Capstone Design

Team 12

20165729 Park Sangwoo
20161344 Heo JeongWoo
20161090 Lee Chaemin

Contents

1. Project Summary
2. Feedback
3. Schedule Modification
4. Progress Report
5. Q&A

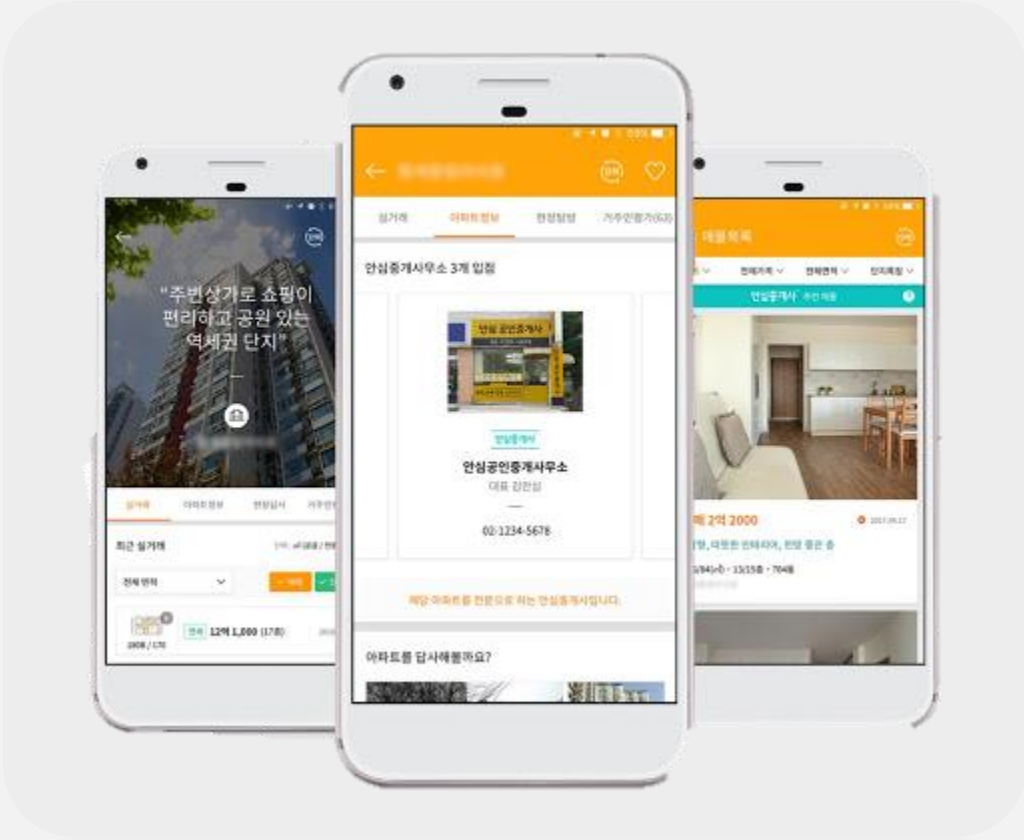
1.

Project Summary

1

Project Summary

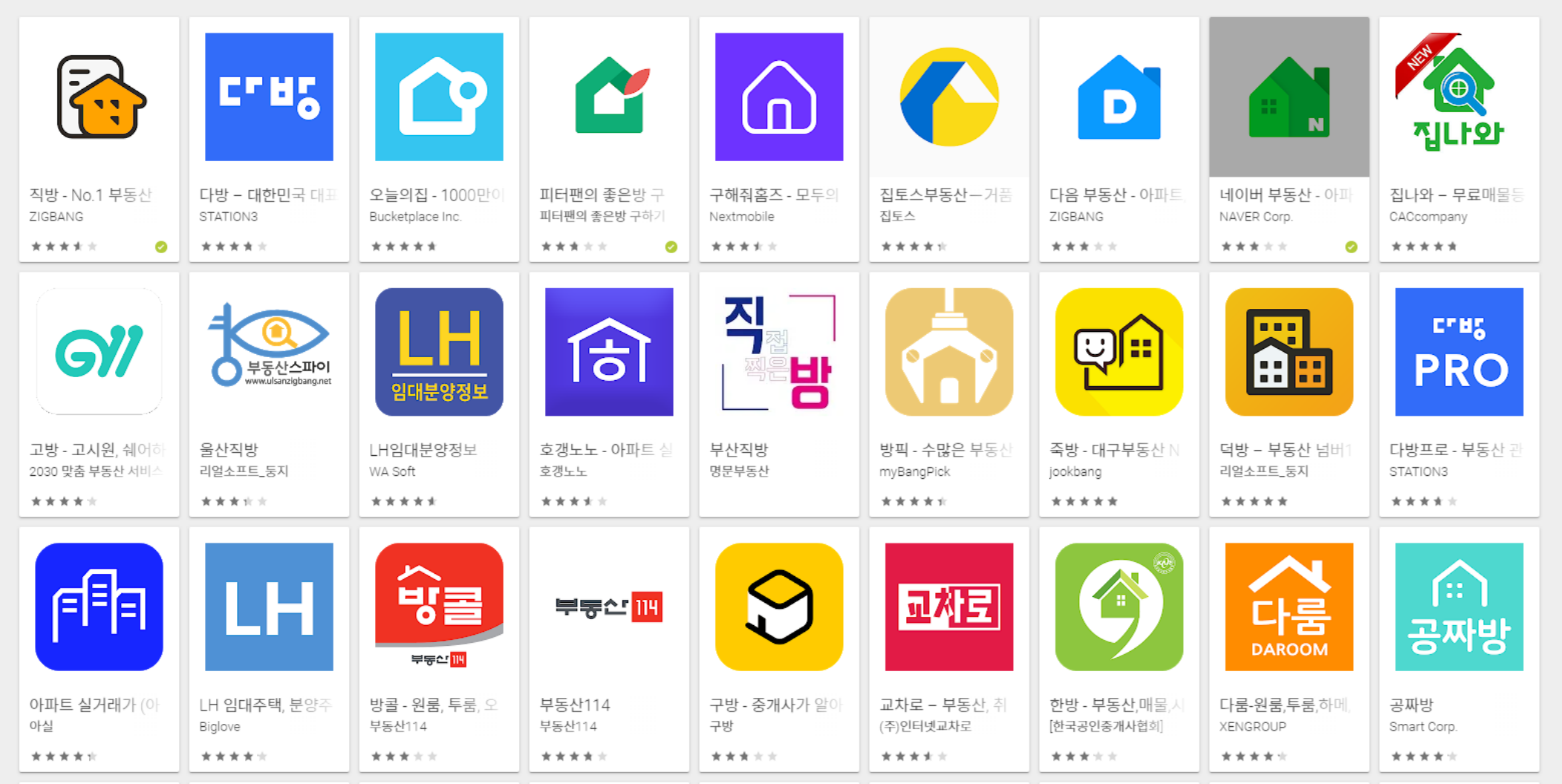
Similar Application



1

Project Summary

Similar Application



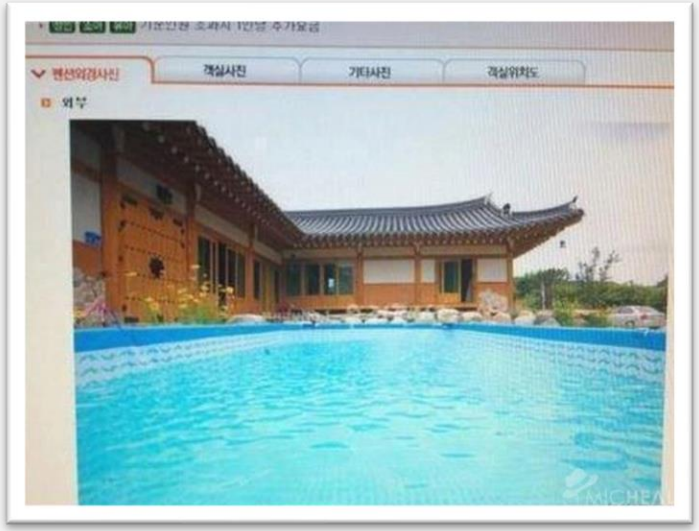
Similar Application

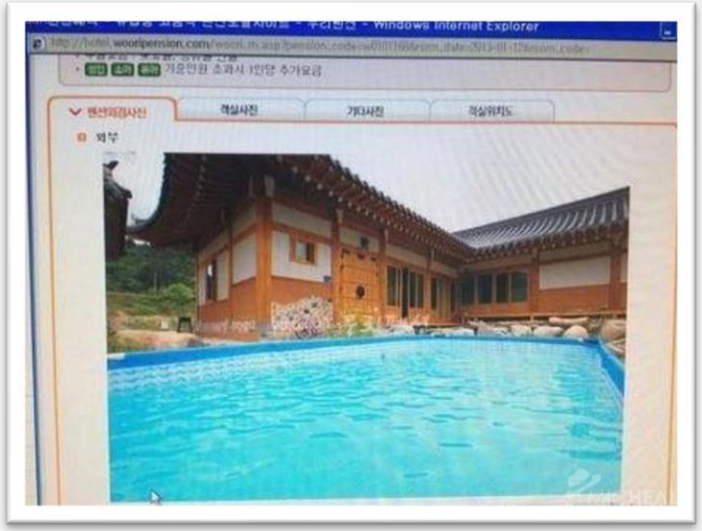
A lot of Real Estate Brokerage Applications are already existing

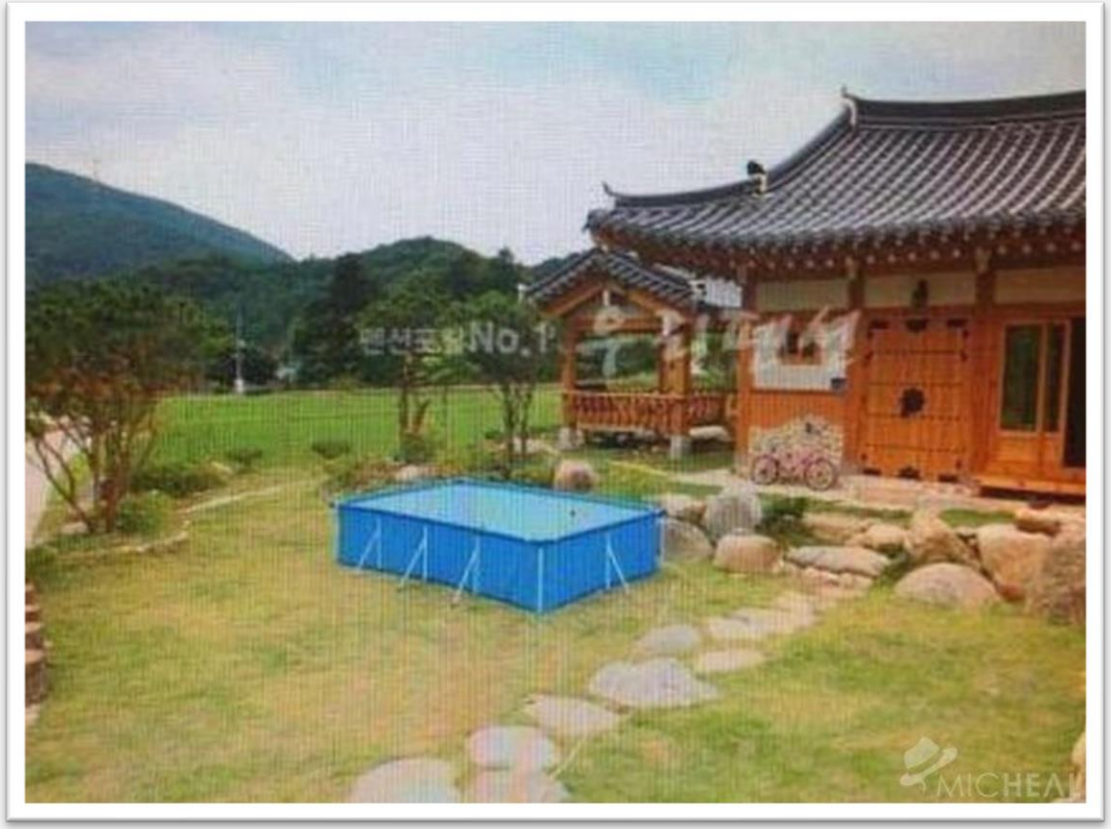
1

Project Summary

Similar Application



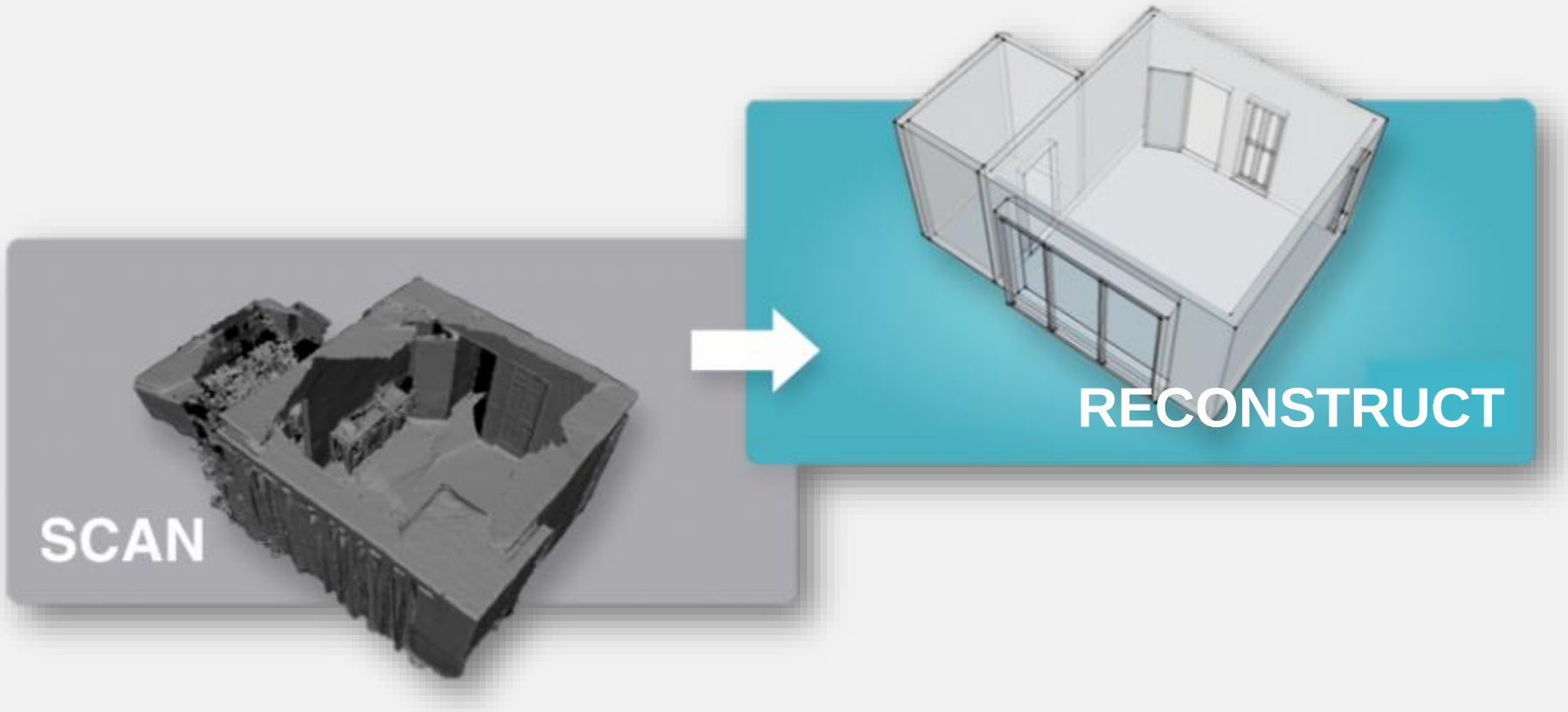




1

Project Summary

Reconstruct



Using 3D Scanning!

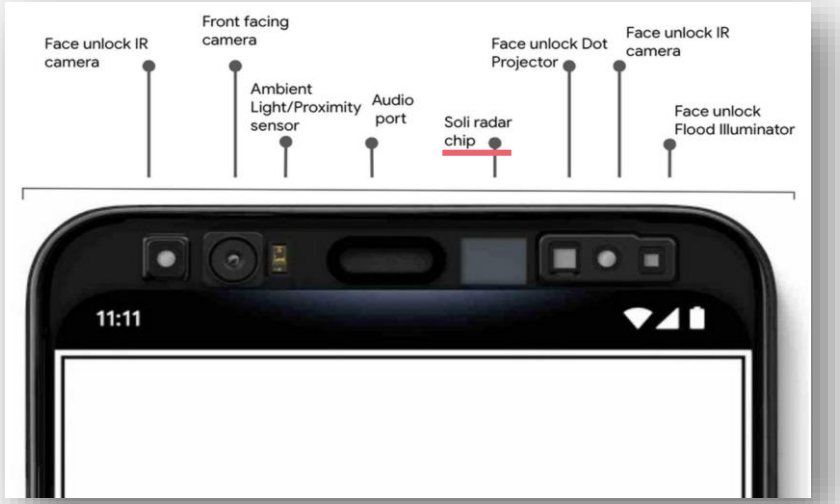
3D Scanning

With ToF, Lidar sensors.

1

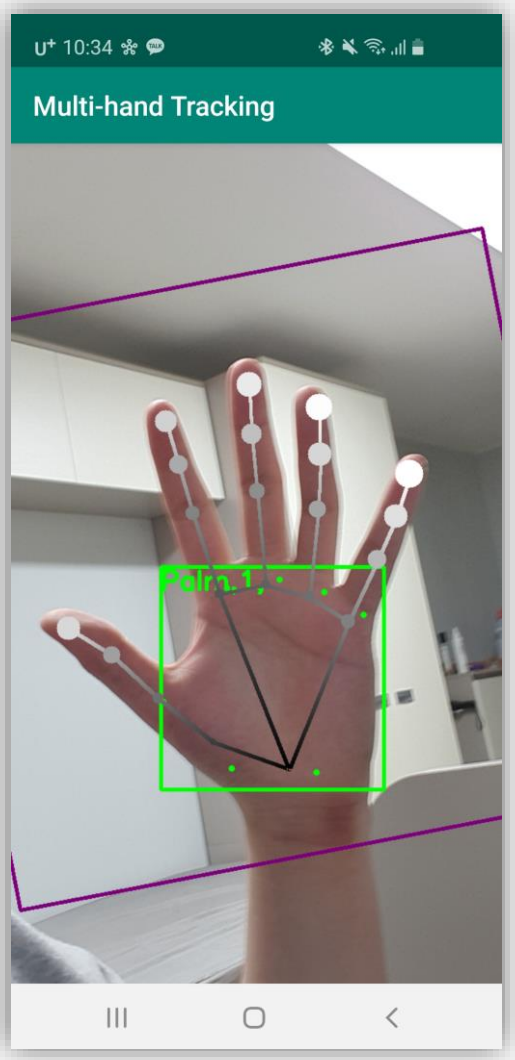
Project Summary

3D Scanning



VR Compatible (Optional)

Optional



Control by Hand Tracking

Not a just virtually place furnishings application

2.

Feedback

1

Feedback

What’s differences?

What are the differences between other applications, such as IKEA applications?

2

Feedback

What’s differences?

SWOT Research

Strength

- You can check the apartment sales reviews in the application
  Categorized according to age, gender, rent-based, etc.
- Convenience and Speed of Real Estate Exploration
  24-hour service via the Internet.
  You can easily check for sale without having to go to a brokerage house.
  Explore things beyond time and space with real-time feedback.

Weakness

- There are many false offerings.
- Difficult to know the exact information about the house.

Opportunity

-An increase in the share of monthly rent, and leaseholds.
-Real estate agent’s interest
  Meditation through mobile services can save both tenants and brokers time and money in real estate sales transactions.

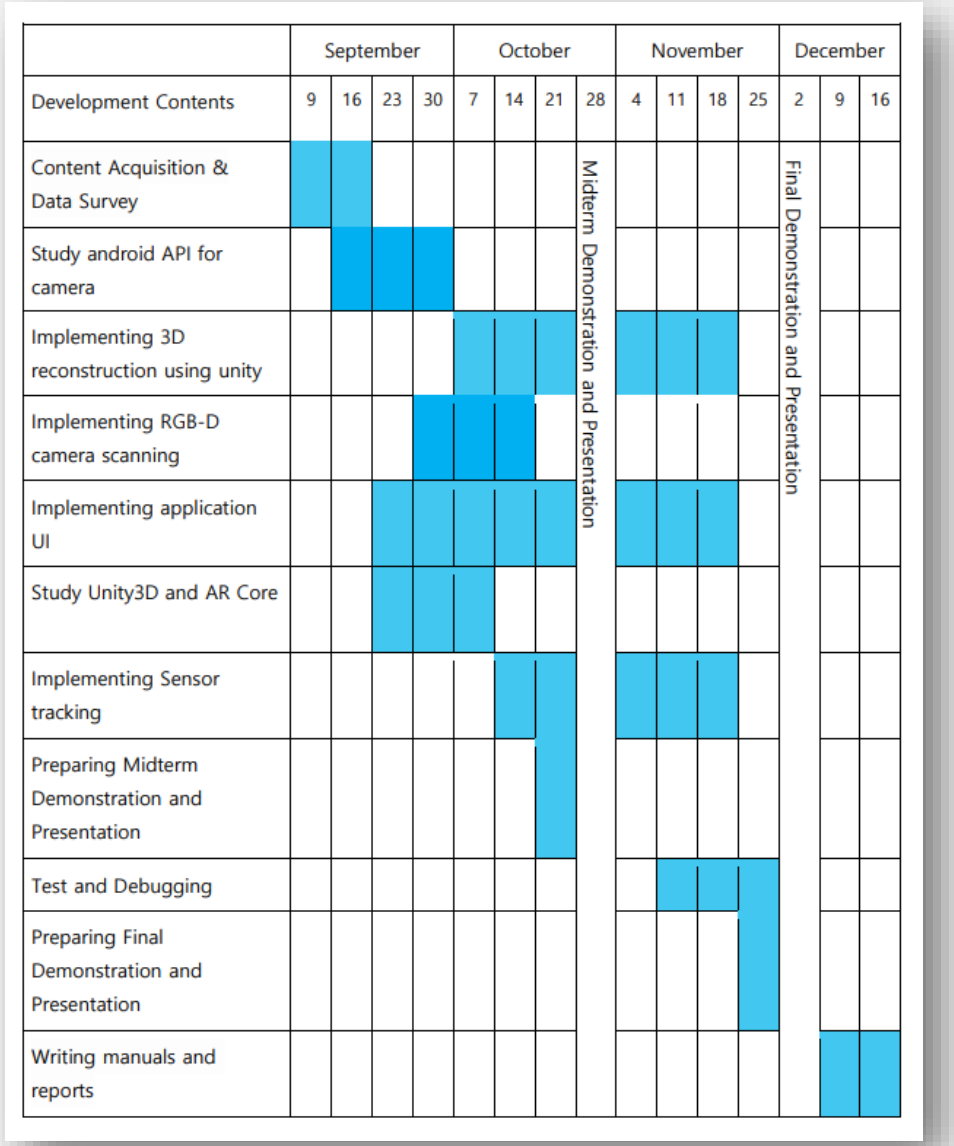
Threat

-Multiple Competitors increase.

3.

Schedule Modification

3 Total Schedule



3 Schedule

1. Heo Jeong-woo

Development Contents	September				October				November				December		
	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16
Content Acquisition & Data Survey								Midterm Demonstration and Presentation					Final Demonstration and Presentation		
Study android API for camera															
Implementing RGB-D camera scanning															
Implementing application UI															
Implementing Sensor tracking															
Preparing Midterm Demonstration and Presentation															
Test and Debugging															
Preparing Final Demonstration and Presentation															
Writing manuals and reports															

2. Lee Chae-min

Development Contents	September				October				November				December		
	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16
Content Acquisition & Data Survey								Midterm Demonstration and Presentation					Final Demonstration and Presentation		
Study android API for camera															
Implementing 3D reconstruction using unity															
Implementing application UI															
Study Unity3D and AR Core															
Preparing Midterm Demonstration and Presentation															
Test and Debugging															
Preparing Final Demonstration and Presentation															
Writing manuals and reports															

3. Park Sang-woo

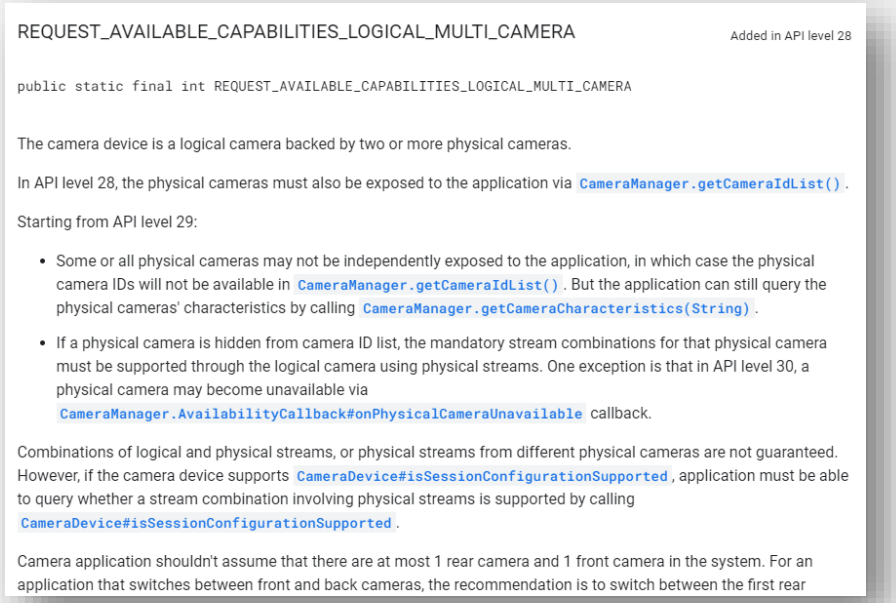
Development Contents	September				October				November				December		
	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16
Content Acquisition & Data Survey								Midterm Demonstration and Presentation					Final Demonstration and Presentation		
Study android API for camera															
Implementing 3D reconstruction using unity															
Implementing application UI															
Study Unity3D and AR Core															
Preparing Midterm Demonstration and Presentation															
Test and Debugging															
Preparing Final Demonstration and Presentation															
Writing manuals and reports															

4.

Progress Report

Progress Report

Android API for Multi-Camera



REQUEST_AVAILABLE_CAPABILITIES_LOGICAL_MULTI_CAMERA

Added in API level 28

```
public static final int REQUEST_AVAILABLE_CAPABILITIES_LOGICAL_MULTI_CAMERA
```

The camera device is a logical camera backed by two or more physical cameras.

In API level 28, the physical cameras must also be exposed to the application via `CameraManager.getCameraIdList()`.

Starting from API level 29:

- Some or all physical cameras may not be independently exposed to the application, in which case the physical camera IDs will not be available in `CameraManager.getCameraIdList()`. But the application can still query the physical cameras' characteristics by calling `CameraManager.getCameraCharacteristics(String)`.
- If a physical camera is hidden from camera ID list, the mandatory stream combinations for that physical camera must be supported through the logical camera using physical streams. One exception is that in API level 30, a physical camera may become unavailable via `CameraManager.AvailabilityCallback#onPhysicalCameraUnavailable` callback.

Combinations of logical and physical streams, or physical streams from different physical cameras are not guaranteed. However, if the camera device supports `CameraDevice#isSessionConfigurationSupported`, application must be able to query whether a stream combination involving physical streams is supported by calling `CameraDevice#isSessionConfigurationSupported`.

Camera application shouldn't assume that there are at most 1 rear camera and 1 front camera in the system. For an application that switches between front and back cameras, the recommendation is to switch between the first rear

Android Multi-Camera Camera2 API

OR

Progress Report

Android API for Multi-Camera

app / com.dhirajgupta.multicam.services / ManagedCamera

ManagedCamera

`class ManagedCamera`

This class is heavily inspired from Google's Camera2Basic sample at https://github.com/googlesamples/android-Camera2Basic Heavy refactoring has been carried out to move all the code from the Fragment in the sample to a self contained ManagedCamera class that can take care of it's own threads and is almost fully self sufficient. By modularizing the Camera in this way we are able to easily create multiple instances (two in this case) while the implementation code remains common between both the cameras.

Constructors

Name	Summary
<init>	`ManagedCamera(systemId: String , threadName: String , textureView: AutoFitTextureView , listener: ManagedCameraStatus )` This class is heavily inspired from Google's Camera2Basic sample at https://github.com/googlesamples/android-Camera2Basic Heavy refactoring has been carried out to move all the code from the Fragment in the sample to a self contained ManagedCamera class that can take care of it's own threads and is almost fully self sufficient. By modularizing the Camera in this way we are able to easily create multiple instances (two in this case) while the implementation code remains common between both the cameras.

Properties

Name	Summary
activity	`val activity: Activity`
	`var cameraState: Int`

Android Camera2Basic Sample

This sample demonstrates using the Camera2 API to capture a JPEG, DEPTH or RAW frame. Check the source code to see a simple example of how to display the camera preview and capture a still image using the default configuration with the selected pixel format.

Introduction

The Camera2 API allows users to capture RAW images, i.e. unprocessed pixel data directly from the camera sensor that has not yet been converted into a format and colorspace typically used for displaying and storing images viewed by humans. The DngCreator class is provided as part of the Camera2 API as a utility for saving RAW images as DNG files.

This sample displays a live camera preview in a TextureView, and saves JPEG and DNG file for each image captured.

Pre-requisites

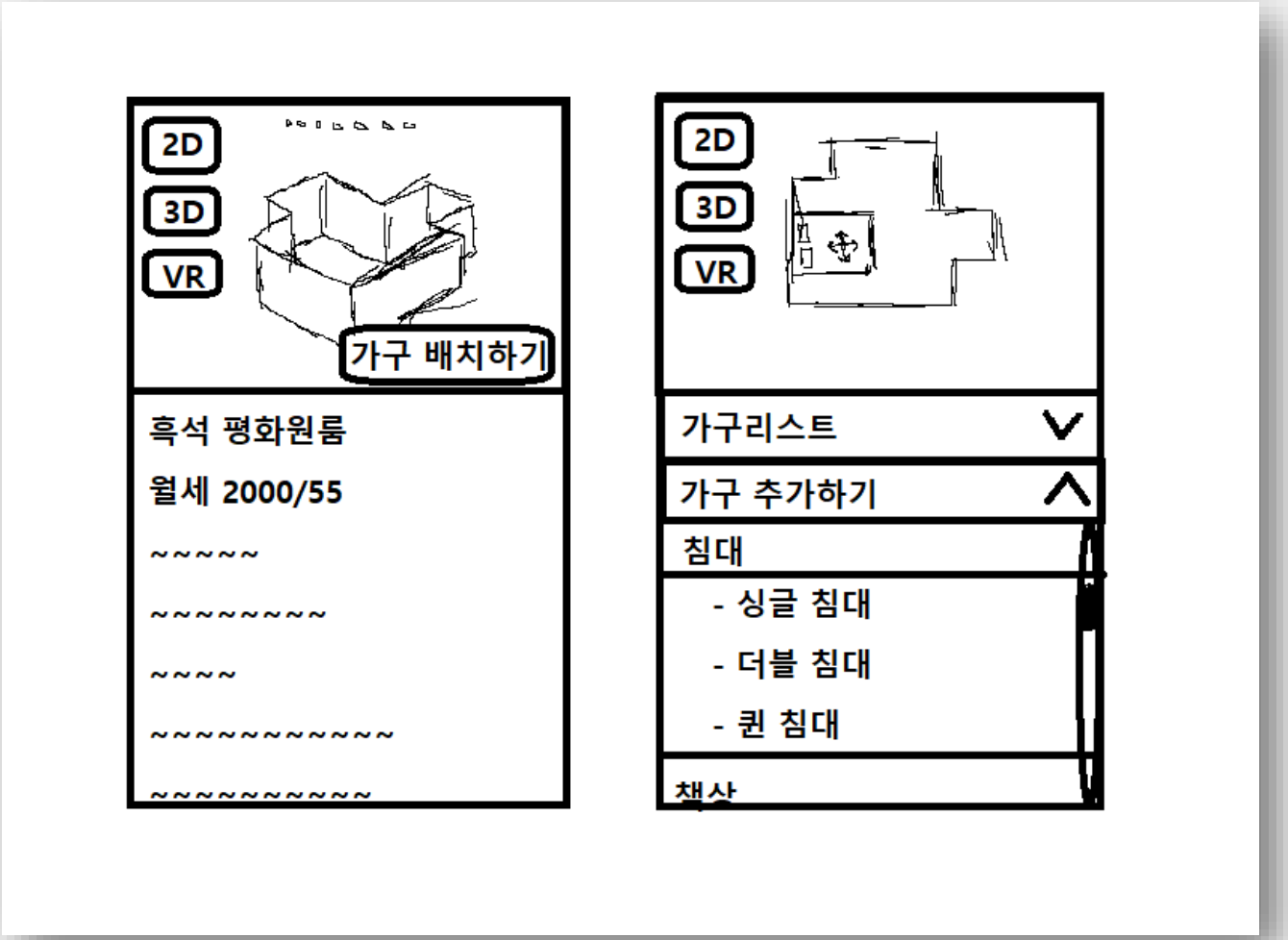
- Android SDK 29+
- Android Studio 3.5+

Screenshots

Progress Report

Android API for Multi-Camera

UI



Progress Report

Unity

ARGuidePanel Import Settings

Open... Execution Order...

AR Panel None (Game Object)

Default references will only be applied in edit mode.

mported Object

ARGuidePanel

Assembly Information

Filename Assembly-CSharp.dll

```
sing System.Collections;
sing System.Collections.Generic;
sing UnityEngine;
sing GoogleARCore;

ublic class ARGuidePanel : MonoBehaviour {

  public GameObject ARPanel; // AR 트래킹에 대한 패널, 동영상

  private List<DetectedPlane> detectedPlanes = new
ist<DetectedPlane>(); // 트래킹 바닥 리스트

  private void Update()
  {
     if (Session.Status != SessionStatus.Tracking) // 현재 바닥을 찾는
태가 아니라면 실행 X
     {
        return;
     }

     Session.GetTrackables<DetectedPlane>(detectedPlanes,
rackableQueryFilter.All); // 트래킹된 바닥 모두 가져온다.
     foreach (DetectedPlane plane in detectedPlanes)
     {
        if(plane.TrackingState == TrackingState.Tracking) // 트래킹 된
바닥이 있으면 ARPanel (동영상)을 끈다.
        {
           ARPanel.SetActive(false);
           return;
        }
     }
  }
```

5.

Q&A