Version 1.1

Software Design Document

September 24, 2025

HapTech

Project Sponsor: Dr. Reza Razavian

Project Mentor: Karthik Srivathsan Sekar

Team Members: Landon Coonrod, Matthew Gardner, Peter Hilbert, Karissa Smallwood

Table of Contents

Introduction

Haptic technology is transforming the way humans interact with machines. By bridging the physical and virtual worlds, it enables users to experience force and tactile feedback, creating the sensation of physically interacting with digital objects. Haptic technology comes in various forms, from touch-sensitive surfaces that simulate textures to wearable devices that provide pressure and vibration feedback.

Graspable haptic technology has broad applications across various fields, including bomb disposal, space exploration, and medical training. Robots equipped with haptic technology either simulate or actually perform critical tasks with precision and safety. For example, medical students can practice procedures on virtual patients with haptic technology providing realistic sensations of incisions and suturing, without risking the well-being of a real person. In addition, robots are regularly used to dispose of toxic substances and explosives, which are remotely controlled using haptic technology to feel any forces the robot is exposed to in real time [1].

These applications of haptic technology are driven by the growing research area of human-robot interaction, which Dr. Reza Sharif Razavian, our project sponsor, is involved in. Dr. Razavian is an assistant professor of mechanical engineering at Northern Arizona University, where his research lab, Raz Lab, integrates robotics, neuroscience, and biomechanics. One of his key research areas involves using the Franka Research 3 (FR3) robot in human-robot interaction studies. In these studies, participants are given a task to complete in a 3D simulation, in which they move the robot's arm to control a virtual object. The robot applies forces back, simulating collisions or other movement restrictions, making the participant feel as if they are controlling a physical object. The insights from these studies are crucial for advancing applications such as motor learning, physical rehabilitation, and the design of intuitive robotic interfaces, where understanding the nuances of touch and movement can lead to more effective human-robot collaboration.

The labs current workflow requires objects to be hardcoded into the system which drastically reduces the efficiency of operations. With the system lacking modularity, creating new experiments is slowed, and reproducing results is difficult. This project aims to solve this issue by offering a modular, intuitive system that utilizes configurable settings an operator UI loads, runs and monitors in multi trial sessions. The participants will interact with these 3D objects via Franka with realtime force feedback and a data pipe line will record robot states, events, and meta data for analysis. Figure 1 shows the high level architecture of our system. Together these components aim to reduce experiment setup time, improve repeatability, and offer high quality data, while preserving the low latency behavior required for haptic interactions.

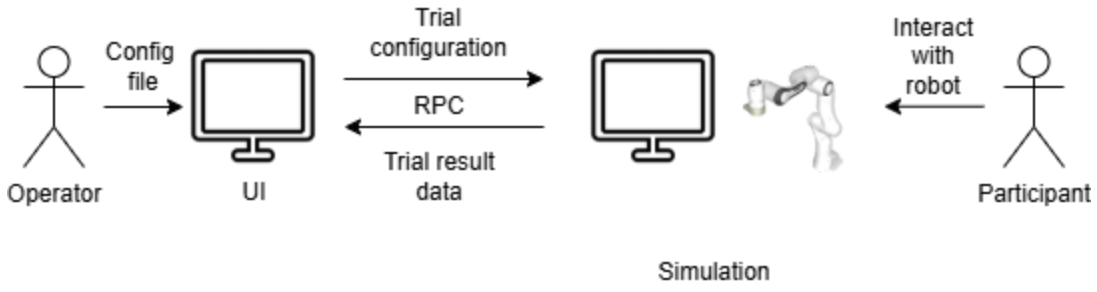


Fig. 1 - High-Level System Diagram

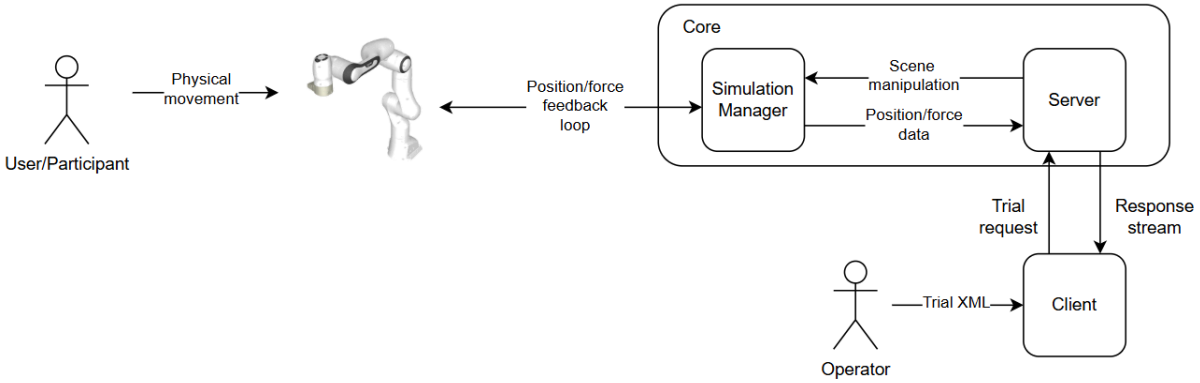
Implementation Overview

Our solution is to implement a modular, flexible system that allows operators to configure different trials using an XML file. The participant will interact with 3D objects through Franka with real-time haptic feedback. The system will log the experiment data and capture the metadata and configurations. Overall, our goal is to improve experiment templates, reduce setup time, and increase performance.

The overall architecture of our system will be a client-server design. The client is the operator side that sends trial requests in the form of XML config files to the server. The server manages trial execution and forwards the config files to the Simulation Manager where it sets up the scene. RPC is used for client-server communication. The core simulation, which is CHAI3D and the Simulation Manager, handles the graphics and the force feedback on the robot. The participant interacts with the robot through movement of the arm and receives that force feedback.

The technologies and tools we've chosen are CHAI3D, the haptic rendering engine that handles 3D scenes and force computations. Libfranka is the robots API library used for precise, real-time control of the Franka Research 3. gRPC/RPC is the communication protocol used between the Client and Server over Ethernet. C++ is the primary implementation language used to code the project as well as used for its performance. Lastly, we are using CSV logging because it is lightweight and its structured format makes it easy to capture trial results.

Architectural Overview



The system follows a client-server architectural pattern, in which the client, or the UI that the experiment operator is using, will send trial requests to the server via RPC, which contains trial configuration data parsed from an XML file. The server will process this request, setting up the trial, which the participant will interact with via the robot. The server will stream data points containing position and force data, as well as trial status updates back to the client in real time for the operator to see, and for the client to ultimately write the data points to a CSV file for permanent storage. Below is a more detailed discussion of each major component of the system.

Core

The "core" is made of two main components: the simulation manager and the server. The simulation manager provides the main CHAI3D simulation which is being interacted with. It is responsible for initializing and maintaining the graphical window and the 3D scene within it, and provides an interface for the server to manipulate the scene. Finally, it is also responsible for the haptics, which involves taking in robot input and providing the force feedback on the robot. The server refers to the RPC server which listens for trial requests from the client, applies scene manipulations to manage the trial, and reads position and force data from the simulation to stream back to the client. The simulation manager and the server work hand-in-hand to manage trials.

Client

The client is a command line interface for the trial operator to configure and start trials. It parses the XML configuration file provided by the operator and sends an RPC request to the server over the network to start the trial. It then reads the response stream containing status updates and data points, and writes those data points to a CSV file.

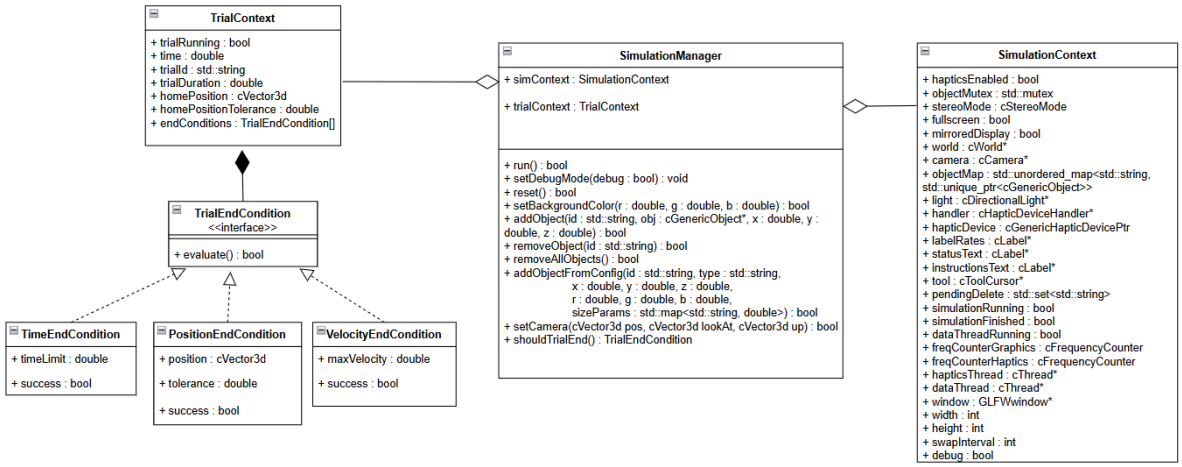
Franka-CHAI3D integration

There is an integration component which facilitates communication between CHAI3D and Franka robot using the libfranka controller library. This component provides the interface for CHAI3D to read the position of the robot and apply force commands based on interactions between objects in the simulation, creating a feedback loop between CHAI3D and the robot. This feedback loop must be fast and robust, so it is necessary that the computer running the simulation and the robot are connected to one another via Ethernet.

Module and Interface Descriptions

Core - Simulation Manager

The SimulationManager is designed as a singleton class and is responsible for setting up and tearing down the CHAI3D simulation and integrating with Franka. It also provides an interface for the server to manipulate the scene as it receives requests. Below is a UML class diagram illustrating the design of SimulationManager and its related classes.



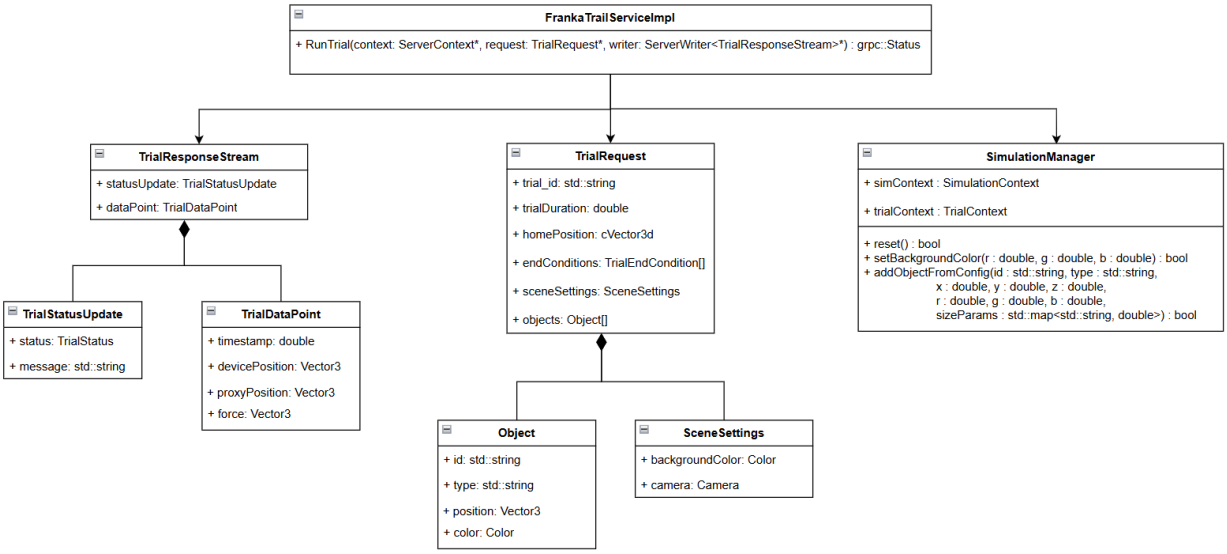
Below are the important methods in the SimulationManager's public interface which the server will use to manipulate the scene and run trials. Note that these scene manipulation methods return a bool value which indicates its success.

- **run()**
  Initializes the display window, sets up the CHAI3D environment, and connects to the robot, preparing the simulation environment for trials. This method is called when the server first starts up.
- **reset()**
  Resets the scene, removing objects and restoring scene settings back to their defaults. This method is called by the server when the trial ends.

- **setBackgroundColor()**
  Can be used to set the background color of the scene with the given red, green, and blue values.
- **addObject()**
  Inserts CHAI3D object into the scene at the position given by x, y, and z, and assigns it the given ID.
- **removeObject()**
  Removes the object in the environment with the given ID.
- **removeAllObjects()**
  Removes all current objects from the environment, used when the trial ends.
- **addObjectFromConfig()**
  Is a wrapper around addObject, which takes in configuration values from the server, such as the shape type, color, and size parameters specific for different kinds of objects, and adds that object into the environment. This is called by the server when it receives a trial request and begins populating the environment with the objects specified in the configuration data it receives.
- **setCamera()**
  Orients the camera using the given position data.
- **shouldTrialEnd()**
  Evaluates the trial end conditions in the trial context and determines if the trial should end. It iterates through each condition, and returns the one that is determining trial end. If none of them are evaluating to true, it returns null. This method is polled periodically by the server to determine whether or not to end the trial, and will use the condition on which the trial ended to provide feedback to the client.

Core - Server

The Server acts as the intermediary between the client and the simulation/robot. It is responsible for accepting trial requests from the client. Then it parses the trial configurations and applies commands to the Simulation Manager. The Server also reads simulation and Franka data and sends updates back to the client.



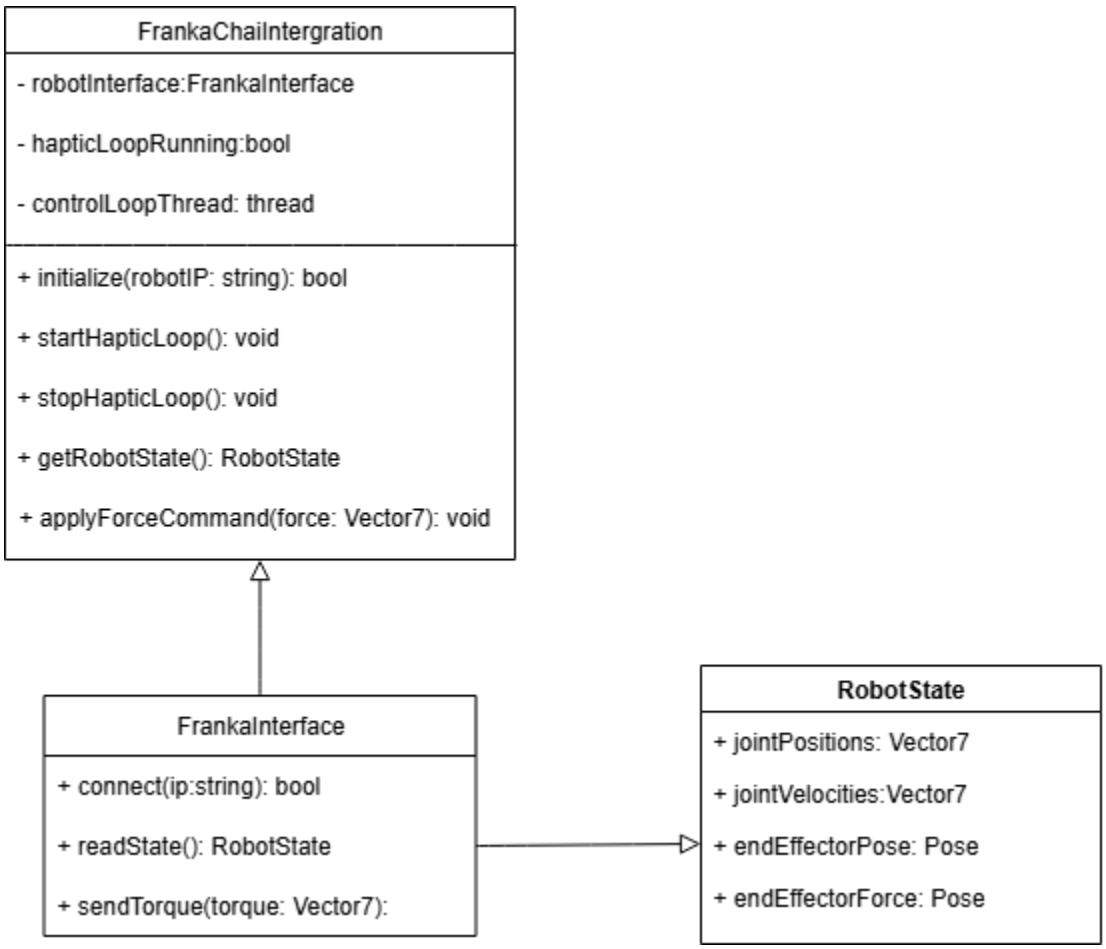
Below are the important methods used in the Server's public interface, which communicates between the Client and the simulation/robot.

- **RunTrial()**
  Is the server's only purpose. It is the entry point for running a trial. It validates the simulation state, loads the trial objects, and sets the trial parameters. It also streams trial status updates and trial data points back to the client.
- **TrialStatusUpdate()**
  Sends updates on the trial's current state, used for real-time monitoring.
- **TrialDataPoint()**
  Contains sampled position and force data collected during the trial.
- **trial_id()**
  A unique string identifier for the trial, Used to track and distinguish trials in logs.
- **trialDuration()**
  A double value specifying how long the trial should run in seconds.
- **homePosition()**
  A 3D vector defining the robot's starting position before the trial begins.
- **endCondition()**
  A list of conditions that determine when the trial stops.
- **reset()**
  Clears trial progress and prepares the system for a new run.
- **setBackgroundColor()**
  Changes the background of the simulation environment.
- **addObjectFromConfig()**
  Loads and adds an object to the environment based on details provided in a configuration file.

CHAI3D-Franka Integration

The CHAI3D–Franka Integration module serves as the bridge between the CHAI3D environment and the Franka Research 3 robot. Its primary responsibility is to create a closed-loop feedback system that enables the robots to interact with objects in the simulation in real time. This involves reading the robot's joint positions, velocities, and torques through libfranka, feeding that information into CHAI3D's engine, and then sending computed force feedback from CHAI3D back to the robot.

This module is critical to achieving the project's goal: providing participants with a realistic, low-latency haptic experience. It fits into the larger system by allowing the simulation manager to stay least impactful to the hardware interface details. The simulation manager simply queries the integration module for robot state and issues force commands through its interface**.**



Listed below are the important methods in the CHAI3D-Franka Integration public interface. The main purpose is to make a connection between the CHAI3D environment and the Franka Research 3 robot.

- **initialize(robotIP: string) : bool**

Connects to the Franka robot using its IP address. Must be called before starting the haptic loop.

- **startHapticLoop() : void**
  Launches the real-time control loop in a dedicated thread, continuously polling robot state and updating CHAI3D simulation.
- **stopHapticLoop() : void**
  Gracefully terminates the haptic control loop and sends zero torque to the robot.
- **getRobotState() : RobotState**
  Returns the latest joint positions, velocities, end-effector pose, and force data from the robot.
- **applyForceCommand(force: Vector3) : void**
  Applies a Cartesian force command to the robot's end-effector based on CHAI3D's haptic computation

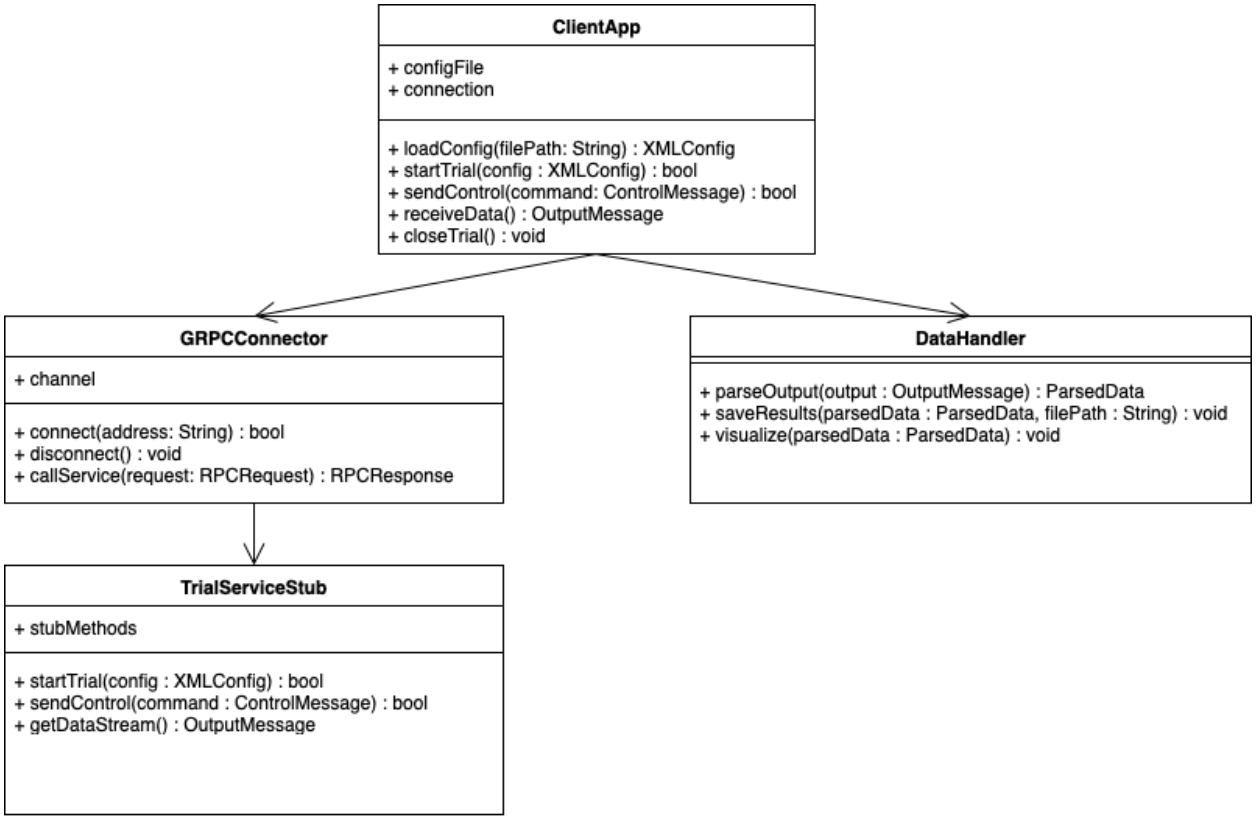
This component's design isolates all hardware-specific complexity in a single integration layer, allowing other modules (like the Simulation Manager) to remain hardware-agnostic. It provides a clear, testable, and maintainable interface achieving the project's real-time haptic feedback goals.

Client

The client module serves as the external interface for end-users and external systems to interact with the simulation environment. Running on a separate machine from the main simulation server, the Client starts and manages the simulation trial through a gRPC connection. The clients responsibilities include:

- **Trial Initialization**: The client sends an XML configuration file to the server to specify simulation parameters and start the trial.
- **Control Inputs**: The client provides control inputs to the simulation, such as ending trial early, pausing etc.
- **Data Retrieva**l: The client receives and processes the real-time output data from the simulation, enabling real time monitoring, visualization, and analysis
- **User Interaction**: Exposes a simple interface to researchers/operators, removing gRPC complications to allow full focus on usability.

In the larger architecture of the project, the Client is the entry point for the external users of the product. It communicates with the Simulation Server via gRPC, enabling distributed experimentation and collaboration.

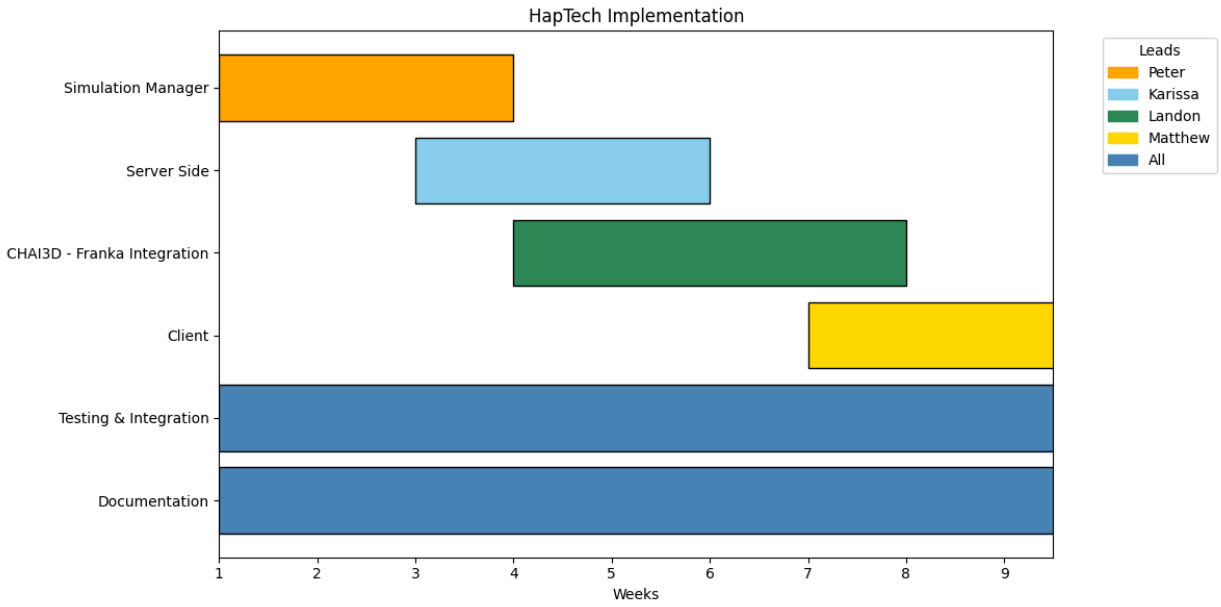


Below are the important methods in the **Client's public interface**, which users will use to start, control, and monitor simulation trials through gRPC. These methods expose an interaction layer while hiding underlying networking complexity, making it much easier for researchers to conduct and interact with the experiments being performed.

- **loadConfig(filePath: String): XMLConfig**
  Loads and parses an XML configuration file that specifies simulation parameters. Returns the configuration object to be sent to the server.
- **startTrial(config: XMLConfig): bool**
  Sends the XML configuration to the Simulation Server and starts a new trial. Returns `true` if the trial was successfully initialized.
- **sendControl(command: ControlMessage): bool**
  Transmits control commands (pause, stop, resume) to the running trial. Returns `true` if the server accepts the command.
- **receiveData(): OutputMessage**
  Receives the latest stream of simulation output data from the server. This can be used for real-time monitoring, visualization, and analysis.
- **closeTrial(): void**
  Gracefully ends the current trial session and closes the connection with the server.

- **connect(address: String): bool**
  Opens a gRPC channel to the Simulation Server. Returns `true` if the connection succeeds.
- **disconnect(): void**
  Closes the gRPC connection after the trial or when the client application shuts down.
- **parseOutput(output: OutputMessage): ParsedData**
  Processes raw output data from the simulation into a structured form usable by researchers.
- **saveResults(parsedData: ParsedData, filePath: String): void**
  Persists processed trial results to disk for later review or analysis.
- **visualize(parsedData: ParsedData): void**
  Provides live visualization of simulation results during the trial, enabling researchers to track outcomes in real time.

Implementation Plan



The implementation of the HapTech project will follow a design-centric, phased schedule that emphasizes modular development, early testing, and seamless integration. The timeline is structured to ensure that each subsystem is designed, implemented, and validated before integration into the larger system. This approach reduces risk and allows the team to identify and address issues early.

Development Phases

1. Core - Simulation Manager (Weeks 1 - 3)
   Establishes the simulation environment that coordinates and schedules the various trail components. Ensures easy accessibility and access for future modules and changes to the simulation system.
   - Lead: Peter
   - Support: Landon
2. Core - Server Side (Weeks 3 - 5)
   Handles data transfer, communication, and trial coordination with the client side. Provides gRPC endpoints for said communication between the different systems.
   - Lead: Karissa
   - Support: Matthew
3. CHAI3D - Franka Integration (Weeks 4 - 7)
   Connects together the CHAI3D framework with the Franka Robotic arm, enabling the haptic interactions and force feedback
   - Lead: Landon

    - Support: Peter
4. Client (Weeks 7 - 9)
   Provides a user interface that runs on a separate computer. Users start trials via XML config files and interact with real-time simulation data.
    - Lead: Matthew
    - Support: Karissa

Notes on Scheduling

- Parallel Development: Hardware and software will progress in parallel
- Iterative Testing: Testing is embedded throughout development, not only in the formal testing phase
- Non-visible Activities: Activities like documentation, and other class related details will run continuously in the background of development

Conclusion

HapTech project represents a step toward improving human-robot interaction research in Dr. Razavian's lab. By designing a modular, configurable system, we are reducing experiment setup time and making it easier to reproduce results for future studies. Our client-server architecture, real-time data streaming, and integration with CHAI3D and the Franka Research 3 robot allow researchers to focus on designing experiments rather than manually managing objects or scenes.

The design we have outlined in this document provides a clear roadmap for development, with each module described in detail and a timeline that ensures testing and integration are prioritized. Once implemented, this system will contribute to more efficient workflows, higher-quality data collection, and ultimately better research outcomes in motor learning and rehabilitation.

As a student team, this project also allows us to apply classroom knowledge to a real-world problem, collaborate with our peers, and develop skills in software architecture, robotics integration, and client-server communication. We are confident that the system we are building will have a positive impact on both the lab's research and our own professional growth.

References

[1] Spiceworks, "What Are Haptics?," [Online]. Available: https://www.spiceworks.com/tech/tech-general/articles/what-are-haptics/. [Accessed: 26-Mar-2025].