CS 491 Senior Design Project

Analysis Report

DepthCube

Students:
Alican Büyükçakır
Serkan Demirci
Semih Günel
Burak Savlu

Supervisor:
Prof. Uğur Güdükbay

Jury:
Assoc. Prof. Selim Aksoy
Prof. Özgür Ulusoy

Innovation Expert:
Armağan Yavuz (TaleWorlds)

November 6, 2016

Contents

Introduction

DepthCube is a mobile application that aims to extend mobile devices' environmental understanding for augmented reality (AR) purposes. DepthCube takes your everyday environment and turns it into a virtual playground which you see through mobile device's camera. We want to create an environment where virtual objects interact with the real formation of your environment. In this sense, players will interact with the applications using their mobile devices as gamepads.

In the core application, DepthCube will provide applications with the precise location of the device relative to the room with the fine details about formation of the environment. The problem with the current state of the AR applications is the lack of computational power on the mobile devices, combined with the difficulty of providing real-time environmental formation. We want to tackle these obstacles using state-of-the-art computer vision algorithms, minimizing the need of an additional hardware.

Current Systems

There are some products that use similar technologies to the DepthCube. There are game consoles such as Kinect, or additional sensors like Structure Sensor use depth information to build 3D structure of the environment. Although, they can perform camera tracking and 3D model construction faster, these sensors have limited range for their depth measurements. They cannot construct 3D model of the large environments. Additionally, there are some AR frameworks that run for mobile devices. With these frameworks, developers can build AR applications that can interact with the environment without need of any external hardware. However, they cannot perform well due to low computational power of the target mobile devices. In addition, they use a lot of battery for their calculation.

The most similar technology with DepthCube is Project Tango, currently developed by Google. Project Tango wants to solve the problem of area awareness in real time, and their ultimate goal is the simultaneous localization and mapping. Tango devices include a powerful NVIDIA GPU, a depth sensor, and RGB camera with large field of view. All these to support real-time computations. However these which makes the Tango device relatively expensive, compared to main-stream mobile devices, and it is almost stands like specifically built for AR and VR purposes, rather than serve as a general mobile device. Tango device is heavy and thick compared to the main-stream phones and tablets.

Proposed System

3.1 Overview

DepthCube is an API that utilizes mobile devices' camera for information on real life environment in order to create a virtual playground for users. Our ultimate goal is to create a cheap and effective API that can enable many AR applications including but not limited to games. When we look at similar systems, we can see many innovations originated from the Kinect console, which provided developers with the depth sensor and skeletal tracking. With DepthCube, it is possible to do the same thing, but this time without need of any additional hardware. We believe this makes the product very appealing to customers not only due to its low price but also because it is very accessible for anyone with a mobile device.

To clarify how DepthCube functions, assume that we have a game that is already developed and ready to play for the users. Our exemplary new game will be very similar to Pong, but this time balls bounce on the real walls and objects, rather than on rectangular borders. Users will interact with the virtual ball using their mobile device as a virtual surface where virtual ball bounce on. Here is how the system will work:

- First user will scan her environment using the mobile device's camera. Simultaneously DepthCube will create a sparse representation of the environment, and show the 3D model on the mobile device.
- When scanning phase has been completed, DepthCube will create a dense representation of the room in our dedicated cloud servers. This construction might take several hours. At the end of this construction, DepthCube will create the environment in Unity model, which correctly represents the room.
- Then, Unity model will be transferred to the mobile device which runs the DepthCube Android application, which is built upon Unity Engine.
- Once Unity model is received and wireless connection between the mobile device and server is established, users can run games or applications that are built with DepthCube Unity Framework on their phones.
- During the game(or application) phase, using the constructed scene model from the previous step, location information will be available for the user.

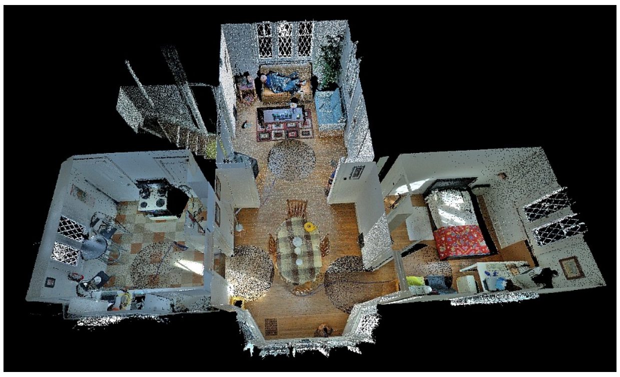

Figure 3.1: DepthCube's understanding the room after scan phase. Each vertex corresponds to a 3D point.

- Since Unity engine has the fine details of the room with the precise location of the mobile device, Unity engine will draw the virtual objects in the correct location and size without disrupting the perspective.

3.2 Requirements

3.2.1 Functional Requirements

- Players should have their own unique username/password.
- DepthCube should not require anything other than an Android device to run.
- Games should be controlled using a mobile device(smartphone/tablet).
- There should be two types of users for the product: developers and players.
- Players should be able to experience the Augmented Reality, given an already developed and ready-to-play game.
- Players should be able to upload videos for 3D construction.
- DepthCube should process user videos to create 3D structure
- Players should be able to download games from the online library.
- Developers should be able to design and develop their own games using our API that provides the location and structural information about the environment.

3.2.2 Non-Functional Requirements

- Each player should have access to the last 5 3D structures they created.
- DepthCube should support Android 4.4 and above.
- A local and fast connection between DepthCube' server and mobile device is required.
- Mobile interface of the DepthCube application should be user-friendly, clean and simplistic. Users must not have any problems with navigating themselves inside the application.
- Games developed for DepthCube should have at least 30 FPS.
- The provided API for developers should be well-documented, precise and succinct.
- DepthCube should scan user video and create a 3D structure in less than 1 hour.
- DepthCube should support games developed using Unity3D version 5.0.0 and above.
- A system crash should not result in data loss

3.2.3 Pseudo-Requirements

- The server code should be written in C++.
- User informations (username, password and email) should be kept in a relational database (e.g. MySQL).
- The provided API should be developed on Unity3D.

3.3 System Models

3.3.1 Use Case Model

The following is the Use Case diagram that shows the functionalities of our application. Below the figure, use cases are further explained.

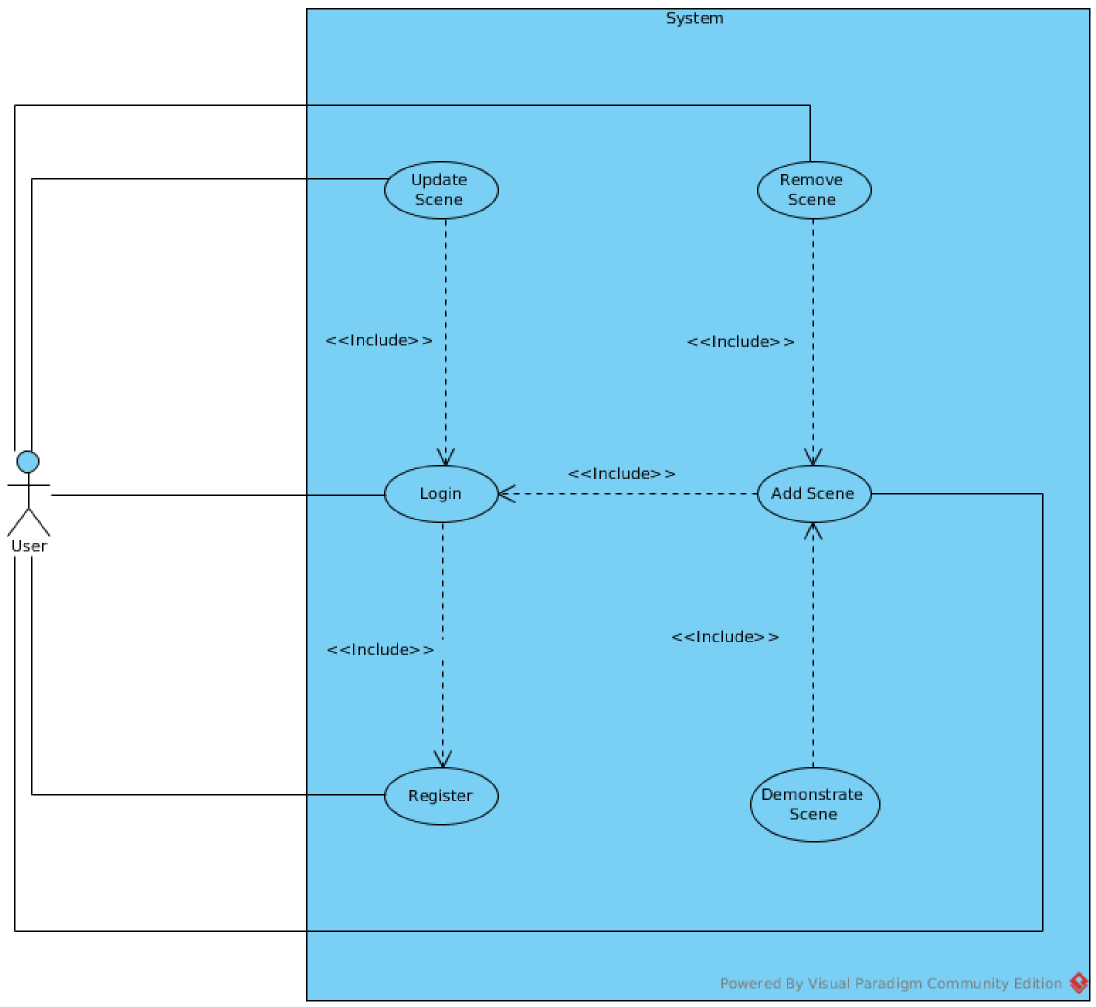

Figure 3.2: Use Case Diagram.

Firstly, user has to be registered to be able to log in. A logged in user can add a scene. This scene can be further modified (updated or deleted). Finally, user demonstrates a scene which has to be previously added to the user's scene list.

Use Case Name: Register
Participating Actors: User
Entry Condition: None.
Flow of Events:

1. User opens the main menu of the application.
2. User clicks on 'Register' button.
3. User fills the blanks with his credentials.
4. User clicks on 'Sign Up' button.

Exit Conditions:

1. User clicks on “Back to Main Menu” button.

Use Case Name: Login
Participating Actors: User
Entry Condition: Register
Flow of Events:

1. User opens the main menu of the application.
2. User clicks on 'Login' button.
3. User fills the blanks with his credentials.
4. User clicks on 'Login' button and logs in.

Exit Conditions:

1. 1. User clicks on “Back to Main Menu” button

Use Case Name: Add Scene
Participating Actors: User
Entry Condition: User clicks to 'My Scenes' button
Flow of Events:

1. User clicks 'Add Scene' button.

2. User starts recording his environment
    (a) User receives feedback about the construction of the scene
3. User stops recording the scene.
4. User waits for the scene construction to be completed.
5. User is shown the constructed scene.
6. User clicks ’Accept’, ’Update’ or ’Cancel’ button to keep, update or discard the scene respectively.

Exit Conditions:

1. User clicks on “Back to Main Menu” button

Use Case Name: Remove Scene
Participating Actors: User
Entry Condition: User clicks to ’My Scenes’ button
Flow of Events:

1. User chooses a scene to remove from the list of the scenes.
2. User clicks on ’Remove Scene’ button.
3. User is given a confirmation prompt to finalize his answer.
4. User clicks on ’Yes’ or ’No’ depending on his decision.

Exit Conditions:

1. 1. User clicks on “Back to Main Menu” button

Use Case Name: Update Scene
Participating Actors: User
Entry Condition: User clicks on ’Update’ at the end of ’Add Scene’
Flow of Events:

1. User chooses a scene to remove from the list of the scenes.

2. User clicks on ’Remove Scene’ button.
3. User is given a confirmation prompt to finalize his answer.
4. User clicks on ’Yes’ or ’No’ depending on his decision.

Exit Conditions:

1. 1. User clicks on “Back to Main Menu” button

Use Case Name: Demonstrate Scene
Participating Actors: User
Entry Condition: User clicks on ’My Scenes’ button
Flow of Events:

1. User chooses a scene.
2. User clicks on ’Demonstrate’ button.
3. User is shown his location in the reconstructed scene.
   (a) The shown location of user in the demonstration is updated real-time.

Exit Conditions:

1. User clicks on “Back to Main Menu” button

3.3.2 Object and Class Model

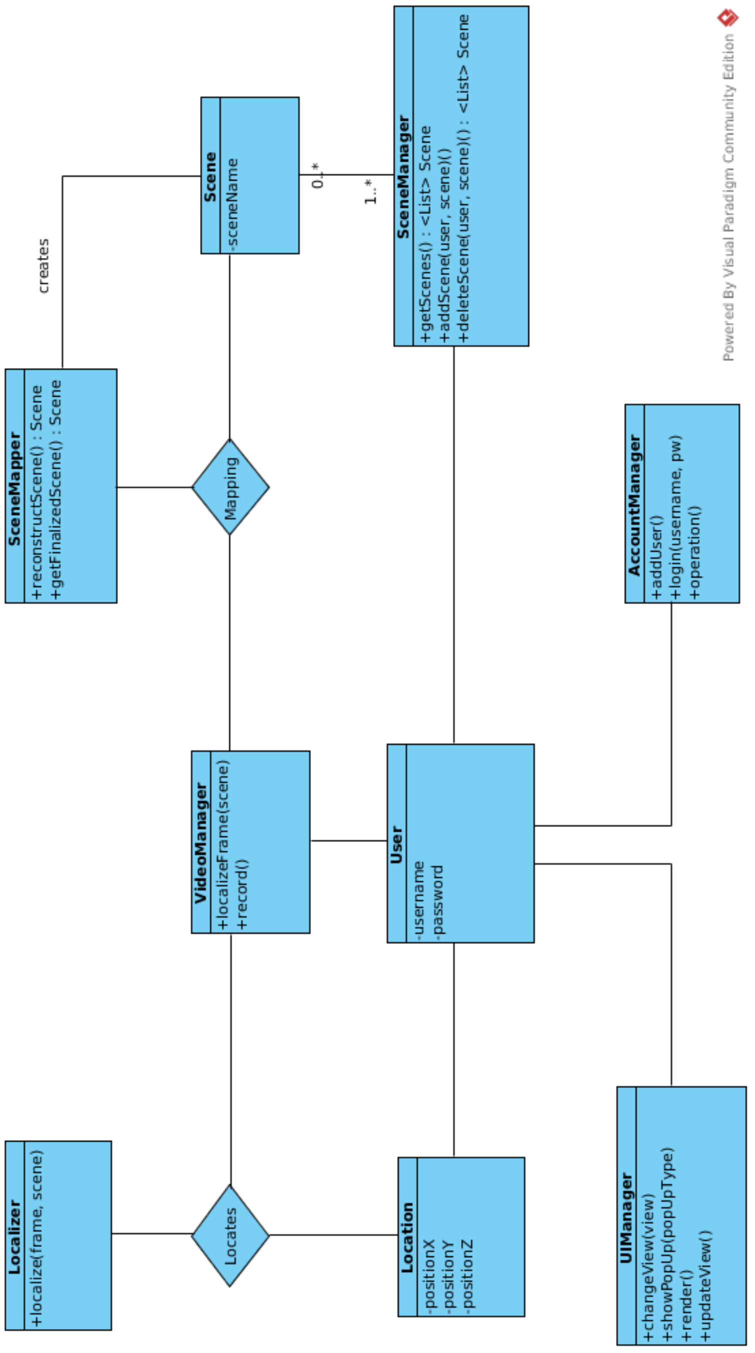

Figure 3.3: Class Diagram.

3.3.3 Dynamic Models

Sequence Diagrams

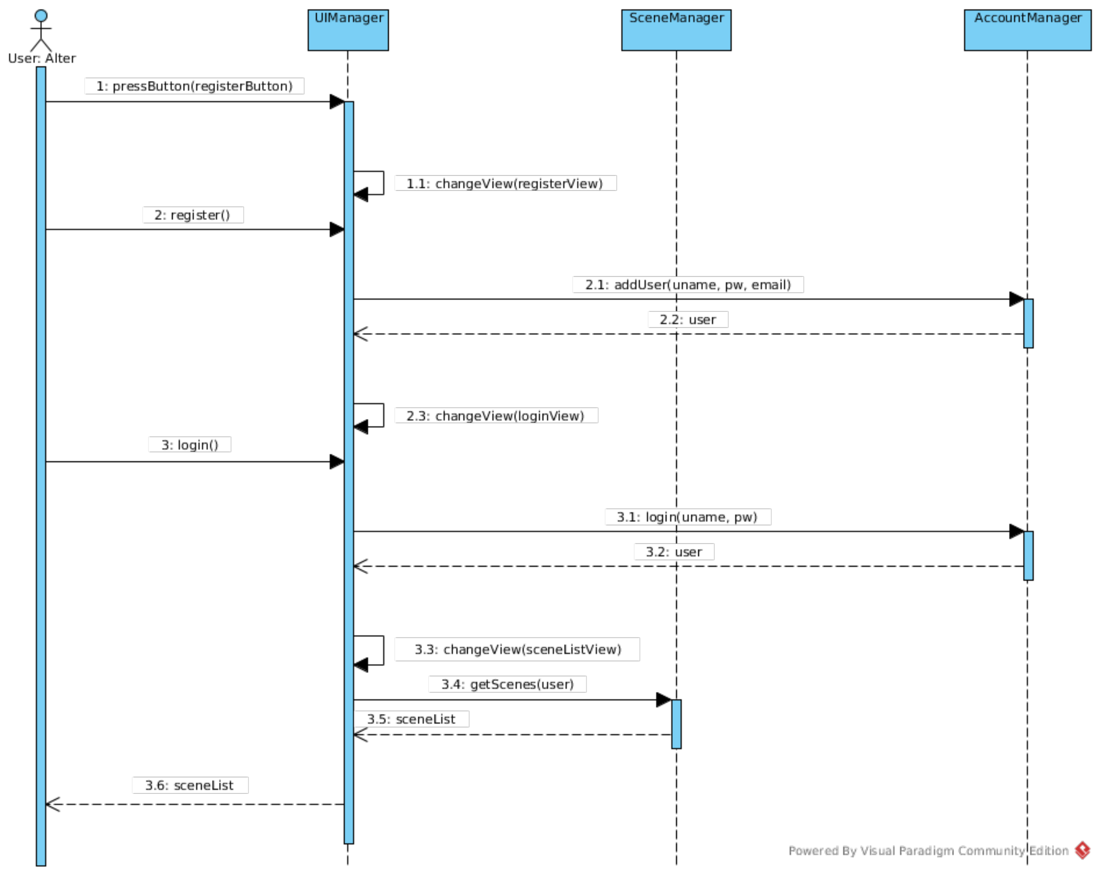
User: Alter
UIManager
SceneManager
AccountManager
1: pressButton(registerButton)
1.1: changeView(registerView)
2: register()
2.1: addUser(uname, pw, email)
2.2: user
2.3: changeView(loginView)
3: login()
3.1: login(uname, pw)
3.2: user
3.3: changeView(sceneListView)
3.4: getScenes(user)
3.5: sceneList
3.6: sceneList
Powered By Visual Paradigm Community Edition

Figure 3.4: Sequence Diagram for Register and Login processes.

Scenario Name: Register and Login
Actors: User
Flow of Events:
This is Alter's first time opening the application. He first clicks on 'Register' button and fills in the required fields for the registration to be completed. Then, he presses 'Register' button again to submit his registration. After registration is completed, application returns back to main menu. Alter, happily, fills in his username and password and logs in to the system. He then encounters with the Scene List screen.

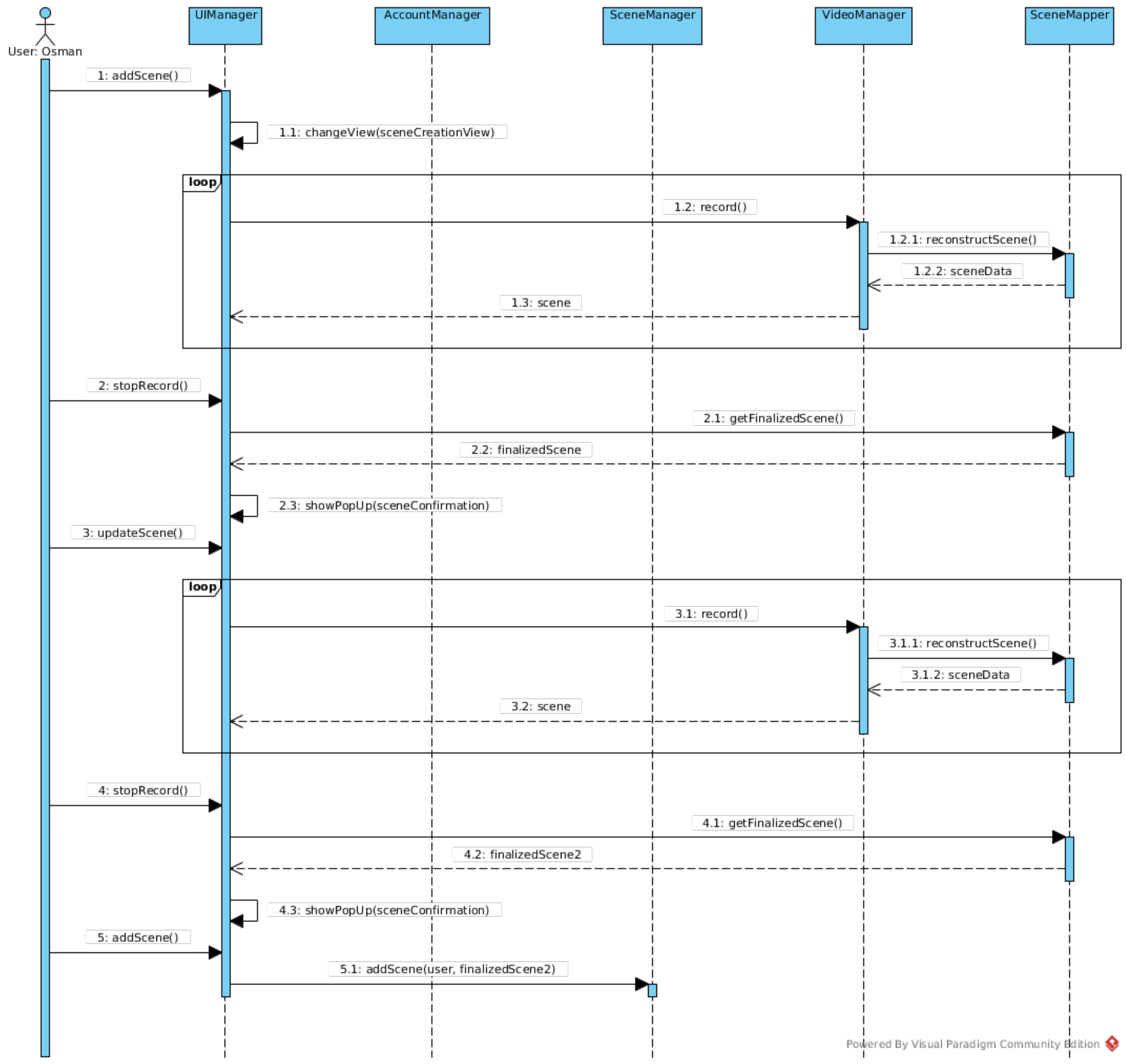

Figure 3.5: Sequence Diagram for Scene Creation.

Scenario Name: Scene Creation
Actors: User
Flow of Events:
Osman is already registered to the system and logged in to the application. He presses 'Add Scene' button and his device's camera starts recording his environment. Meanwhile, he can see the current state of the reconstructed scene in his device's screen. He walks around his environment and he sees that the reconstructed scene that is displayed gets better and

better as he shows more and more frames of the environment. Osman acknowledges that the reconstructed scene looks good enough and stops recording. He waits for some time so that the finalized version of the scene is constructed. He does not like the final version and wants to improve some parts of the reconstructed scene. He presses 'Update Scene' and stars recording with his device's camera again. At the end of this recording session, he clicks on 'Accept' to confirm the scene's reconstruction as satisfactory. The scene is saved in the scene list.

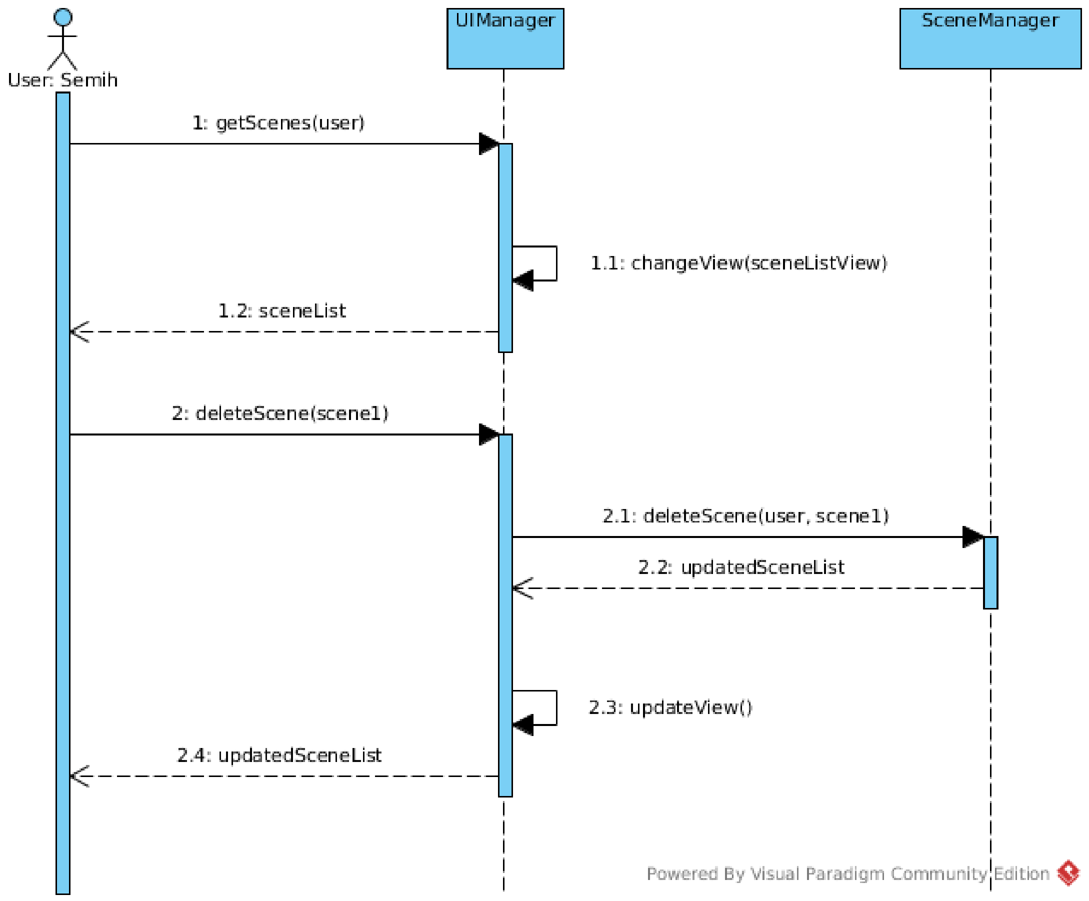

Figure 3.6: Sequence Diagram for Scene Deletion.

Scenario Name: Scene Deletion
Actors: User
Flow of Events:
Semih moved from his old house and wants to delete the scene data of his house from his scene list. He enters the scene list and selects the scene that he wishes to delete. Then he presses 'Remove' button. The scene is deleted from the scene list.

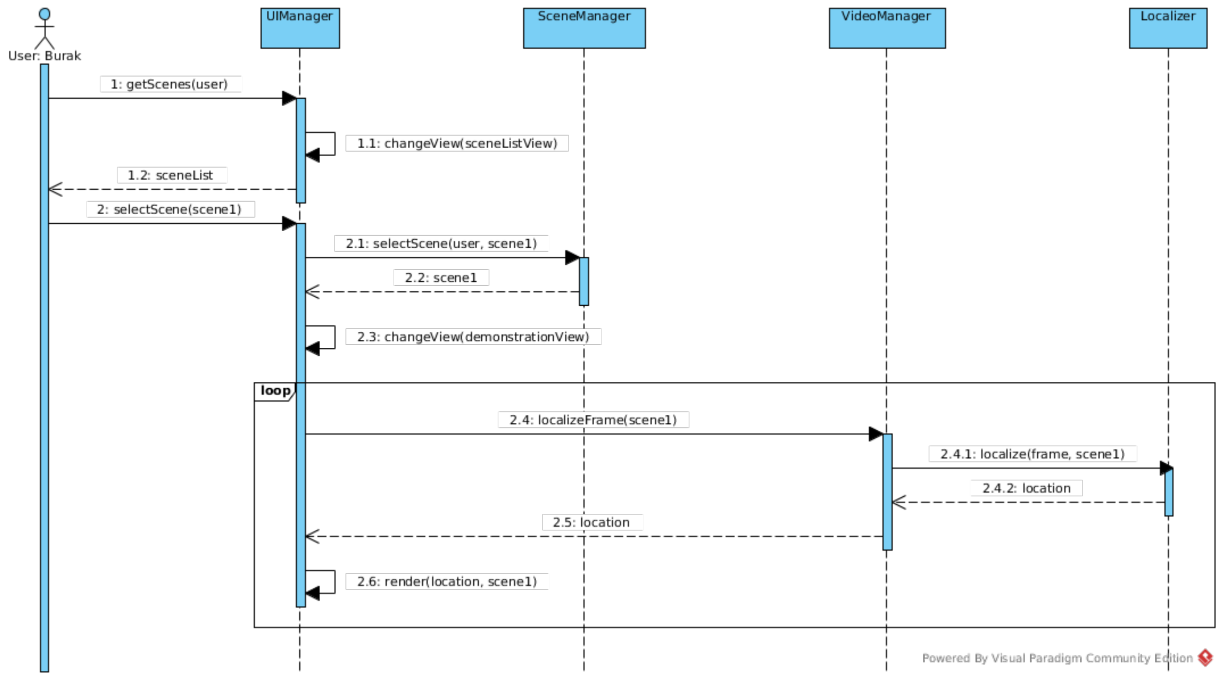
User: Burak
UIManager
SceneManager
VideoManager
Localizer
1: getScenes(user)
1.1: changeView(sceneListView)
1.2: sceneList
2: selectScene(scene1)
2.1: selectScene(user, scene1)
2.2: scene1
2.3: changeView(demonstrationView)
loop
2.4: localizeFrame(scene1)
2.4.1: localize(frame, scene1)
2.4.2: location
2.5: location
2.6: render(location, scene1)
Powered By Visual Paradigm Community Edition

Figure 3.7: Sequence Diagram for Scene Demonstration.

Scenario Name: Scene Demonstration
Actors: User
Flow of Events:
Burak wants to show preciseness and amazingness DepthCube to his friends. He enters a room that he previously built a scene from. Then, he selects a scene from the scene list. He presses 'Demo' button to initiate augmented reality showcase. As he moves around his room, he sees on his phone's screen that his location is updated and precisely located in the 3D-model of his room.

3.3.4 User Interface and Screen Mockups

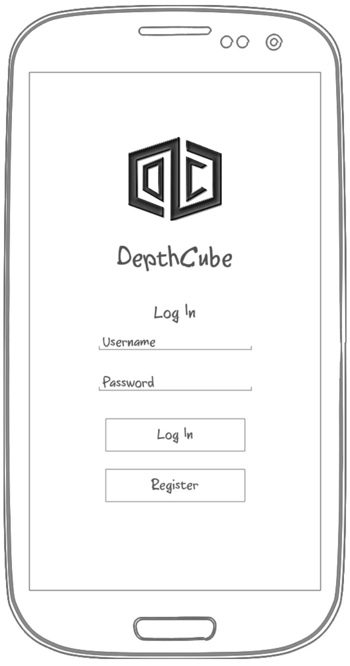

Figure 3.8: Main Menu Screen.

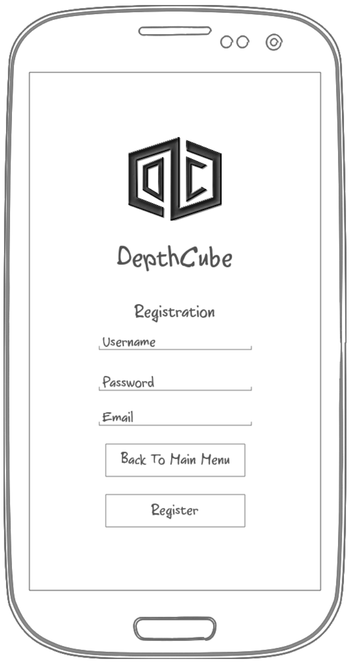

Figure 3.9: Registration Screen.

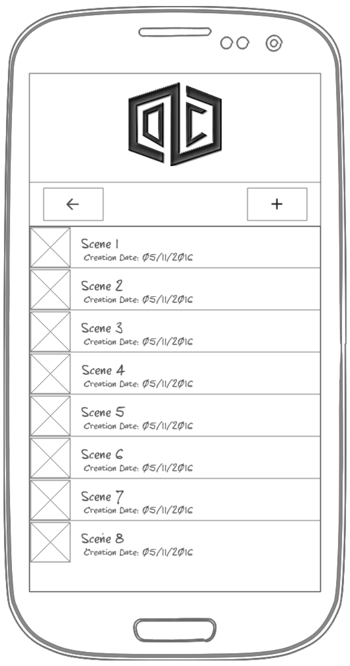
←
+
Scene 1
Creation Date: 05/11/2016
Scene 2
Creation Date: 05/11/2016
Scene 3
Creation Date: 05/11/2016
Scene 4
Creation Date: 05/11/2016
Scene 5
Creation Date: 05/11/2016
Scene 6
Creation Date: 05/11/2016
Scene 7
Creation Date: 05/11/2016
Scene 8
Creation Date: 05/11/2016

Figure 3.10: Scene List.

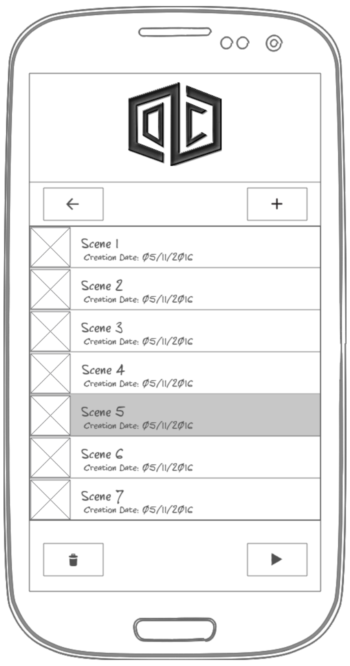
←
+
Scene 1
Creation Date: 05/11/2016
Scene 2
Creation Date: 05/11/2016
Scene 3
Creation Date: 05/11/2016
Scene 4
Creation Date: 05/11/2016
Scene 5
Creation Date: 05/11/2016
Scene 6
Creation Date: 05/11/2016
Scene 7
Creation Date: 05/11/2016
▶

Figure 3.11: Scene List when a scene is selected.

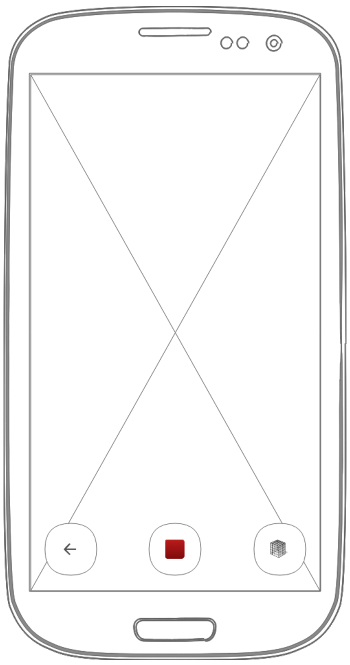

Figure 3.12: Record Reconstruction Screen.

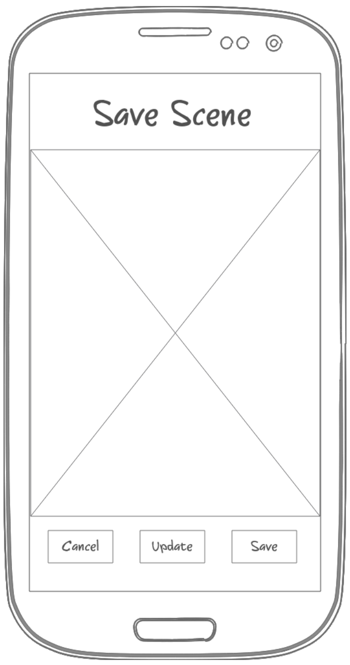

Figure 3.13: Record Dialog Screen.

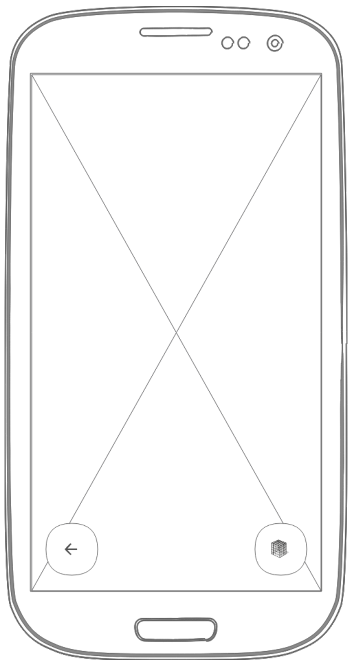
Figure 3.14: Demonstration Screen.

References

Project Tango:
`http://get.google.com/tango/`

Structure From Motion:
`http://bit.ly/2dSxD7L`

OpenCV Camera Motion Estimation:
`http://docs.opencv.org/3.1.0/d5/dab/tutorial_sfm_trajectory_estimation.html`

CS491 Senior Design Project I - Guidelines:
`http://www.cs.bilkent.edu.tr/CS491-2/CS491.html`

ORB-SLAM, structure from motion implementation for monocular cameras:
`http://webdiis.unizar.es/~raulmur/MurMontielTardosTRO15.pdf`

Kinect with Skeletal Tracking:
`https://msdn.microsoft.com/en-us/library/jj131025.aspx`

Kinect Sensor:
`https://msdn.microsoft.com/en-us/library/hh438998.aspx`

Kinect Sensor Hardware Specifications:
`https://developer.microsoft.com/en-us/windows/kinect/hardware`

Structure Sensor:
`https://store.structure.io/store`