Android VirtuTrace Remote (VT Remote)

Project Plan II

Group 21

Kollin Burns
Tanner Borglum
Sheil Patel
Alexander Maxwell
Lukas Herrmann

Project Overview

Currently, the process for testing a scene within the C6 is rather cumbersome. If a user is inside the C6 and notices something in the scene that they want to change, they must exit the chamber, shut down the simulation, edit the configuration file, restart VirtuTrace, step back into the cave, and wait a few more minutes for the scene to load.

Our project's goal is to completely deprecate this entire process. With an Android application, we will allow the user to modify a scene in real-time and save the results without ever having to leave the C6 or stop the simulation. Additionally, the app will provide some useful features to aid in debugging the current simulation.

User Interface Prototypes and Descriptions

Scene Graph Tab (Figure 1A): This tab will allow the user to change the properties of a desired Scene Graph object. The parent node will be listed in a list view on the left side and the options for that particular object will be displayed on the right. For example, the Visible toggle button will change whether the object appears in the scene and can be manipulated at runtime.

While this prototype only shows a few options, more properties will be available, including a color chooser to change the color of objects and various text boxes for rotating/skewing object in 3D space (a text box for each coordinate in 3D space, x, y, z). The details that are displayed will vary based on what object is currently selected, only displaying modifiable properties of the given object.

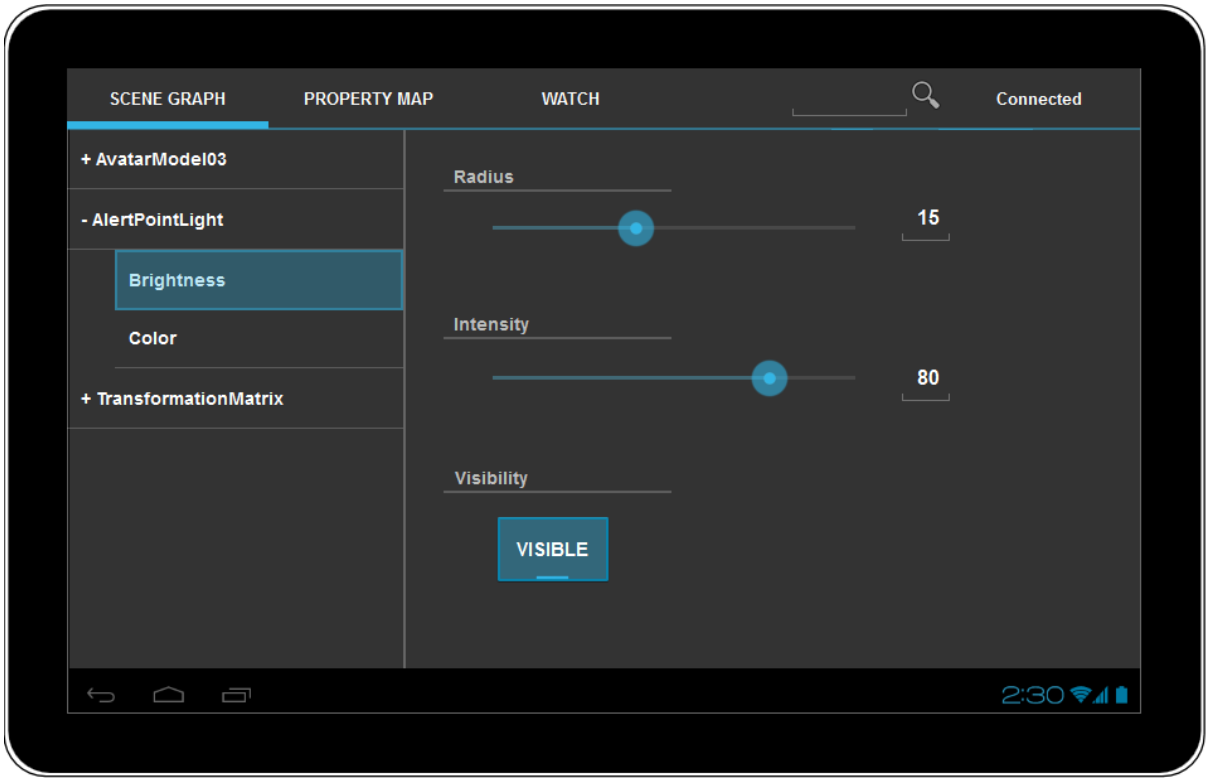


Figure 1A

The **Property Map Tab** will be similar to the Scene Graph tab in terms of design and layout.

Watched Tab (Figure 1B): The Watched tab will show various debug information for any object that has been tagged as watched. Example of information will be information about the object position and information about how the object has been changed thus far. The user will be able to select a Watched Object on the left hand side and see the various debugging information on the right hand side.

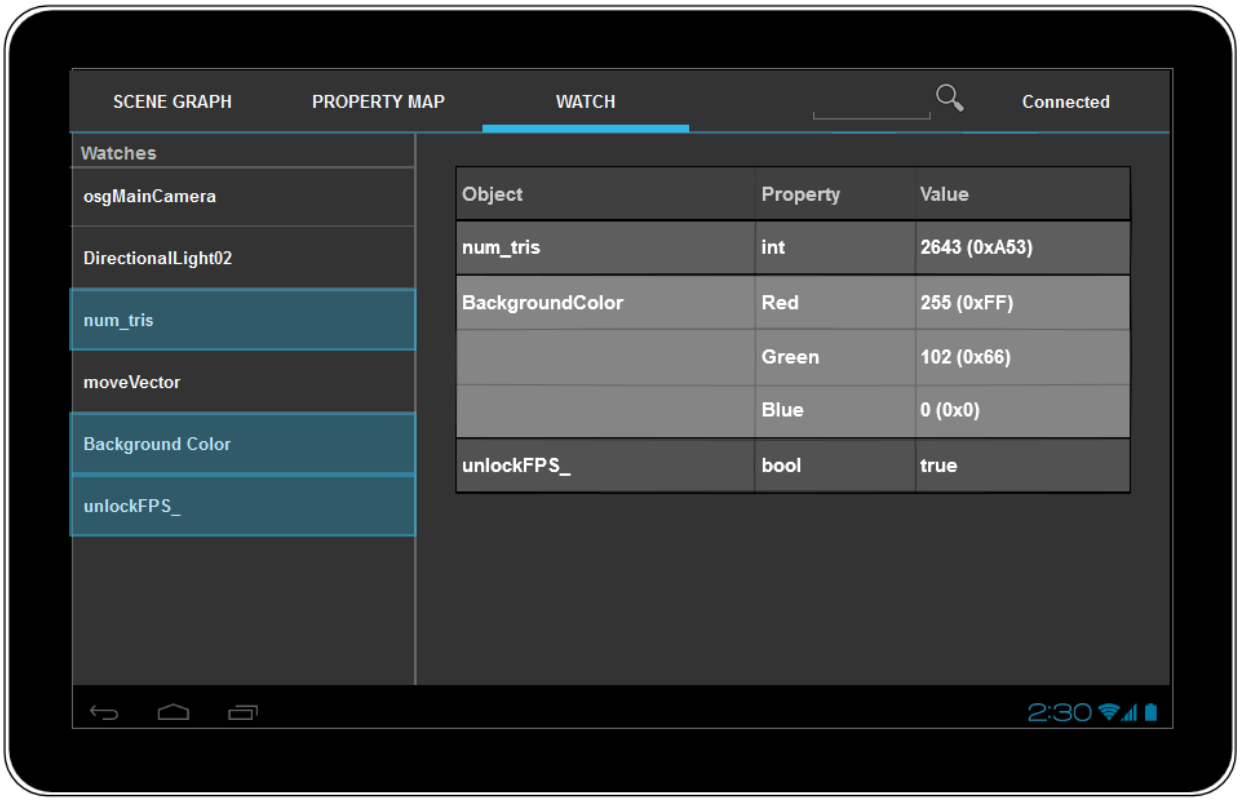
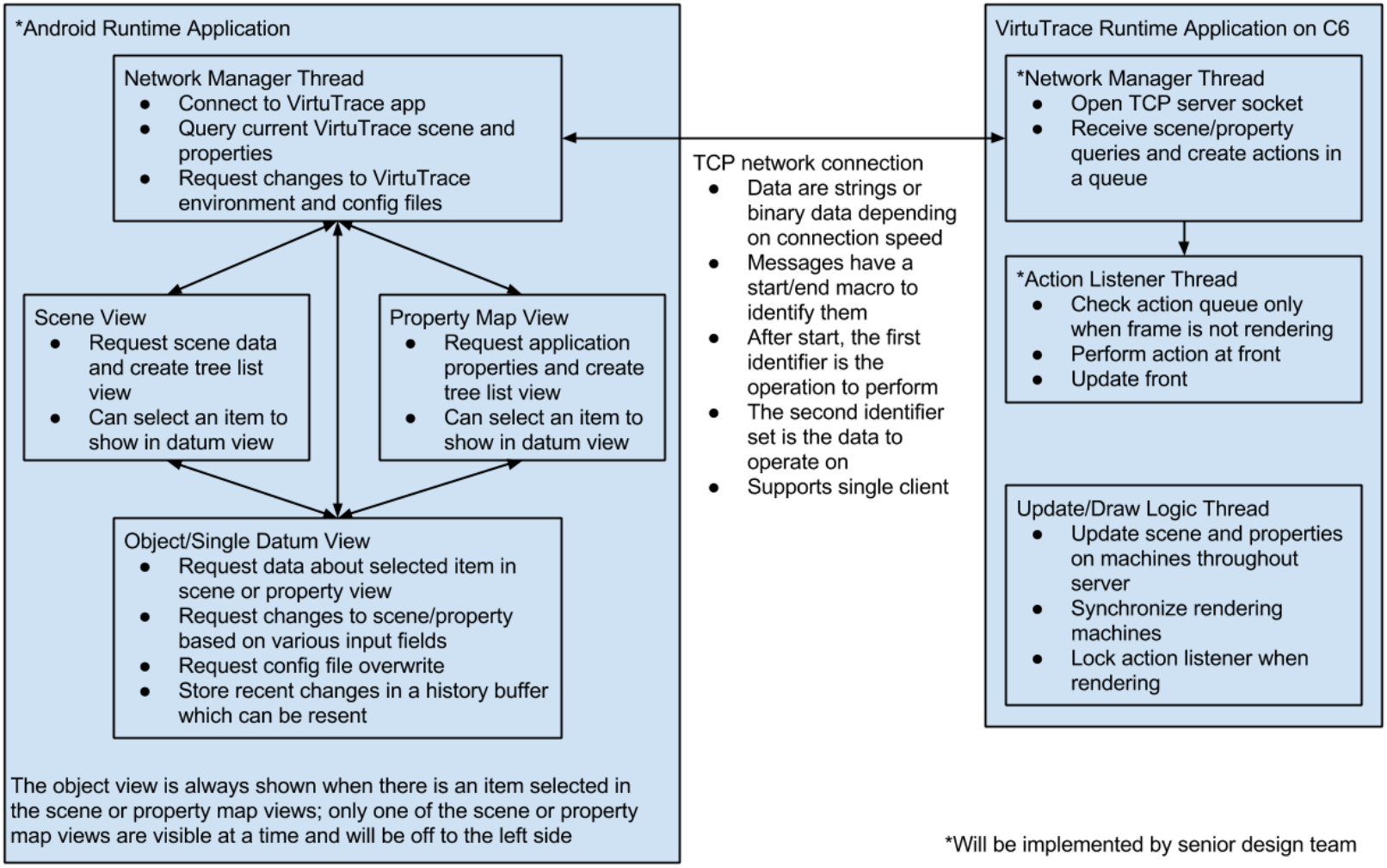


Figure 1B

While the specific app content is subject to change, the general structure will likely remain throughout the project lifetime. The initial design of the app will be optimized for tablet use, but our plans do include development of a phone layout as well, if time permits.

System Block Diagram



*Will be implemented by senior design team

Operating Environment

VT Remote will be used in conjunction with the C6 and is primarily intended to provide real time changes to the system. Users of the application will be inside of or near the C6 while a simulation is in progress. Additionally, VT Remote will utilize the university's wireless network for communicating with the C6 and its cluster of controllers. The app will run on the Android OS, version 4.2 or higher, and is being designed for use with tablet ranging from 7" to 10.1" in screen size.

Functional Requirements

- VT Remote shall provide a dynamic list view of the scene graph.
- VT Remote shall provide a dynamic list view of the property map.
- VT Remote shall allow instantaneous switching between tabs.
- VT Remote shall provide dynamic detail views based on the contents of the property being edited or examined.
- VT Remote shall provide the ability to "watch" certain objects and view their properties at any time during the simulation (provided they exist in the given scope), regardless whether they're from the scene graph or property map.
- VirtuTrace shall only send the information of serializable objects to VT Remote.
- VT Remote shall only send the changes to be applied to the simulation to the C6.
- VT Remote shall provide the ability to capture and save a snapshot of the configuration of the current scene running in VirtuTrace to a network file location.
- Both VT Remote and VirtuTrace shall create a separate thread for managing and processing network communication.
- Network communication between VT Remote and VirtuTrace shall be asynchronous.
- VT Remote shall alert the user and allow them to reconnect to VirtuTrace in the event that the connection is lost.

Nonfunctional Requirements

- VT Remote shall always remain responsive to user input, i.e. the UI thread may not be blocked for a noticeable amount of time.
- All dialog boxes in VT Remote shall be modal.
- A single message between VT Remote and VirtuTrace shall not exceed 1KB.
- VT Remote shall not require more than 64MB of memory at any point during its lifetime.
- VT Remote shall not crash due to any form of user input.
- VT Remote's installation shall not exceed a size of 20MB.
- VT Remote shall not crash due to a loss of connection to VirtuTrace.

Market Research

Since the C6 and VirtuTrace is a rather unique set of equipment and software, there are not many setups like it. Therefore, there currently are not many programs or applications for us to compare with. The most closely related application areas are in tablet and other mobile devices where the device is used to interface with other common commercial products.

The main idea is to offer the functionality of an existing application on a more mobile platform, and even possibly enhance the functionality of the application through a mobile application layer. Since no real applications exist that are comparable to the services we aim to provide, the UI design will be mostly driven by the customer.

Deliverables

Our main deliverable will the Android application that will run on an Android tablet. Secondary deliverables will be various documentation on how to use the app from within the C6 and also information about the implementation for the app so future developers can easily understand how to maintain and extend it. Additionally, we are providing some minor additions to the VirtuTrace code base.

The plan is to deliver a prototype application that can communicate with VirtuTrace and perform run-time modifications by the end of the semester. For the spring semester, we will then produce incremental reiterations, polishing the UI and adding custom views for specific data types.

Work Breakdown

The current work breakdown is as follows:

Sheil Patel
- Understanding of VirtuTrace basics.
- Android GUI design and implementation.
- Network connection using TCP protocol within VT Remote.
- Documentation and presentation preparation as needed.

Alexander Maxwell
- Understanding of VirtuTrace basics.
- Android GUI design and implementation.
- Documentation and presentation preparation as needed.

Kollin Burns
- Understanding of VirtuTrace basics.
- Android backend implementation and optimization.
- Network connection using TCP protocol within VirtuTrace.
- Documentation and presentation preparation as needed.

Tanner Borglum
- Understanding of VirtuTrace basics.
- Serialization and deserialization for VirtuTrace.
- VirtuTrace OpenSceneGraph wrapper library for Java.
- Documentation and presentation preparation as needed.

Lukas Herrmann
- Understanding of VirtuTrace basics.
- Serialization and deserialization for the Android application.
- Real time updates to C6 display after changes are made in VirtuTrace.
- Documentation and presentation preparation as needed.

Resource Requirements

This project will require is an Android device, preferably a tablet with at least a 7.0” screen and density of at least 200 pixels per inch, running Android 4.2 or above and access to a Linux OS running an instance of VirtuTrace. Additionally, a wireless network will be required for communication. Finally, since the device is meant to assist testing within the C6, time for testing within the C6 simulator environment will be a valuable resource as well.

Project Schedule

Week of 10/13
- Design and begin to implement basics Android application interface.
- Design test code for round tripping data on the VirtuTrace side.
- Draft design document 1.

Week of 10/20
- Have functional code for interpreting a property map and/or scene graph.
- Begin design of network code in parallel with serialization code.
- Finalize design document 1.

Week of 10/27
- Have Android application able to receive data across the network from VirtuTrace.
- Test run-time modification of VirtuTrace.

Week of 11/3
- Design group presentation.
- Have deserialization available for VirtuTrace.
- Have serialization available for the application.

Week of 11/10
- Draft group presentation.
- Have full serializability and deserializability for property map.

Week of 11/17
- Finalize group presentation.
- Have full serializability and deserializability for scene graph.

* Thanksgiving Break *

Week of 12/3
- Finalize design document.
- Finalize project plan.
- Round out any reasonable parts in VirtuTrace or the Android application.
- Have basic demo ready

Week of 12/10

- Give group presentation.

* Winter Break *

Week of 1/19/14

- Meet with advisor to discuss new schedule for semester and other administrative details.

Week of 1/26

- Discuss plan for fixing or completing broken and/or missing code from previous semester.

Week of 2/2

- Design feature map for the debugging tool.

Week of 2/9

- Determine and design any changes needed for the Android interface to implement the debugger feature(s).

Week of 2/16

- Implement changes to Android interface and any back end changes for data transfer.

Week of 2/23

- Add code to implement the debugger from VirtuTrace.

Week of 3/2

- Test debugging code manually and in the C6.

Week of3/9

- Discuss widgets to be added to the interface.

* Spring Break *

Week of 3/23

- Pair or design widgets as needed for specific actions correlating to the scene graph.

Week of 3/30

- Pair or design widgets as need for specific actions correlating to the property map.

Week of 4/6

- Implement code to back up the widgets in the Android application.

Week of 4/13

- Discuss features that could be reasonably added in the remaining time.

Week of 4/20 to END

- Work on adding feasible features and polish as well as cleaning up code for future extensions.

Potential Risks

Risks	Mitigations
Project goes over schedule	We feel that our projected schedule isn't conservative by any means, but we have considered our potential weaknesses and included them in our estimated timeline.
Team members do not complete individual tasks on time	Constant communication and a consistent schedule will keep everyone on task and assigning specific responsibilities to team members will keep them accountable for their share.
VirtuTrace modifications are too complex	We have already discovered some unknown issues within the VirtuTrace framework that have affected our work. Finding and addressing them early and swiftly will be key in preventing them from becoming significantly detrimental.
Android app sends data that is too large for network to handle	This is something we are currently researching. The amount and frequency of data transfer will affect our network interface. It will need to be tested and designed to accommodate iterations.
Issues with the C6's multithreaded architecture and communication with VTRemote	Sending data over a wireless network introduces a certain amount of uncertainty. We are designing our networking communication to perform asynchronously but intelligently to minimize the potential for conflicts.