Brianna Heersink
Brian Smith
Alex Warren
ECE 573
4 April, 2014

Requirements Verification

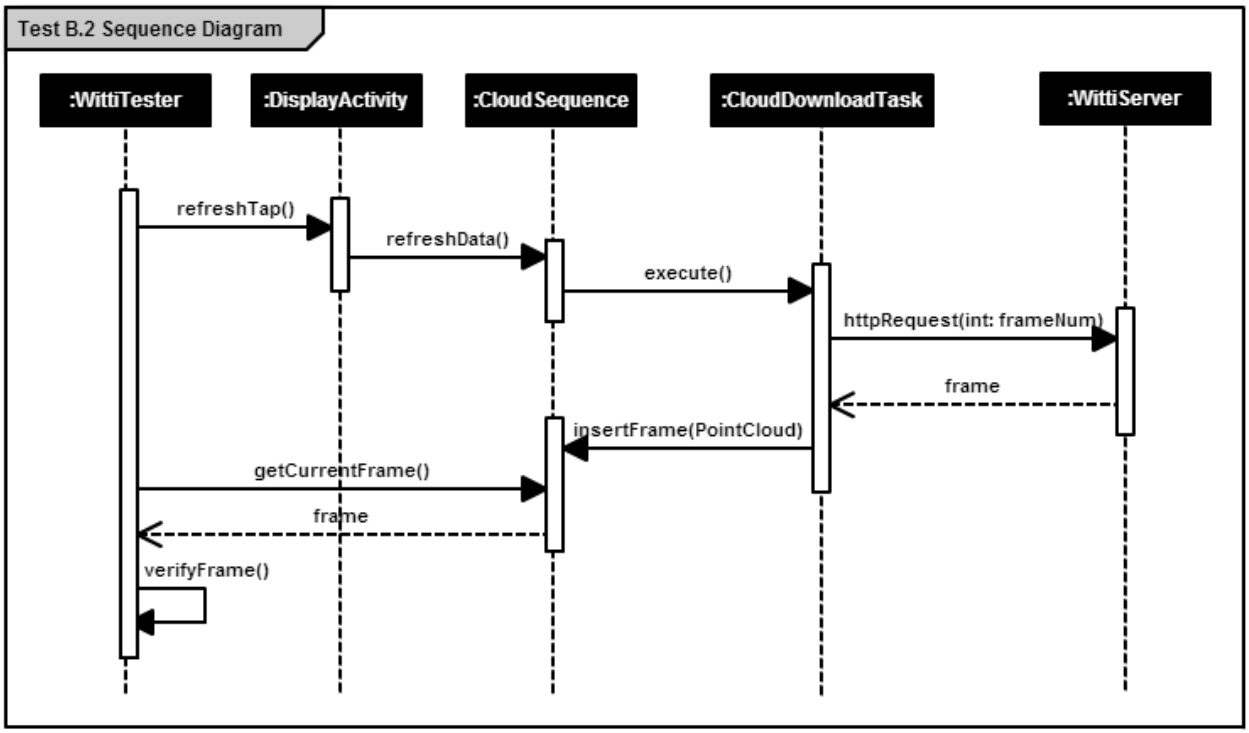
'B' Requirements Verification

B.1 Display Data Requirement: “The phone application software shall load and display Velodyne LiDAR data on the phone.”
Validation: Load a point cloud from a file in the app and check that a buffer of coordinates is created that matches expected data. The point cloud file will be generated procedurally e.g. ((1, 2, 3), (4, 5, 6))
Test:

Initialize application
Launch display activity
Load stored point cloud file
Check that the buffer contains correct number of values
Check that the buffer contains expected values
Check that the PointCloud class has correct metadata

B.2 Manual Data Refresh Requirement: “The phone application software shall be capable of refreshing the displayed Velodyne LiDAR data manually based on user input.”
Validation: Load a dummy point cloud frame on the server. Run Test B.2. If the correct frame is loaded in the CloudSequence, the requirement is met.
Test: The WittiTester sends a touch event to the DisplayActivity of the Witti application, which triggers the application to refresh the data to be rendered. After a specified time delay, the WittiTester compares the current frame with the expected dummy point cloud frame that was loaded on the WittiServer. Reference Figure 1.

Figure 1

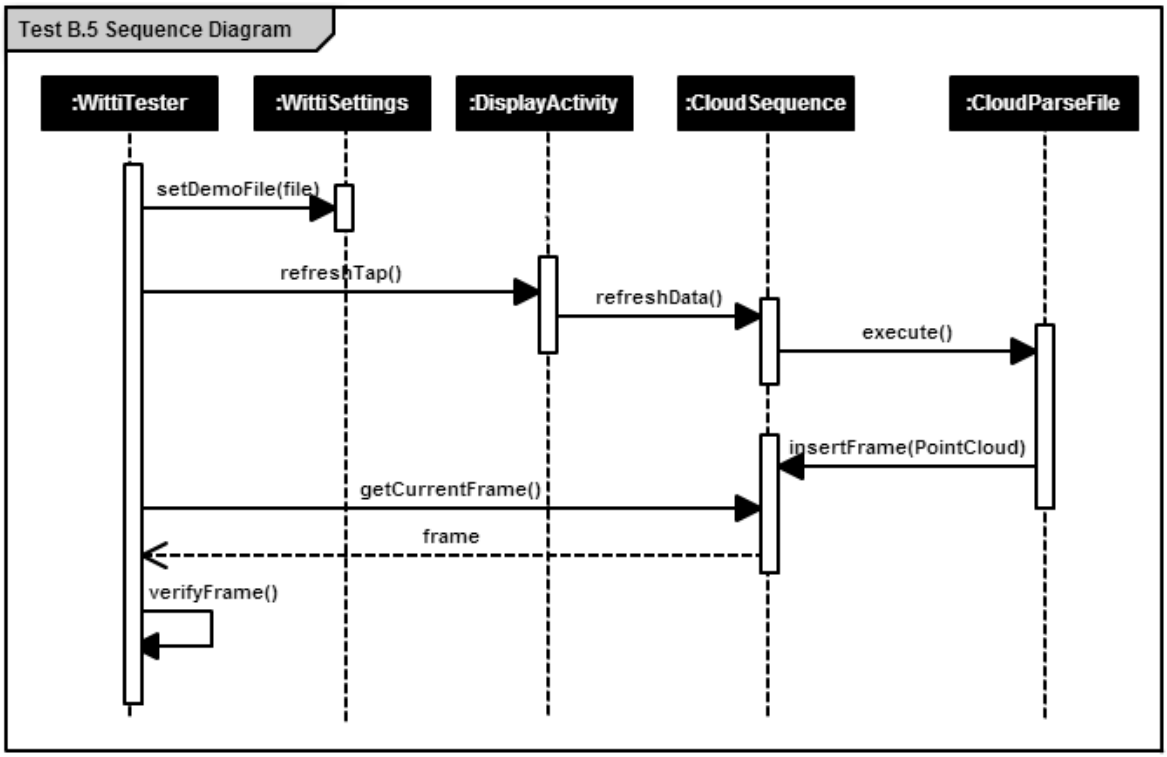


B.3 Server Download Requirement: "The phone application software shall be capable of downloading Velodyne LiDAR data in the form of an XYZ-point binary file from a remote server."
Validation: Load a dummy point cloud frame on the server. Run Test B.2. If the frame downloaded and loaded into the CloudSequence matches the dummy frame previously loaded on the server, the requirement is met.
Test: The WittiTester executes Test B.2. Reference the sequence diagram in Figure 1.

B.4 Server Upload Requirement: "The host computer software shall be capable of uploading or directly serving Velodyne LiDAR data in the form of an XYZ-point binary file."
Validation: Load a dummy point cloud frame on the server. The automated loading from the software on the host computer to the server is not required unless requirement A.6 is completed. Run Test B.2. If the frame downloaded and loaded into the CloudSequence matches the dummy frame previously loaded on the server, the requirement is met.
Test: The WittiTester executes Test B.2. Reference the sequence diagram in Figure 1.

B.5 Demo Playback Requirement: "The phone application shall be able to use XYZ-point text files stored locally for Manual Data Refresh mode."
Validation: Load a dummy point cloud frame on the server. Run Test B.5. If the correct frame is loaded in the CloudSequence, the requirement is met.
Test: The WittitTester configures the demo file setting in setUp(). In runTest(), the WittiTester sends a touch event to the DisplayActivity of the Witti application, which triggers the application to refresh the data to be rendered. After a specified time delay, the WittiTester compares the current frame with the expected dummy point cloud frame that was loaded on the WittiServer. Reference Figure 2.

Figure 2



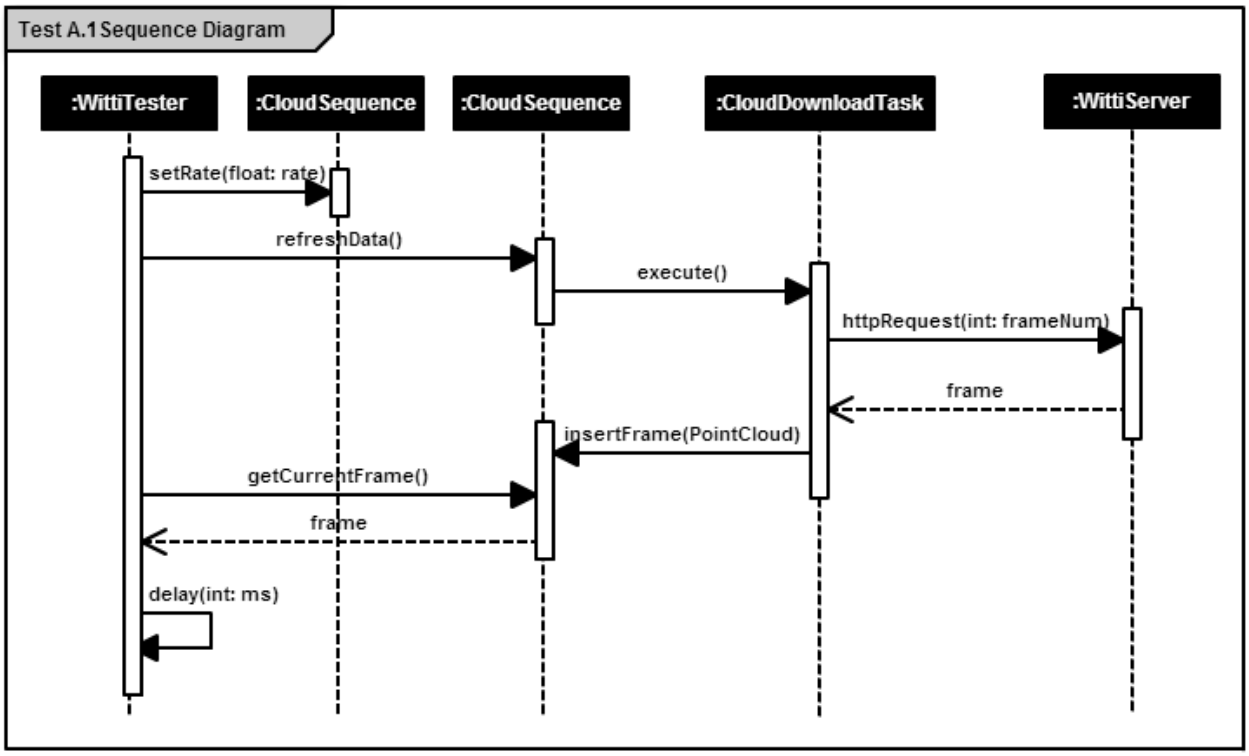
'A' Requirements Verification

A.1 Automatic Data Refresh Requirement: "The phone application software shall be capable of refreshing the displayed Velodyne LiDAR data automatically through a set refresh rate."

Validation: Load multiple dummy point cloud frames on the server. Run Test A.1 IIf WittiTest receives back a frame with the correct timestamp each time, the requirement is met. Alternatively, each point cloud frame can be verified by comparing it to the frame on the server.

Test: Run Test A.1 according to the sequence diagram in Figure 3. After the delay, which aligns with the set data rate for the app, the WittiTester will requests another data refresh. Although not shown in the sequence diagram, this will be repeated multiple times.

Figure 3



A.2 Draw Trajectory Requirement: "The phone application software shall read a trajectory selected by the user through touch events on the phone."
Validation: Verify that the array of points collected by the app from the touch events match the array of test touch event inputs, i.e. the coordinates, order and amount are the same for both arrays.
Test:

Initialize application
Launch display activity
Launch trajectory input mode
Initialize the touch event array (TEA) used for testing
While there are still TEA elements remaining
    Send the current TEA element
Verify the app's trajectory array matches the TEA

A.3 Trajectory Clearance Requirement: "The phone application software shall indicate if the user selected trajectories would be too close to obstacles."
Validation: A total of five test cases will be used to check for successful valid path condition and five test cases will be used to check successful invalid path condition. The paths will be created manually using one frame of real LiDar data to ensure that the valid test cases only contain valid paths and the same for the invalid cases.
Test:

Initialize application
Launch display activity
Initialize the test trajectory array (TTA) for each valid and invalid test cases
While there are more valid test cases

Launch trajectory input mode
While there are more elements in current valid test case's TTA
Send the current TTA element
Verify and log an invalid or valid trajectory response
While there are more invalid test cases
Launch trajectory input mode
While there are more elements in current invalid test case's TTA
Send the current TTA element
Verify and log an invalid or valid trajectory response
Verify correct trajectory response for each test case

A.4 Trajectory Drivability Requirement: "The phone application software shall reject trajectories that are not possible to follow due to vehicle limitations."
Validation: A total of five test cases will be used to check for successful valid trajectory cases and five test cases will be used to check successful invalid trajectory cases. Valid and invalid trajectories will be created manually. A valid path will contain no curve with a radius below the vehicle's threshold and an invalid path will contain at least one curve with a radius below the vehicle's threshold.
Test:

Initialize application
Launch display activity
Initialize the test trajectory array (TTA) for each valid and invalid test cases
While there are more valid test cases
Launch trajectory input mode
While there are more elements in current valid test case's TTA
Send the current TTA element
Verify and log an invalid or valid trajectory response
While there are more invalid test cases
Launch trajectory input mode
While there are more elements in current invalid test case's TTA
Send the current TTA element
Verify and log an invalid or valid trajectory response
Verify correct trajectory response for each test case

A.5 Perspective Change Requirement: "The phone application software shall be capable of changing the perspective of the displayed data on the phone."
Validation: A successful test will consist of capturing the appropriate user touch event that indicates a change in perspective and then verifying the camera matrix was updated. The camera matrix will be updated appropriately according to how long the touch event was, i.e. the distance between the touch down event and the touch up event.
Test:

Initialize application
Launch display activity
Initialize the touch event array (TEA) simulating a perspective change
Save the initial camera matrix
While there are still TEA elements remaining
Send the current TEA element
Verify the camera matrix has been updated correctly corresponding to the TEA used to test

A.6 Dynamic LiDAR Conversion Requirement: “The host computer shall be capable of processing and converting the PCAP files to phone readable data as they are received within a bounded delay that will be established.”
Validation: Given that test A.1 successfully runs, instead of running using preloaded files, the LiDAR will be attached to the host computer and the capture software will be capable of saving the files to a web accessible directory, or of serving the files directly.
Test: Running the test as shown in figure 3 for requirement A.1 but with the further check, that the files loaded from the server have a timestamp indicating that they were recorded within a set time interval.

A.7 JAUS Compatible Requirement: “The host computer software shall be capable of sending JAUS messages containing converted LiDAR data.”
Validation: A JAUS component will be running on the host computer that can send LiDAR data. The phone shall receive data that it is capable of parsing that has originated in a JAUS component.
Test: To test that the data is successfully transferred, metadata will be included that indicates that the data originated in the JAUS component. The phone will request a frame as in figure 1, and this frame will be tested for the correct metadata and parseable points.