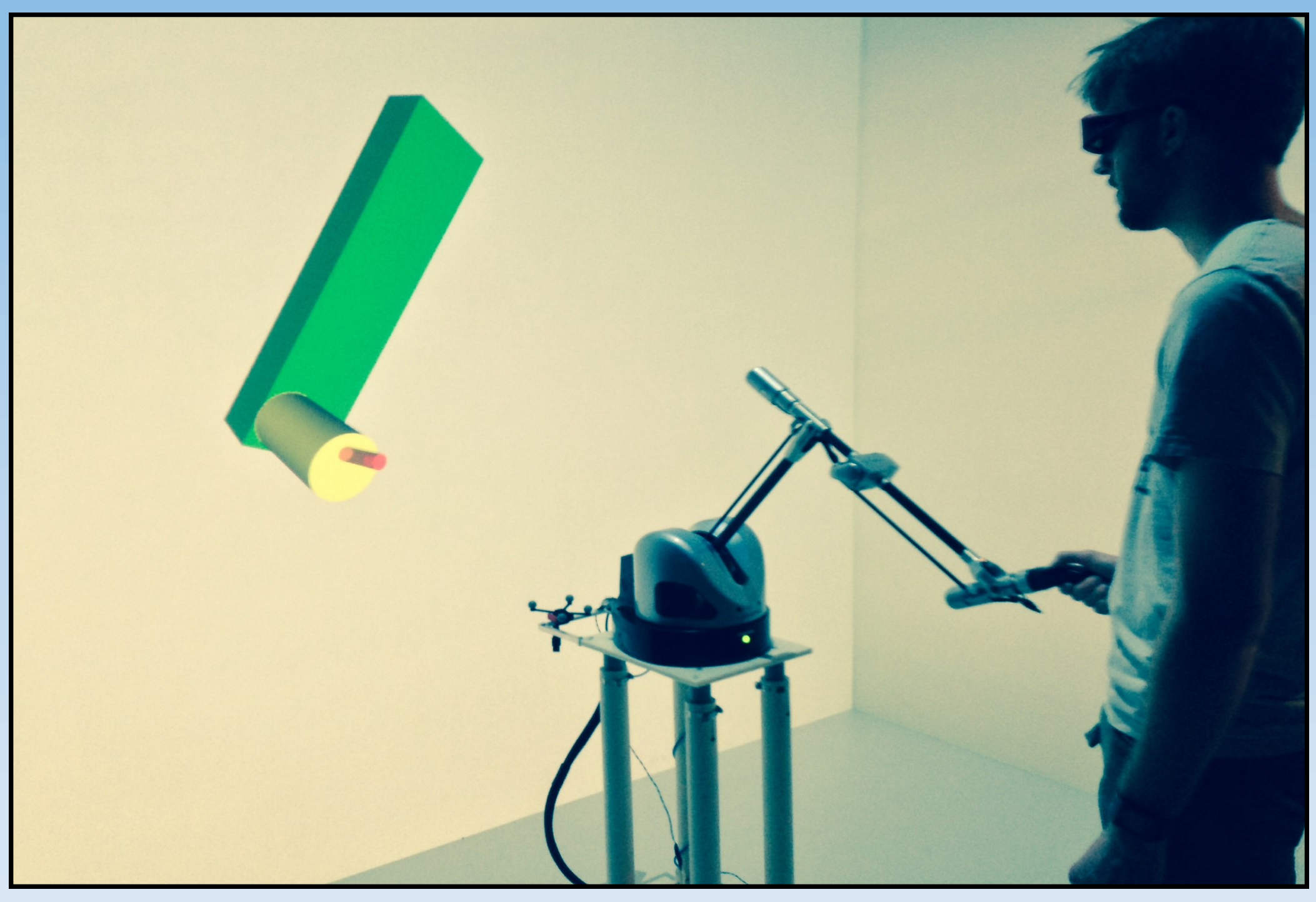
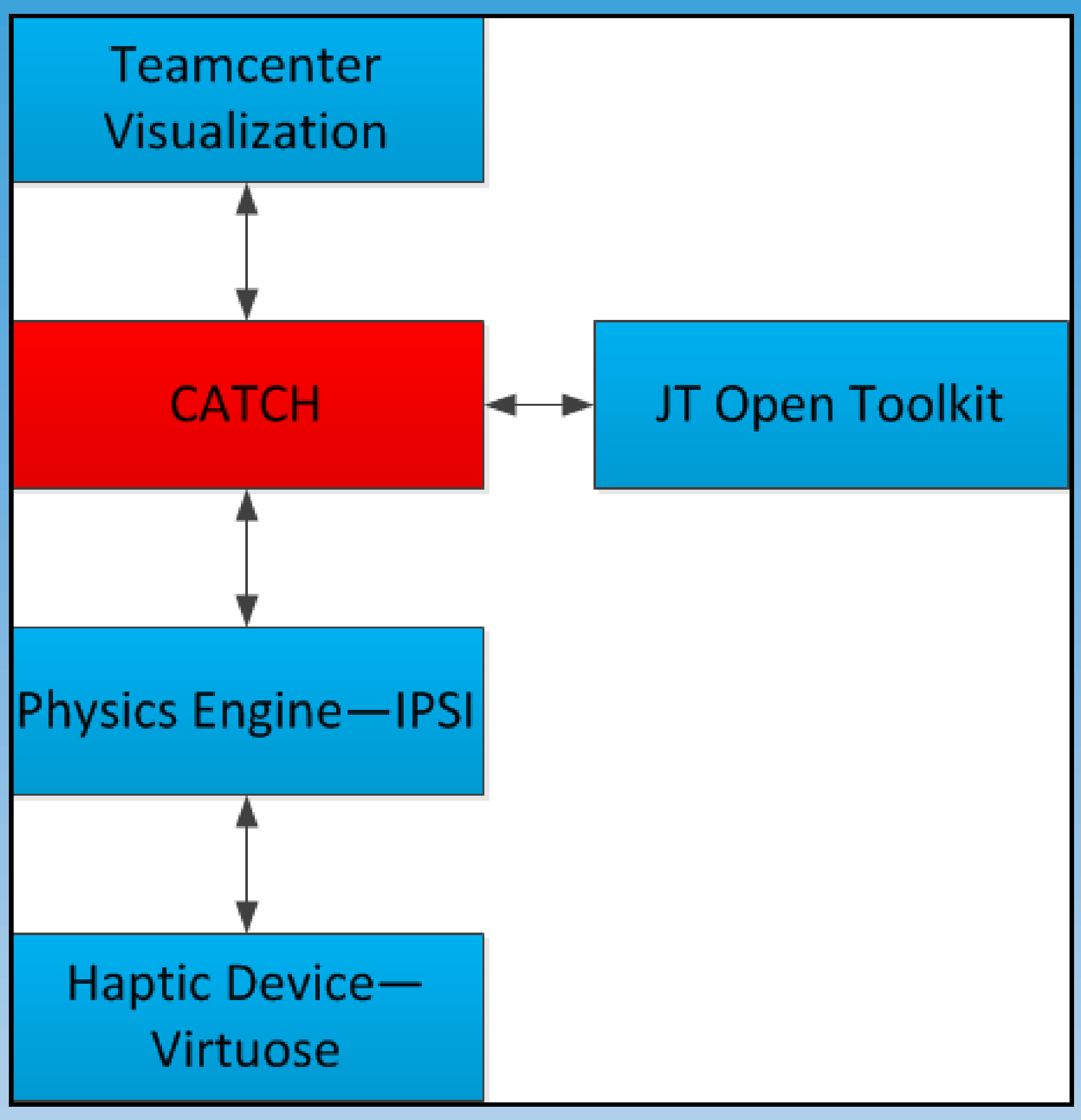
CATCH: Collision Detection and Team Center Haptics

Introduction

The field of virtual reality (VR) has advanced greatly in the past few years. One factor continuing to slow the advancement of virtual reality is its lack of reality. Haptic technology is adding this missing reality to the virtual world by introducing tactile feedback to virtual interactions. This allows the user to experience virtual objects through the use of force feedback, vibration, or motion. While researchers like Dr. Vance have had success making use of these devices, many companies that would like to use VR technology do not use haptics in their design process for two reasons. The first is that the devices themselves are very expensive. The second issue is a lack of commercial software support for the integration of haptic technology with existing 3D modeling software. CATCH seeks to demonstrate the feasibility of integrating a commercial 3D visualizer with haptic devices.





Project Scope

We are designing and creating the CATCH library. All other components including Teamcenter Visualization, JT Open Toolkit, the IPSI physics engine, and the Virtuose haptic device were provided by our clients.

Requirements

Functional

- A cursor in the 3D visualizer will be manipulated by the haptic device
- Parts in the 3D scene can be selected, deselected, and manipulated by the haptic device
- Parts will be loaded into the scene from a standard 3D modeling file type
- Collisions between bodies will be calculated and haptic feedback is to be provided based on these collisions *(this is primarily performed by IPSI, not our software)*
- The deliverable must run in a Windows 7 x64 environment
- The deliverable must be capable of interacting with 3D models with simple 3D geometry
- The deliverable must be capable of running in a CAVE environment

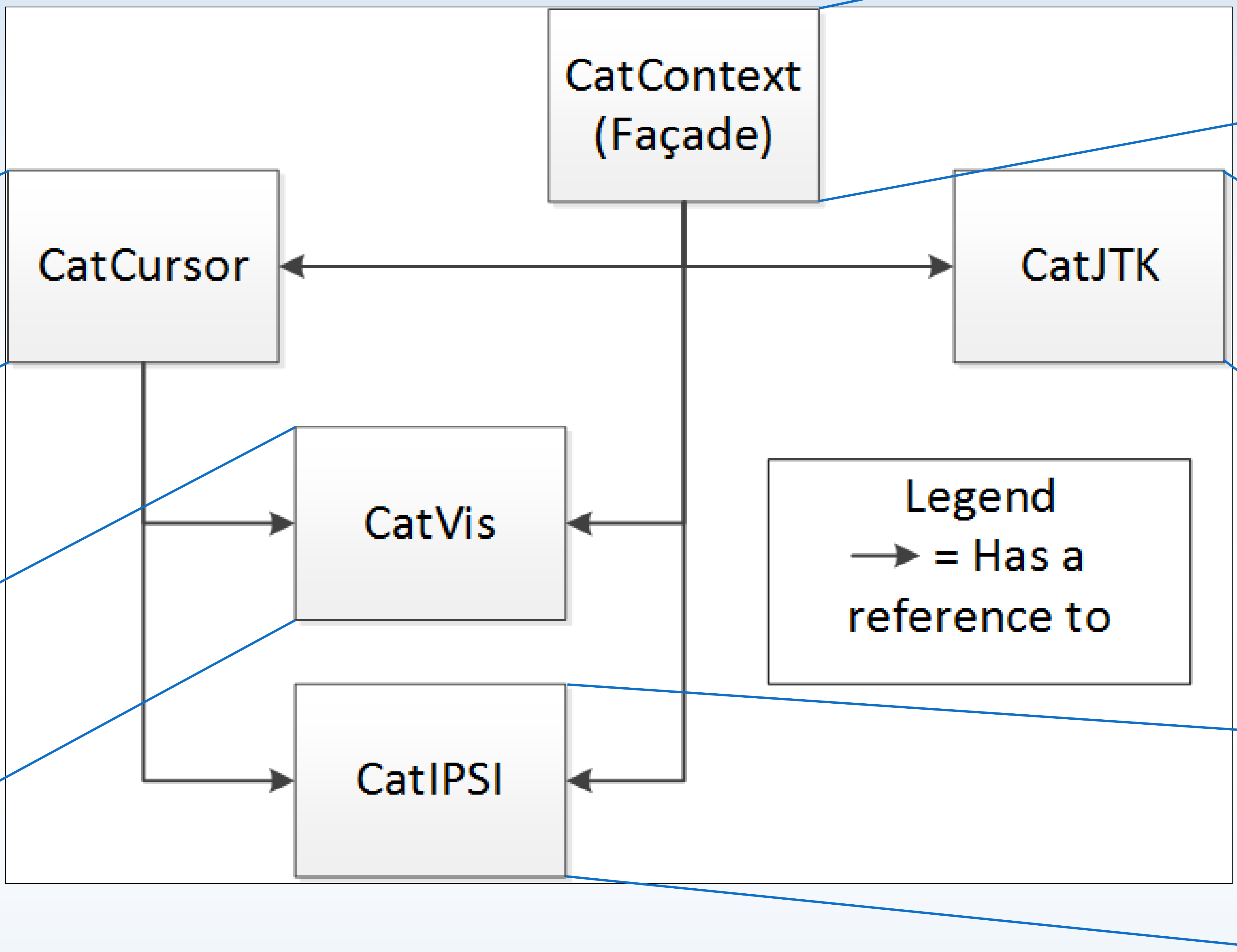
Non-Functional

- Teamcenter Visualization Mockup must be used as the 3D visualizer
- The modules must be written in C++ using Microsoft Visual Studio 2010
- The VisController API must be used to interface with the 3D visualizer
- The JT Open Toolkit API must be used for importing Jupiter Tessellation (JT) files into the physics engine scene
- The IPSI physics engine must be used for collision detection and interactions with the haptic device
- The software must interface with a Virtuose haptic device
- There must be less than 200ms lag time between input and output for best user experience
- After accounting for lag all objects must be synchronized between the physics engine and the visualizer

Use Cases

- Future VRAC projects to expand software capabilities (Users: VRAC students)
- In-house testing module at Siemens for the VisContoller API (Users: Siemens developers)

Module Diagram



CatContext

CatContext is the Facade class of the CATCH library. It includes all construction, initialization, and callback functions for the CATCH library. CatContext also includes all of the functions an end user of CATCH needs to call.

CatCursor

CatCursor contains the primary process flow of the CATCH library. It is responsible for controlling the execution speed and most inter-module communication.

CatJtk

CatJtk is responsible for converting Jupiter Tessellation (JT) files into trimesh objects that can be added to the physics simulation scene.This is done through the use of the JT Open Toolkit API. CatJtk is used when loading a JT assembly file into the physics scene.

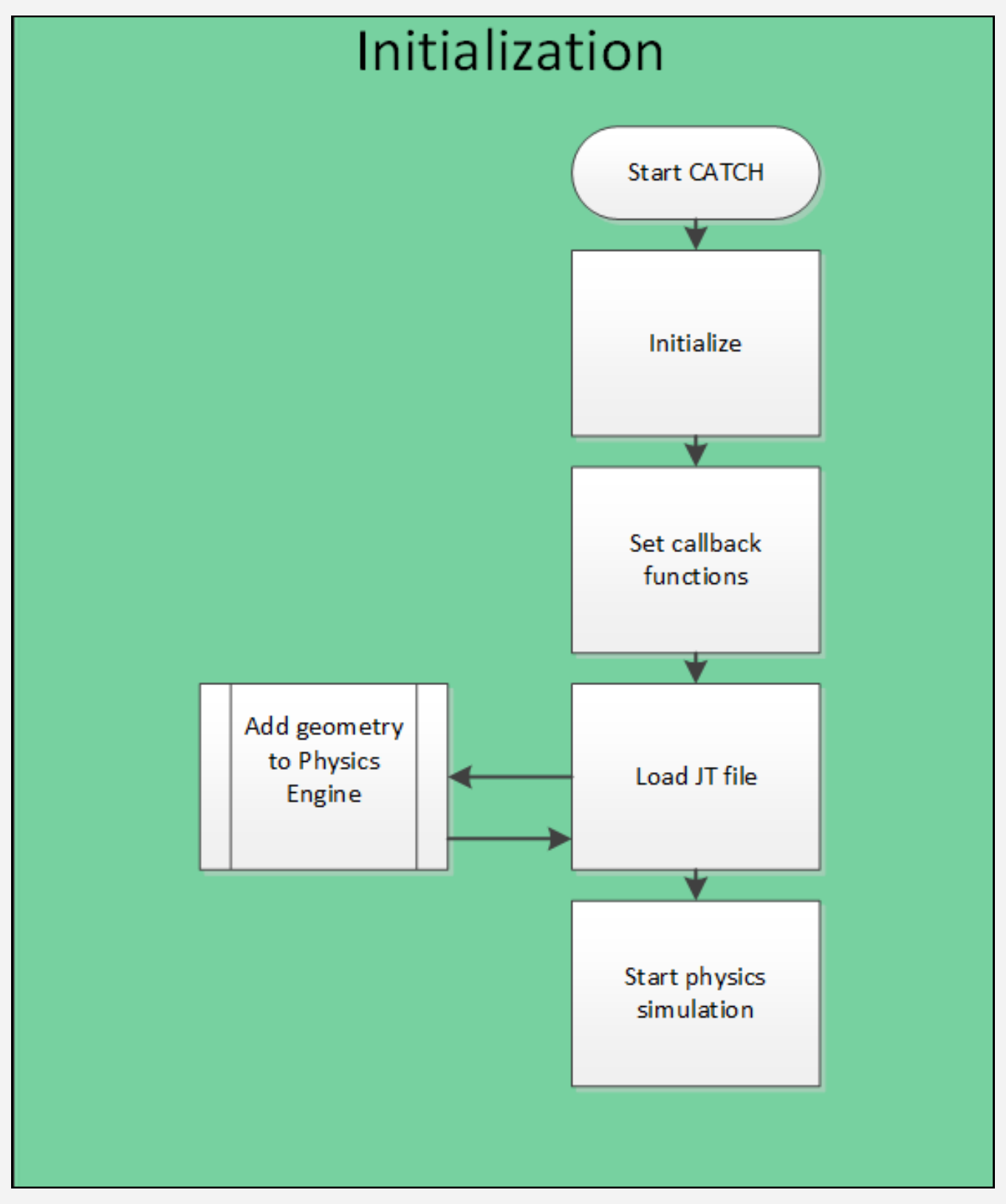
CatVis

CatVis facilitates communication between the visualizer and the rest of the program. This module sends click messages to the visualizer and receives various updates which it processes and forwards to appropriate modules.

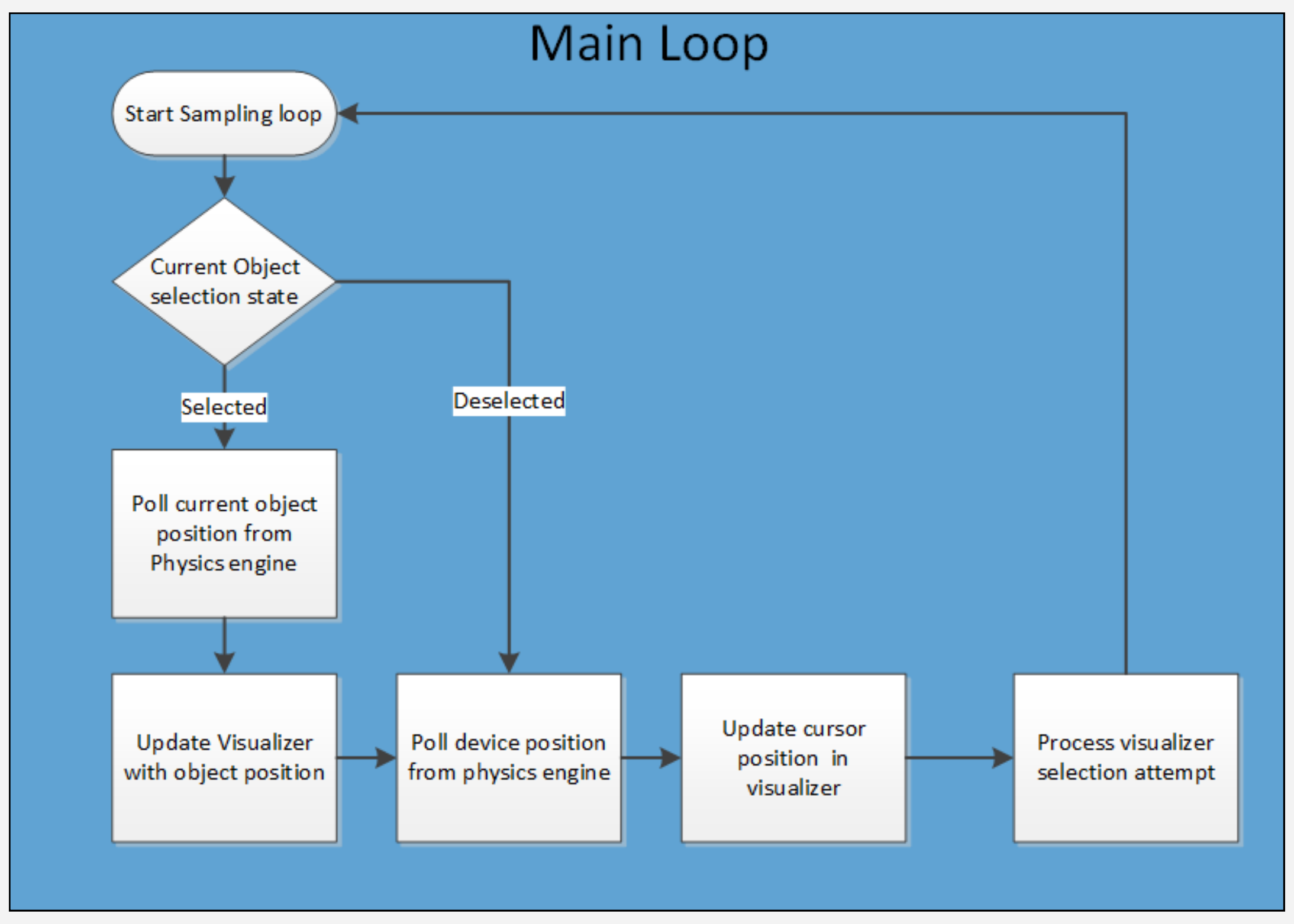
CatIPSI

CatIPSI is responsible for interacting with the physics scene and the haptic device. The simulation is run by the IPSI physics engine API. CatIPSI is responsible for setting up the simulation, loading provided objects, and polling device and object positions within the simulation scene, as well as any other interactions with IPSI.

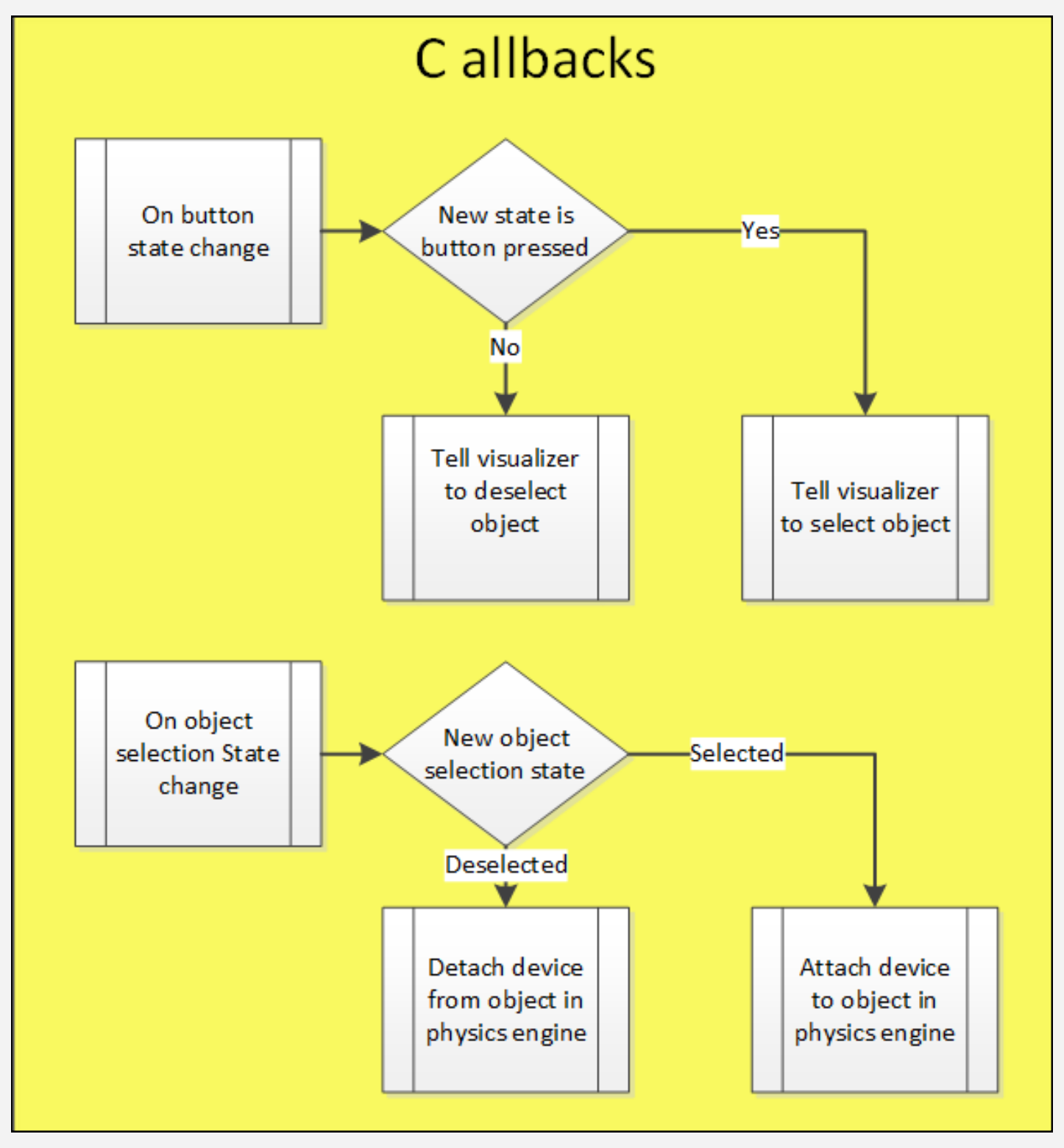
Functional Block Diagrams



Initialization phase: In this phase, all of the callbacks for the modules are initialized and the scene is loaded in from a JT file.


Main Loop
Start Sampling loop
Current Object selection state
Selected
Deselected
Poll current object position from Physics engine
Update Visualizer with object position
Poll device position from physics engine
Update cursor position in visualizer
Process visualizer selection attempt

Main Loop: This is the loop that runs until the program is terminated. The loop samples the state of the physics engine and updates the visualizer with the new position of all of the objects in the physics engine. Then, the device position in the physics engine is used to update the cursor position in the visualizer.


C allbacks
On button state change
New state is button pressed
Yes
No
Tell visualizer to deselect object
Tell visualizer to select object
On object selection State change
New object selection state
Selected
Deselected
Detach device from object in physics engine
Attach device to object in physics engine

Callbacks: The callbacks are steps that run occasionally based on event input.

The top process triggers on a button state change (if the button is pressed or released) and attempts to select/deselect an object in the visualizer.

The bottom process triggers when CatVis is notified of a state change from the visualizer. At this point, depending on the message received from the visualizer the program will attempt to attach or detach an object from the cursor.

Testing

Due to the nature of the project as a proof-of-concept, it was decided that formal testing wasn't necessary. Instead, demos of the current project progress were shown at client meeting. This not only helped with showing our current progress to the client and validate our design, but it also helped us set reasonable goals that could be accomplished between client meetings.

Each demo showed new or improved functionality. Initial testing was not done with the haptic device but with either a six-degree-of-freedom mouse or Virtuose Sim, a program that simulates the haptic device. Once enough functionality was implemented, it was initially planned that CATCH would be tested with the haptic device. By the time this point was reach in early March 2014, the haptic device had to be sent to the manufacturer in France for emergency repairs. The haptic device retuned on April 17th, 2014. At this point, we were able to test our library with the device in METaL and validate that we successfully meeting all functional requirements set forth by the client.

Outcome

The CATCH library has full functionality at this point. It is capable of interfacing with the haptic device, providing force feedback, and synchronizing the scene in the physics engine with the visualization scene. Overall, CATCH successfully meets the requirements set forth by the client.

May 14-30:

http://seniord.ece.iastate.edu/may1430

Team:

Logan Scott
Matt Mayer
James Erickson
Anthony Alleven
Paul Uhing

Clients:

Dr. Judy Vance — Virtual Reality Applications Center
Dr. Tsung-Pin Yeh — Siemens

Advisor:

Dr. David Weiss