ImagesTo3DModels - Requirements Verification

Requirement 1.1: “TSS will allow the user to take pictures with the smartphone camera.”
Validation 1.1: Run Test 1.1
Test 1.1: Capture a picture by following sequence model 1.1 and inspect result by looking for pictures by the phone’s external storage directory (Images_To_3D_Models/modelName/images).

Requirement 1.2: “TSS save multiple images as internal data structures.”
Validation 1.2:Run Test 1.2:
Test 1.2: run unit test testForMultipleImages() in the ImageCaptureActivityTest.java class. This android unit test tests to see if multiple .jpg images are read into the Object3DModel internal data structure.

Requirement 1.3: “TSS automatically differentiate a user selected object in an image from the background of that image.”
Validation 1.3: Run Test 1.3
Test 1.3: Create a model by following sequence model 1.3 and inspect result by looking for the no background picture in the phone’s external storage directory (Images_To_3D_Models/test13/noBackground.jpg).

Requirement 1.4: “TSS automatically detect outer edges of a user selected object.”
Validation 1.4: Run Test 1.4
Test 1.4: Run unit test testOuterEdges() in the ImageCaptureActivityTest.java class. This creates a model following sequence model 1.2 and ensures that the edge maps for one image contain valid vertices.

Requirement 1.5: “TSS match similar features from multiple images taken of the same object from different known positions.”
Validation 1.5: Run Test 1.5
Test 1.5: Create a model by following sequence model 1.5 and inspect result by looking at the .obj or .ply file saved in the phone’s external storage directory (Images_To_3D_Models/test15/test15.obj). Ensure the edges of the rectangles match up as in the Figure 1.5.

Requirement 1.6: “TSS convert images into a collection of vertices with xyz coordinates.”
Validation 1.6: Run Test 1.6

Test 1.6: Run unit test testVerticesAndFaces() in the ImageCaptureActivityTest.java class. This creates a model following sequence model 1.2 and ensures that the vertexArray for one image contains valid vertices.

Requirement 1.7: “TSS combine vertices into triangulated faces.”
Validation 1.7: Run Test 1.7
Test 1.7: Run unit test testVerticesAndFaces() in the ImageCaptureActivityTest.java class. This creates a model following sequence model 1.2 and ensures that the triangleFaceArray for one image contains valid faces.

Requirement 1.8: “TSS save output to the external storage directory on the device.”
Validation 1.8: Run Test 1.8
Test 1.8: Run unit test testFileOutput_OBJ() in the ImageCaptureActivityTest.java class. This creates a model following sequence model 1.3 and ensures that the obj file exists and has data.

Requirement 1.9: " TSS allow sharing of output files with other devices (e.g. email, cloud storage, etc.) "
Validation 1.9: Run Test 1.9
Test 1.9: Executes unit tests on the intent.action_send start activity function and tests to see if the output file was attached. If intent.action_send prompt is begun then we know that the user has the capability of then choosing a specific medium for sharing output files.

Requirement 1.10: “TSS allow users to rotate and view 3D models from different angles.”
Validation 1.10: Run Test 1.10
Test 1.10: Open .obj file in model viewer by following the steps in sequence model 1.10 (not implemented).

Requirement 1.11: "On shutdown of device or app, TSS compress current object files."
Validation 1.11: Run Test 1.11
Test 1.11: Following sequence model 1.11 execute unit test on the compressFile() function in the Object3DModel class of file by compressing the file and then uncompress it and comparing the the uncompressed file with the original file (not implemented).

Requirement 1.12: “TSS match similar features from multiple images taken of the same object from different unknown positions.”

Validation 1.12: Run Test 1.12
Test 1.12: Create a model by following sequence model 1.5 and inspect result by looking at the .obj or .ply file saved in the phone's external storage directory (Images_To_3D_Models/test112/test112.obj). Inspect results (not implemented).

Requirement 2.1: "TSS be compatible with Andoird API 14 (Ice Cream Sandwich) to ensure compatibility with at least 85% of Android devices."
Validation 2.1: Run Test 2.1
Test 2.1: Inspect Android Manifest for "minSdkVersion". If the API version is 14 then the test passes.

Requirement 2.2: "TSS requires 6 images of an object from different angles before 3D conversion." (Requirement Modified)
Validation 2.2: Run Test 2.2
Test 2.2: testNumberOfImages() in ModelPhotoGalleryActivityTest.java class. This will execute a unit test to check If 6 images are present before allowing 3D conversion, which follows the 2.2 sequence model.

Requirement 2.3: "TSS output and save .obj files to the external storage directory."
Validation 2.3: Run Test 1.8
Test 1.8: Run unit test testFileOutput_OBJ() in the ImageCaptureActivityTest.java class. This creates a model following sequence model 1.3 and ensures that the obj file exists and has data.

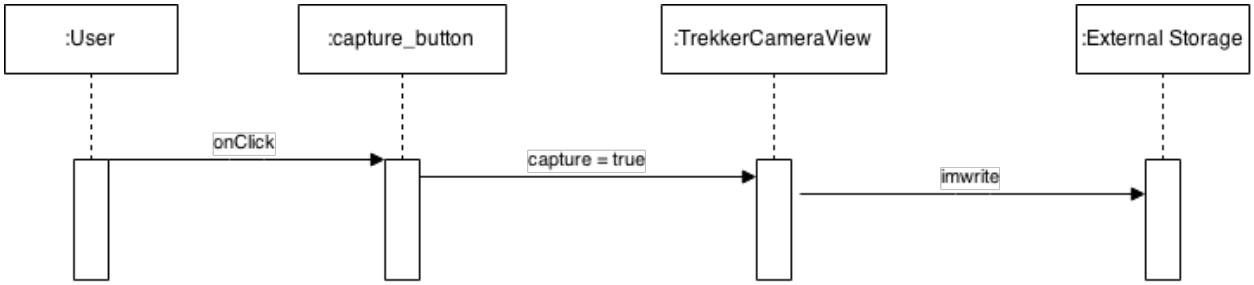
Requirement 2.4: "TSS output and save .ply files to the external storage directory."
Validation 2.4: Run Test 2.4
Test 2.4: Run unit test testFileOutput_PLY() in the ImageCaptureActivityTest.java class. This creates a model following sequence model 1.3 and ensures that the ply file exists and has data.

Requirement 2.5: "TSS import .obj files and render them as 3D models."
Validation 2.5: Run Test 1.10
Test 1.10: Open .obj file in model viewer by following the steps in sequence model 1.10 (not implemented).

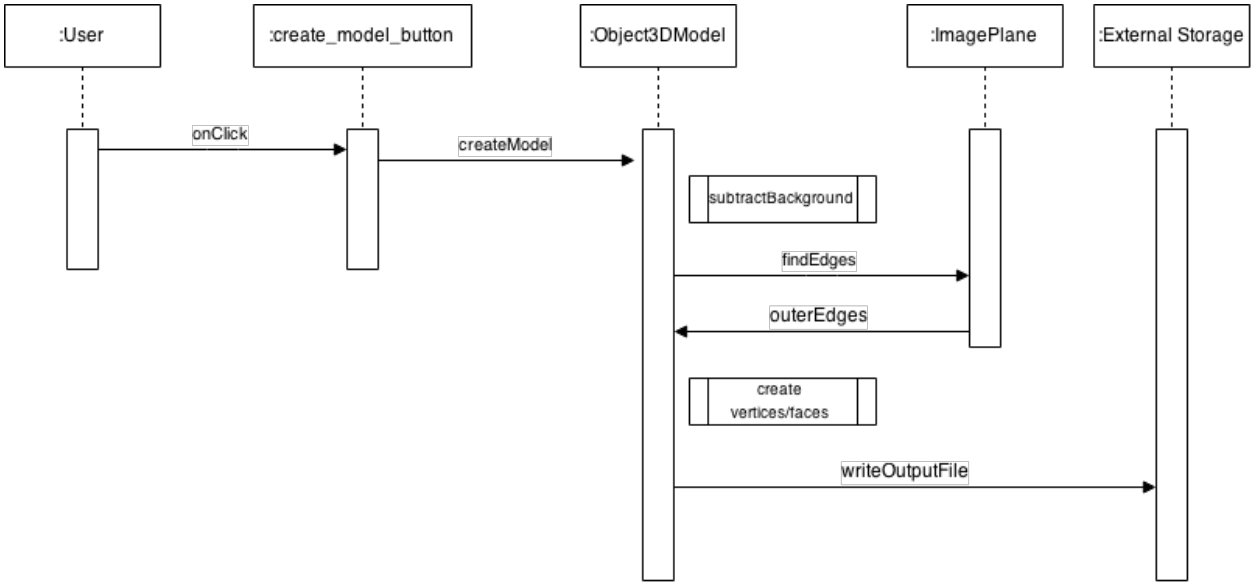
Requirement 2.6: "TSS will give dimensional ratios (height, width, and length) within a to be determined accuracy."
Validation 2.6: Run Test 2.6

Test 2.6: Create a 3D model of an object with known dimensions by following sequence model 2.6. Open the 3D model in a model viewing software such as meshLab. Use the measuring tool to measure each side of the model and ensure the ratios are approximately the same.

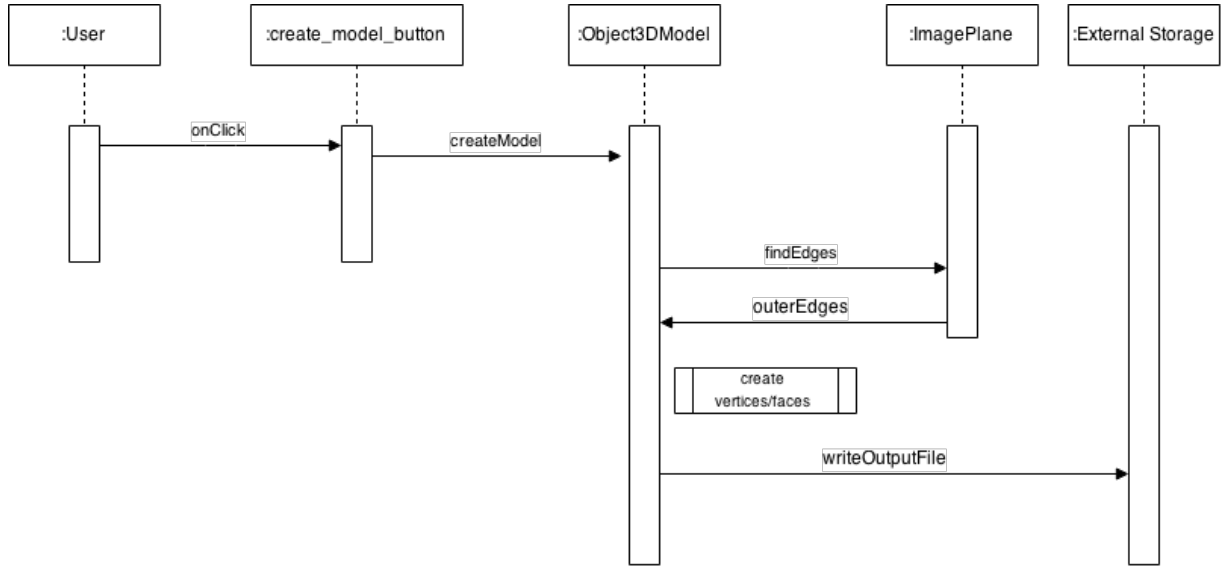
Sequence Models



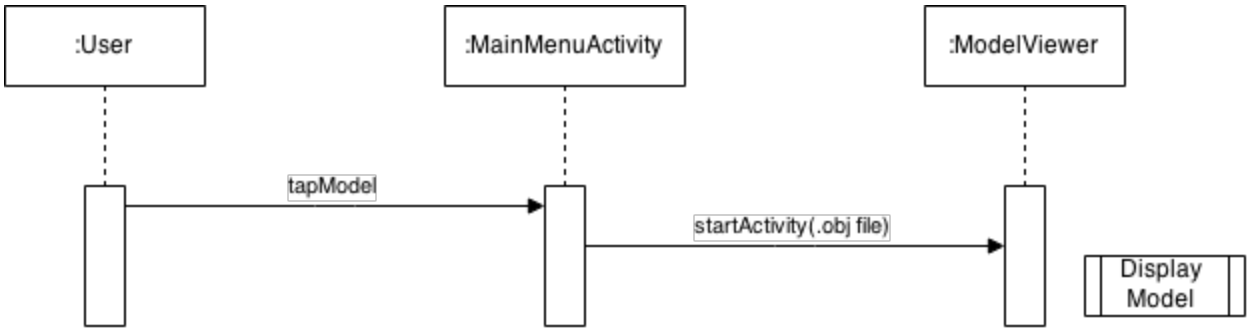
Sequence Model 1.1 - Taking Pictures Sequence



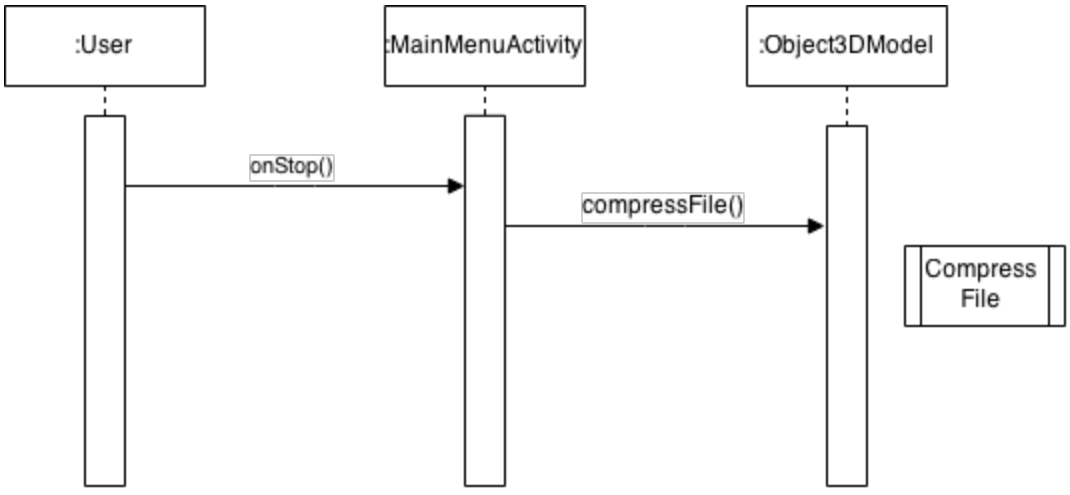
Sequence Model 1.3 - Model Creation Sequence



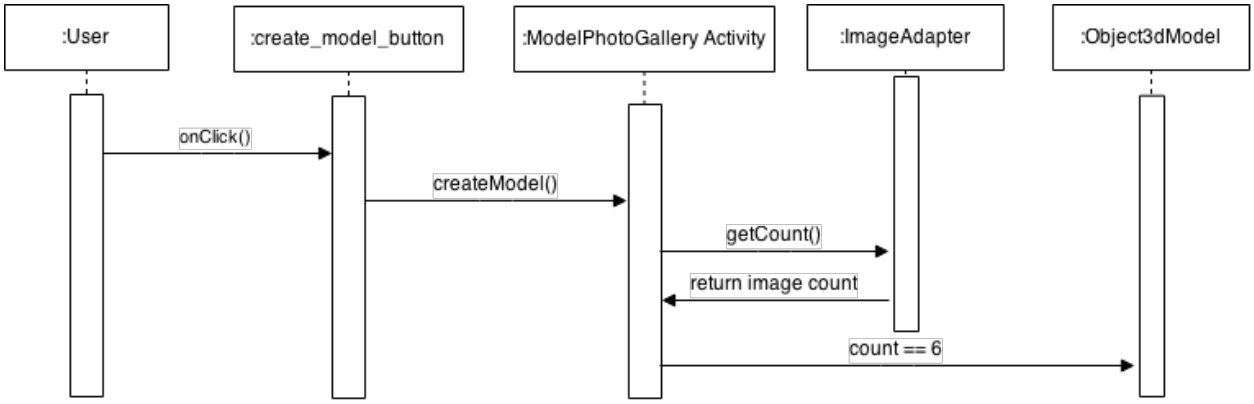
Sequence Model 1.5 - Edge Finding Sequence



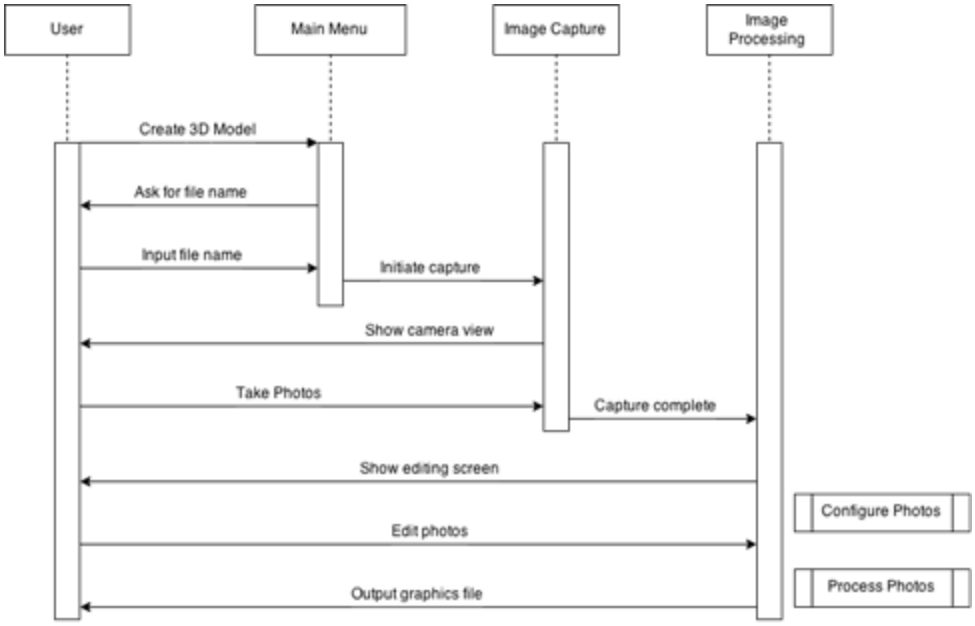
Sequence Model 1.10 - Model Viewing Sequence



Sequence Model 1.11 - File Compression Sequence



Sequence Model 2.2 - Check For Required Number of Images Sequence



Sequence Model 2.6 - Overall Application Sequence

Figures

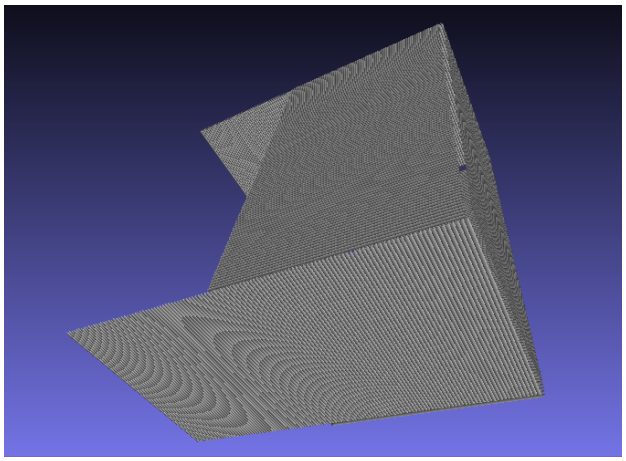


Figure 1.5 - 3D model example