Team Trekker (trkr)

Wesley Folz

Ryan Hoefferle

ECE 473 Project

Spring 2015

Contents

1. Front Matter 3
1.1 Table of Figures 3
1.2 Table of Tables 3
1.3 Executive Summary 3
1.4 Project Overview 3
2. Analysis and Models 4
2.1 Requirements 4
2.2 Application Analysis 5
2.3 Domain Analysis 6
2.4 Important Algorithms: 8
3. Design and Test 9
3.1 Class Design 9
3.2 Testing Strategy 9
3.3 Integration with Platform 10
4. Implementation Plan 11
4.1 Task Allocation and Breakdown 11
4.2 Timeline for Completion 12
4.3 Tasks and Experience 12

1. Front Matter

1.1 Table of Figures

Figure 1: Main Menu Screen Figure 2: Image Capture Screen Figure 3: Image Gallery Screen 5
Figure 6: Use Case Diagram 6
Figure 4: Help Screen Figure 5: Model View Screen 6
Figure 7: GUI State Diagram 6
Figure 8: Sequence Diagram 7
Figure 9: Application Activity Diagram 7
Figure 10: Image to 3D Model Conversion Activity Diagram 8
Figure 11: UML Object Diagram 9
Figure 12: Gantt Chart Timeline 12

1.2 Table of Tables

Table 1: Functional Requirements 4
Table 2: Non-Functional Requirements 5
Table 3: Functional Requirement Testing 10
Table 4: Non-Functional Requirement Testing 10
Table 5: Functional Requirement Assignments 11
Table 6: Non-Functional Requirement Assignments 12

1.3 Executive Summary

The proposed app will create a digital 3D model of an object from multiple photos of an object captured by the camera on a smartphone. This 3D model would be a collection of vertices, edges and potentially faces. The model of the object could then be exported into a standard graphics file such as a ".obj" or ".stl" file. Optionally, the app could display the 3D model that was created and allow the user to move it around and view it from different angles. Our app would start out simple by attempting its function on something like Lego blocks and if successful attempt the 3D capture of more complex shapes.

1.4 Project Overview

Simulating the world around us can be a difficult task; sometimes it requires creating accurate models of physical objects in the real world. Simulations are also much more interesting and easy for humans to understand and enjoy when a visual representation of the simulation is provided. This application hopes to aid in both of these problems with the conversion of simple two dimensional images of physical objects into three dimensional models. The overall goal of this application is to allow users to easily create digital models which accurately capture the physical reality of real world objects. By outputting these three dimensional models into standard graphics file formats, they can be used in all sorts of physical simulations for the purpose of scientific testing or entertainment. Delivering a mobile application that is capable of creating three dimensional models, takes advantage of the convenience of the camera and mobile computing power available in most smartphones.

2. Analysis and Models

2.1 Requirements

Functional Requirements				
Req#	TSS = The Software Shall	Difficulty 1/E - 5/H	Prerequisite Requirements	List
1.1	TSS will allow the user to take pictures with the smartphone camera.	1	n/a	B
1.2	TSS save multiple images as internal data structures.	1	n/a	B
1.3	TSS automatically differentiate a user selected object in an image from the background of that image.	2	1.2	A
1.4	TSS automatically detect outer edges of a user selected object.	2	1.2	A
1.5	TSS match similar features from multiple images taken of the same object from different angles.	3	1.1, 1.2	B
1.6	TSS convert images into a collection of vertices with xyz coordinates.	3	1.5	B
1.7	TSS combine vertices into triangulated faces.	2	1.6	B
1.8	TSS save output to the sd card on the device.	1	1.7	B
1.9	TSS share output files with other devices (e.g. email, cloud storage, etc.)	1	n/a	A
1.10	TSS allow users to rotate and view 3D models from different angles.	3	1.11	A
1.11	On shutdown of device or app, TSS compress current object files.	2	n/a	A

Table 1: Functional Requirements

Non-Functional Requirements				
Req#	TSS = The Software Shall	Difficulty 1/E - 5/H	Prerequisite Requirements	List
2.1	TSS be compatible with Andoird API 14 (Ice Cream Sandwich) to ensure compatibility with at least 85% of Android devices.	1	n/a	B
2.2	TSS require the capture of 2 or more images of an object from different angles before 3D conversion.	1	1.1	B
2.3	TSS output and save .obj files to the sd card.	2	1.7	B
2.4	TSS output and save .stl files to the sd card.	3	1.7	A
2.5	TSS import .obj files and render them as 3D models.	4	n/a	A
2.6	TSS will give dimensional ratios (height, width, and length) within an accuracy of a 10% POE (percent of Error)	2	n/a	B

Table 2: Non-Functional Requirements

2.2 Application Analysis

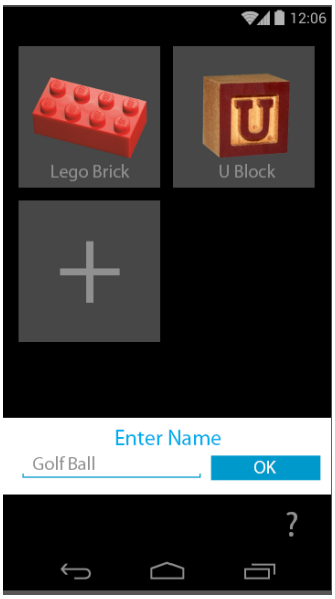


Figure 1: Main Menu Screen

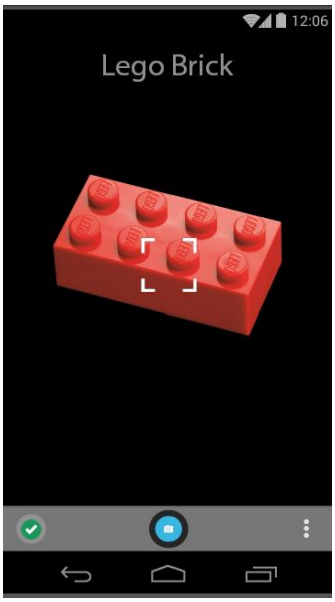


Figure 2: Image Capture Screen

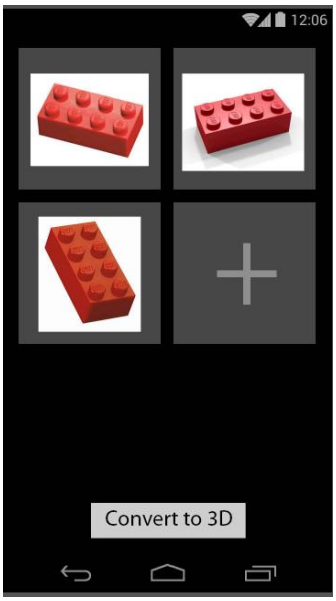


Figure 3: Image Gallery Screen

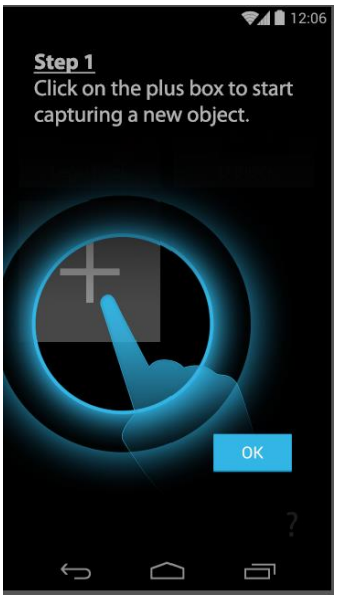


Figure 5: Help Screen

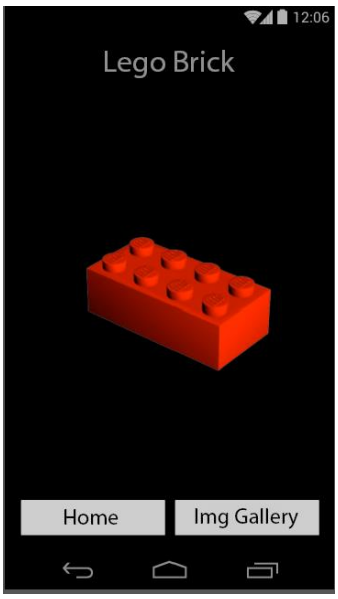


Figure 6: Model View Screen

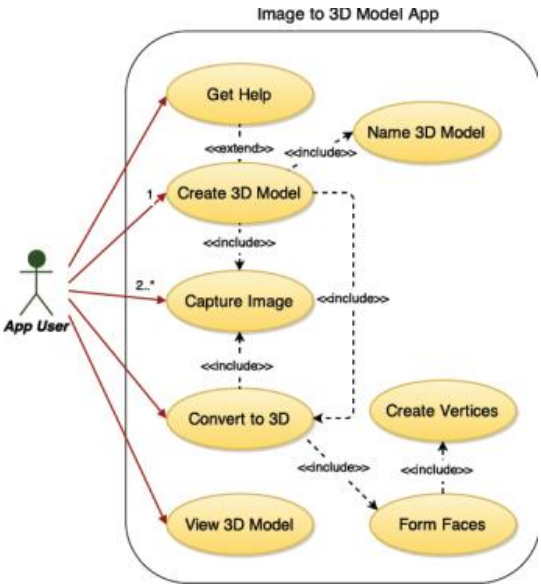


Figure 4: Use Case Diagram

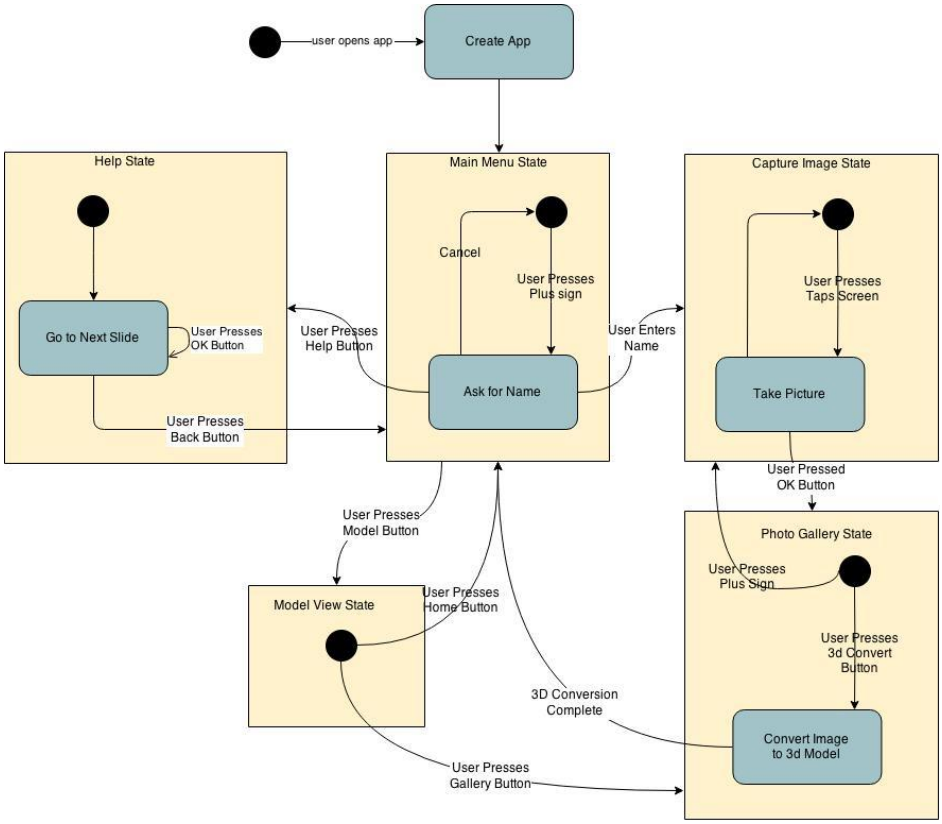


Figure 7: GUI State Diagram

2.3 Domain Analysis

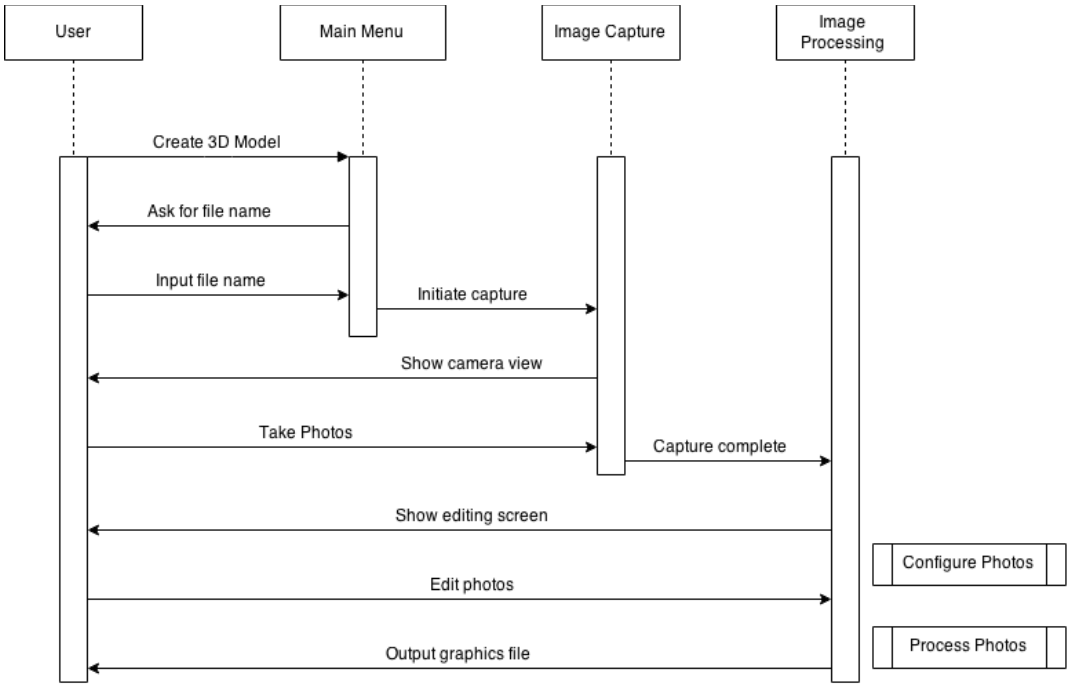


Figure 8: Sequence Diagram

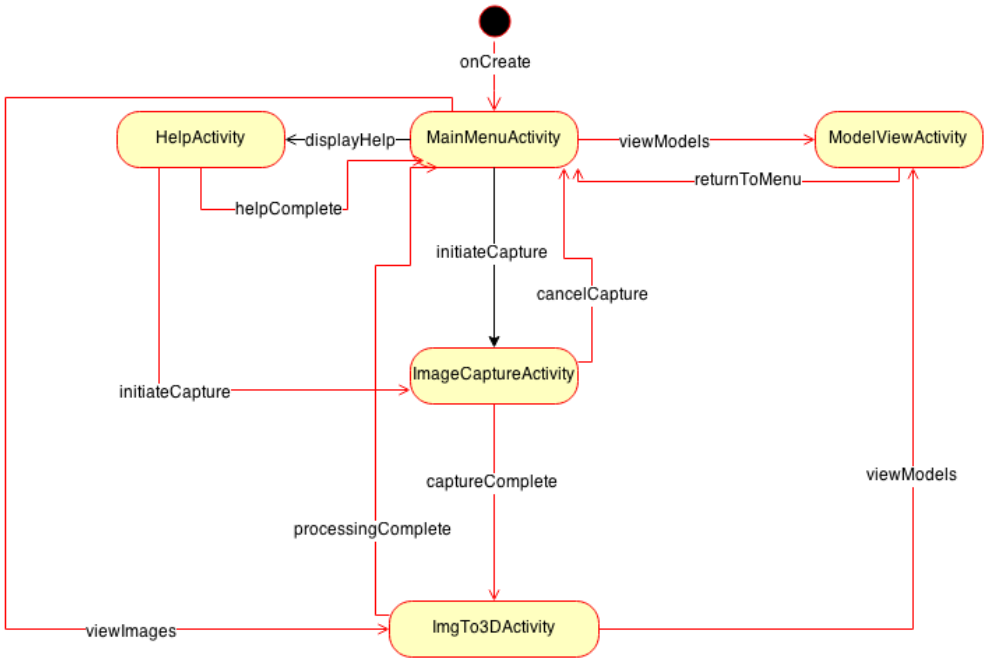


Figure 9: Application Activity Diagram

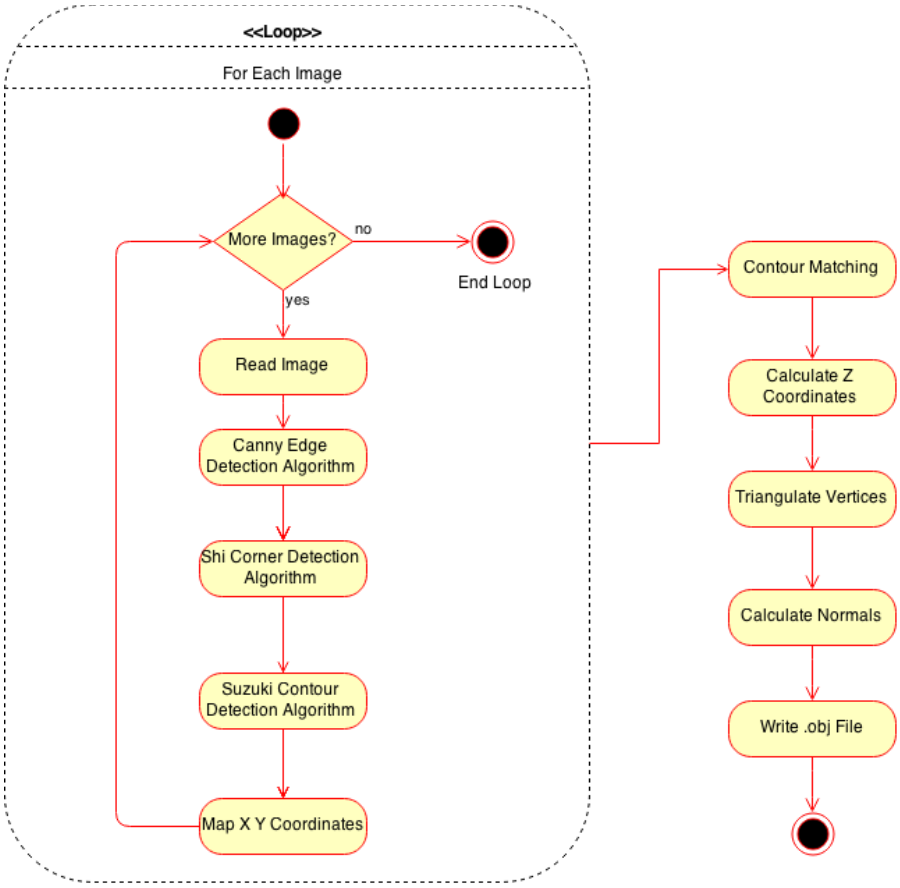


Figure 10: Image to 3D Model Conversion Activity Diagram

2.4 Important Algorithms:

- Canny86 algorithm: This algorithm will be used to detect the edges of an object in an image.
  J. Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

- Suzuki85 algorithm: This algorithm finds color contours which will help with object reconstruction.
  Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. *CVGIP 30 1, pp 32-46 (1985)*

- Shi94 algorithm: This algorithm detects strong corners of an object in an image.
  J. Shi and C. Tomasi. Good Features to Track. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, June 1994.

3. Design and Test

3.1 Class Design

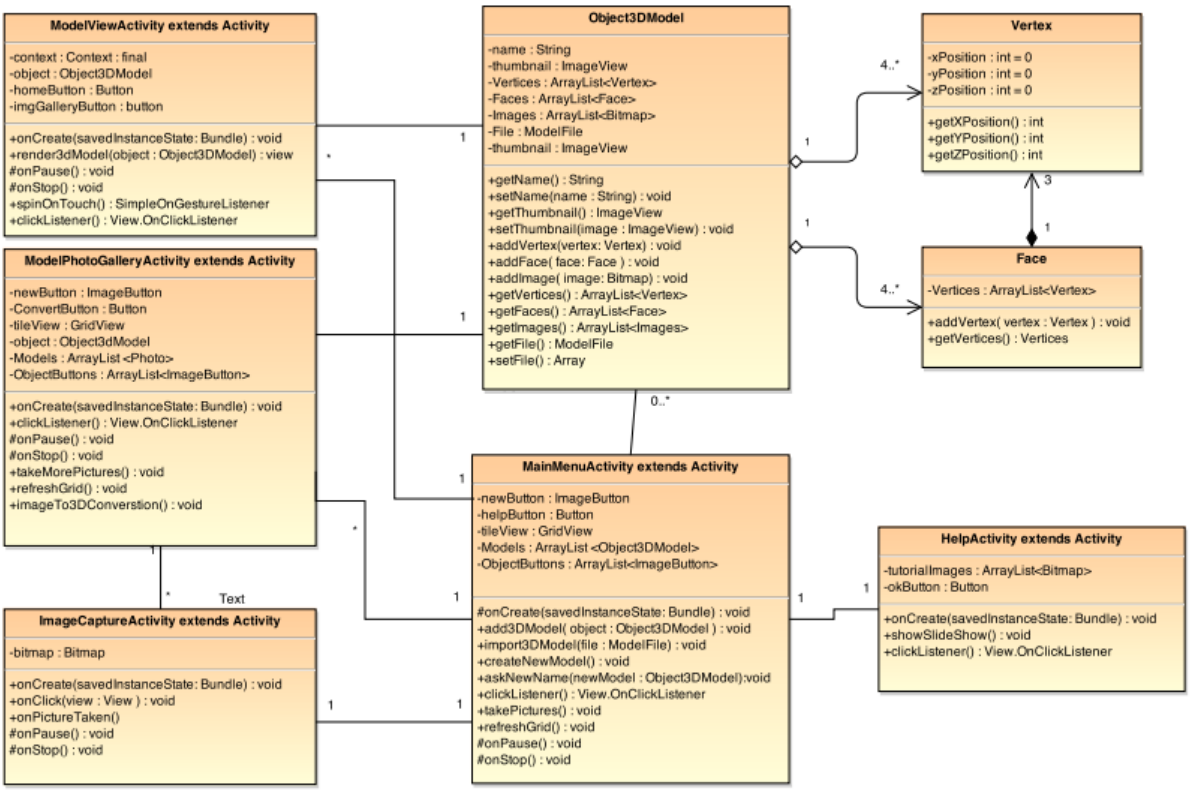


Figure 11: UML Object Diagram

3.2 Testing Strategy

Testing Functional Requirements		U = Unit Test M = Measure I = Inspect		
Req#	TSS = The Software Shall	U	M	I
1.1	TSS will allow the user to take pictures with the smartphone camera.			X
1.2	TSS save multiple images as internal data structures.	X		
1.3	TSS automatically differentiate a user selected object in an image from the background of that image.			X
1.4	TSS automatically detect outer edges of a user selected object.	X		
1.5	TSS match similar features from multiple images taken of the same			X

	object from different angles.			
1.6	TSS convert images into a collection of vertices with xyz coordinates.	X		
1.7	TSS combine vertices into triangulated faces.	X		
1.8	TSS save output to the sd card on the device.	X		
1.9	TSS allow sharing of output files with other devices (e.g. email, cloud storage, etc.)	X		
1.10	TSS allow users to rotate and view 3D models from different angles.			X
1.11	On shutdown of device or app, TSS compress current object files.	X		

Table 3: Functional Requirement Testing

Testing Non-Functional Requirements		U = Unit Test M = Measure I = Inspect		
Req#	TSS = The Software Shall	U	M	I
2.1	TSS be compatible with Andoird API 14 (Ice Cream Sandwich) to ensure compatibility with at least 85% of Android devices.			X
2.2	TSS require the capture of 2 or more images of an object from different angles before 3D conversion.	X		
2.3	TSS output and save .obj files to the sd card.	X		
2.4	TSS output and save .stl files to the sd card.	X		
2.5	TSS import .obj files and render them as 3D models.	X		X
2.6	TSS will give dimensional ratios (height, width, and length) within an accuracy of a 10% POE (percent of Error)		X	

Table 4: Non-Functional Requirement Testing

3.3 Integration with Platform

This application will require integration with the camera on an Android phone. The camera will take multiple pictures of an object that will eventually be converted into a 3D model. The CameraBridgeViewBase view, which is a class in the OpenCV library, will be used to display the camera image on the phone screen. When an image is captured with the camera, the image pixels will be internally stored in a Mat object, which is another class in the OpenCV library.

4. Implementation Plan

4.1 Task Allocation and Breakdown

Functional Requirement Assignments		
Req#	TSS = The Software Shall	Assigned To
1.1	TSS will allow the user to take pictures with the smartphone camera.	Ryan
1.2	TSS save multiple images as internal data structures.	Ryan
1.3	TSS automatically differentiate a user selected object in an image from the background of that image.	Wesley
1.4	TSS automatically detect outer edges of a user selected object.	Wesley
1.5	TSS match similar features from multiple images taken of the same object from different angles.	Wesley
1.6	TSS convert images into a collection of vertices with xyz coordinates.	Wesley
1.7	TSS combine vertices into triangulated faces.	Wesley
1.8	TSS save output to the sd card on the device.	Ryan
1.9	TSS allow sharing of output files with other devices (e.g. email, cloud storage, etc.)	Ryan
1.10	TSS allow users to rotate and view 3D models from different angles.	Wesley
1.11	On shutdown of device or app, TSS compress current object files.	Ryan

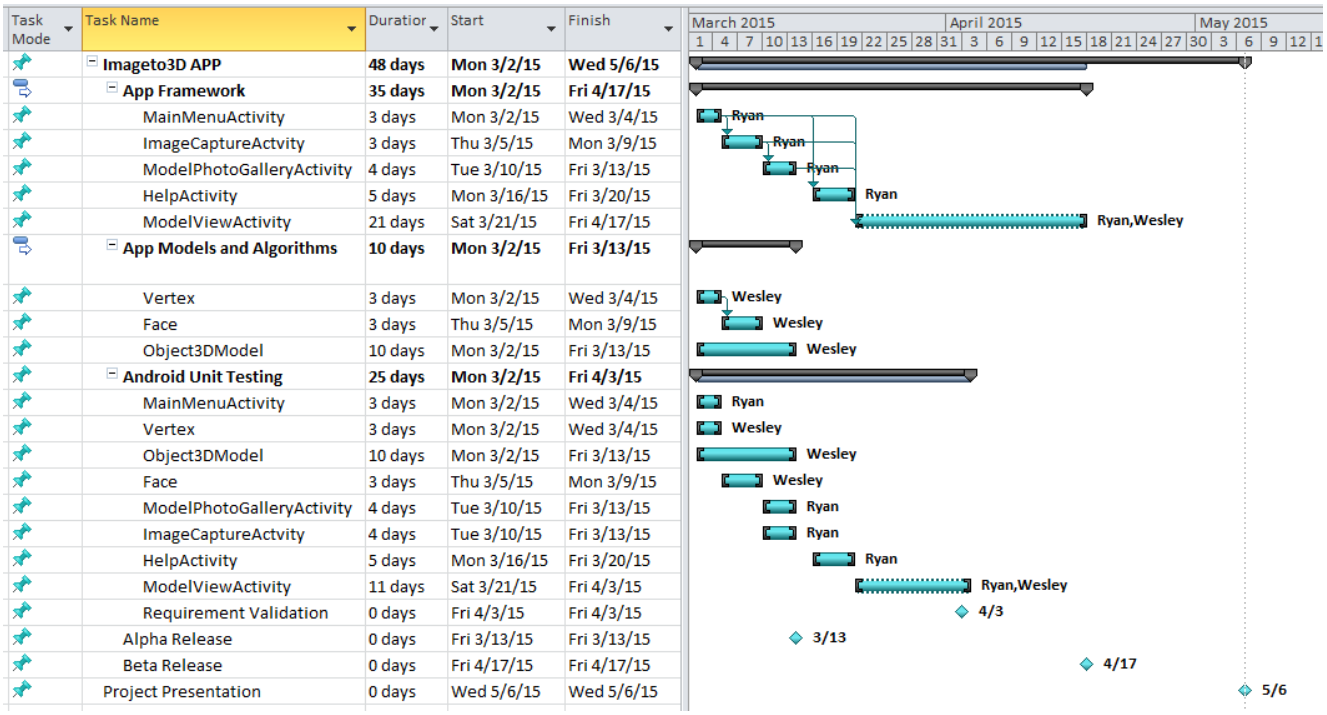
Table 5: Functional Requirement Assignments

Testing Non-Functional Requirements		
Req#	TSS = The Software Shall	Assigned To
2.1	TSS be compatible with Andoird API 14 (Ice Cream Sandwich) to ensure compatibility with at least 85% of Android devices.	Ryan
2.2	TSS require the capture of 2 or more images of an object from different angles before 3D conversion.	Ryan
2.3	TSS output and save .obj files to the sd card.	Ryan

2.4	TSS output and save .stl files to the sd card.	Ryan
2.5	TSS import .obj files and render them as 3D models.	Wesley
2.6	TSS will give dimensional ratios (height, width, and length) within an accuracy of a 10% POE (percent of Error)	Wesley

Table 6: Non-Functional Requirement Assignments

4.2 Timeline for Completion



Task Mode	Task Name	Duration	Start	Finish
	Imageto3D APP	**48 days**	**Mon 3/2/15**	**Wed 5/6/15**
	App Framework	**35 days**	**Mon 3/2/15**	**Fri 4/17/15**
	MainMenuActivity	3 days	Mon 3/2/15	Wed 3/4/15
	ImageCaptureActvity	3 days	Thu 3/5/15	Mon 3/9/15
	ModelPhotoGalleryActivity	4 days	Tue 3/10/15	Fri 3/13/15
	HelpActivity	5 days	Mon 3/16/15	Fri 3/20/15
	ModelViewActivity	21 days	Sat 3/21/15	Fri 4/17/15
	App Models and Algorithms	**10 days**	**Mon 3/2/15**	**Fri 3/13/15**
	Vertex	3 days	Mon 3/2/15	Wed 3/4/15
	Face	3 days	Thu 3/5/15	Mon 3/9/15
	Object3DModel	10 days	Mon 3/2/15	Fri 3/13/15
	Android Unit Testing	**25 days**	**Mon 3/2/15**	**Fri 4/3/15**
	MainMenuActivity	3 days	Mon 3/2/15	Wed 3/4/15
	Vertex	3 days	Mon 3/2/15	Wed 3/4/15
	Object3DModel	10 days	Mon 3/2/15	Fri 3/13/15
	Face	3 days	Thu 3/5/15	Mon 3/9/15
	ModelPhotoGalleryActivity	4 days	Tue 3/10/15	Fri 3/13/15
	ImageCaptureActvity	4 days	Tue 3/10/15	Fri 3/13/15
	HelpActivity	5 days	Mon 3/16/15	Fri 3/20/15
	ModelViewActivity	11 days	Sat 3/21/15	Fri 4/3/15
	Requirement Validation	0 days	Fri 4/3/15	Fri 4/3/15
	Alpha Release	0 days	Fri 3/13/15	Fri 3/13/15
	Beta Release	0 days	Fri 4/17/15	Fri 4/17/15
	Project Presentation	0 days	Wed 5/6/15	Wed 5/6/15

Figure 12: Gantt Chart Timeline

4.3 Tasks and Experience

Wesley has a background in algorithm development and has a keen eye for logical analysis, which would make him more apt to develop the 3D conversion and image processing algorithms. Ryan has experience in GUI development, website development and graphic design, which would make him more suited for creating the app framework.