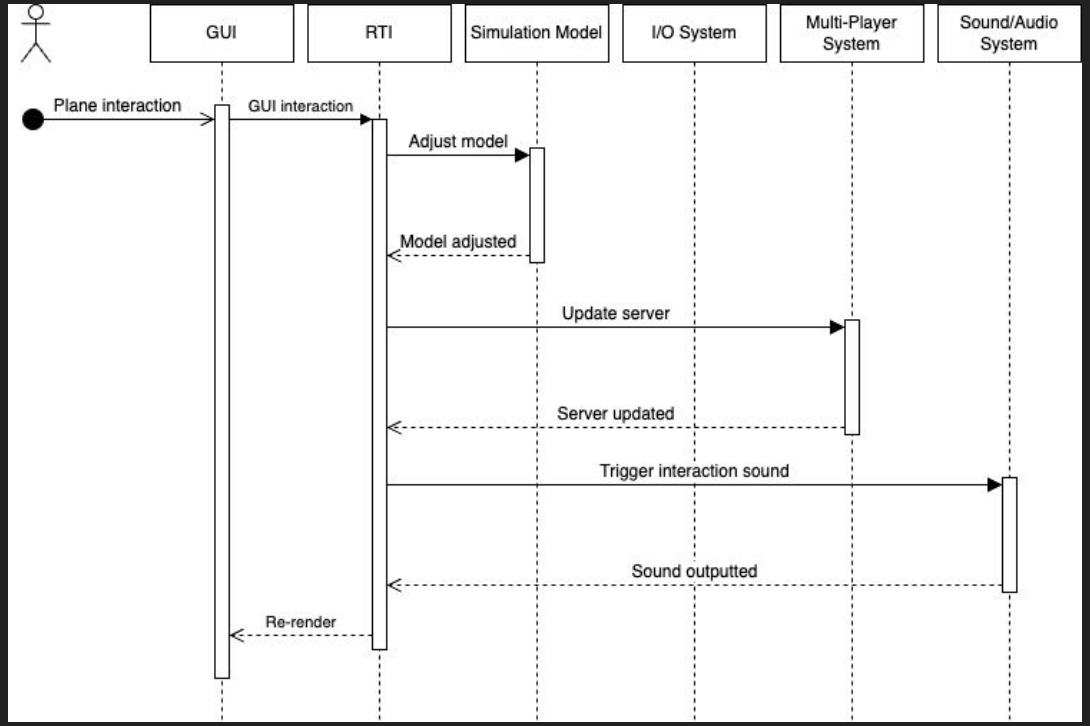
- FlightGear Simulation model utilizes HLA, which divides the system into federates for specific functions
- Separate components of the simulation running on their own threads enables concurrent execution
- The runtime infrastructure manages communication between federates enabling smooth execution of operations

Advantages of Concurrency

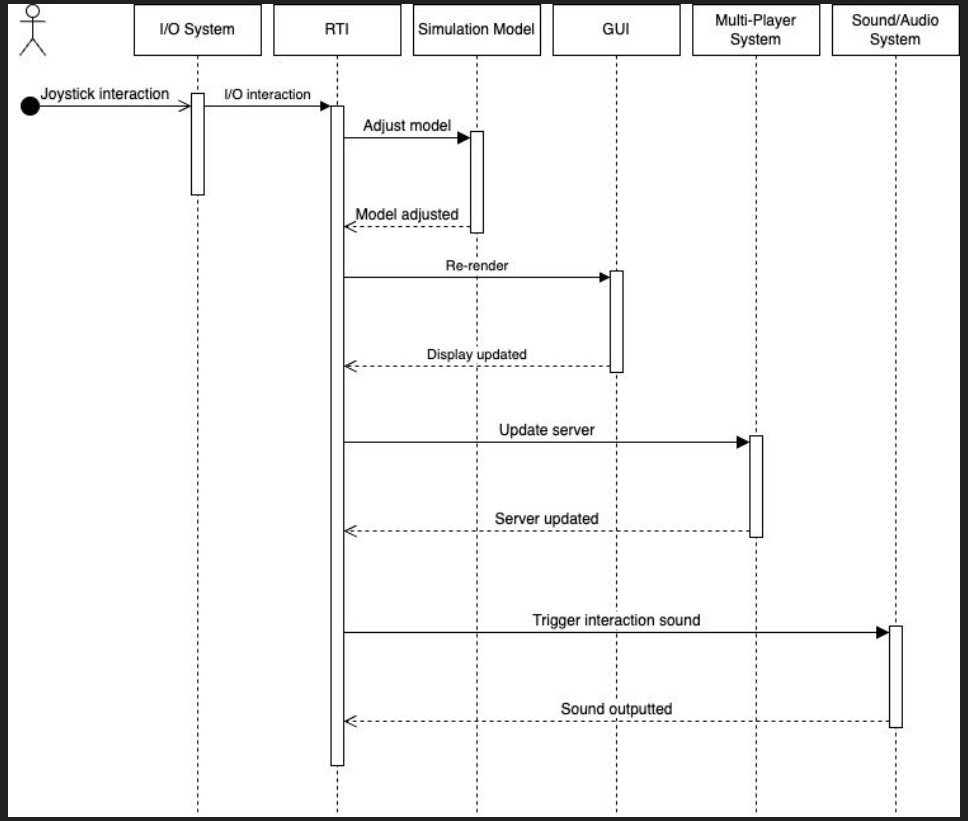
- Enhances performance by utilizing multi-core processors and allowing for execution on different computers
- Isolates critical parts of the simulator (e.g., AI, FDM, Nasal Scripting, Renderer) for consistent frame rates, while less time-critical subsystems (e.g., weather) can run independently
- Provides a framework for adding new components in various programming languages (e.g., Ada, Python, Java) without modifying the core FlightGear code

Sequence Diagrams

Use case #1: User GUI Control:



Use case #2: User I/O Control



Conclusions and Lessons Learned

- **Overview**
  - Comprehensive analysis of FlightGear 2020.3.19 flight simulator.
  - Focus on architecture and subsystems, based on the Implicit-Invocation style.
- **Key Components**
  - Runtime Infrastructure (RTI): Coordinates component interaction and data exchange.
  - I/O System: Manages user inputs and translates them into simulation commands.
  - Graphical User Interface (GUI): Renders visual elements and handles user interactions.
  - Simulation Model: Simulates aircraft behavior and environmental conditions.
  - Multi-Player System: Facilitates online interaction and real-time data sharing between players.
  - Audio/Sound System: Generates and manages auditory feedback and sound effects.
- **System Characteristics**
  - Loosely coupled modules: Enhance flexibility and adaptability of the simulation environment.
  - High-Level Architecture (HLA): Provides a framework for scalability and real-time interaction.
- **Conclusions**
  - FlightGear's architecture demonstrates a collaborative effort in open-source development.
  - The analysis has deepened the understanding of distributed simulation systems.
  - Emphasizes the importance of community-driven development in technological innovation.