

# SKRIPSI

VIRTUAL JOGGING APP UNTUK GOOGLE CARDBOARD



RICHARD WIJAYA

NPM: 2016730014

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2020



**UNDERGRADUATE THESIS**

**VIRTUAL JOGGING APP FOR GOOGLE CARDBOARD**



**RICHARD WIJAYA**

**NPM: 2016730014**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2020**



# **LEMBAR PENGESAHAN**

**VIRTUAL JOGGING APP UNTUK GOOGLE CARDBOARD**

**RICHARD WIJAYA**

**NPM: 2016730014**

Bandung, «**tanggal**» «**bulan**» 2020

Menyetujui,

Pembimbing

**Pascal Alfadian, M.Comp.**

**Ketua Tim Penguji**

**Anggota Tim Penguji**

**Chandra Wijaya, M.T.**

**Natalia, M.Si.**

Mengetahui,

Ketua Program Studi

**Mariskha Tri Adithia, P.D.Eng**



## **PERNYATAAN**

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### ***VIRTUAL JOGGING APP UNTUK GOOGLE CARDBOARD***

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal «tanggal» «bulan» 2020

Meterai Rp. 6000
---------------------

RICHARD WIJAYA  
NPM: 2016730014



## ABSTRAK

Olahraga menjadi salah satu kebutuhan manusia, tetapi banyak orang memiliki kendala seperti rasa malas dan jemuhan. Rasa malas dan jemuhan ini bahkan membuat orang tidak melakukan aktivitas olahraga paling sederhana, yaitu berlari. Jika ada aplikasi yang dapat membawa pemandangan di luar ke dalam ruangan selama berlari, banyak orang dapat ingin melakukannya karena lebih terasa lebih menyenangkan.

Untuk menampilkan suatu dunia di tepat di depan mata, *Google VR* dan *Google Cardboard* dapat dimanfaatkan. Untuk mendapatkan gambar pemandangan di luar, *Google StreetView API* digunakan untuk memperoleh gambar lokasi tertentu. Rute perjalanan dari pelari diperoleh dari *Google Directions API*, lalu perubahan pemandangan diatur oleh sensor gerak, sesuai dengan langkah kaki pengguna. Dengan menyatukan semuanya, aplikasi dapat membawa lingkungan lari yang ada di luar ruangan ke dalam ruangan.

Pengujian dilakukan pada tiga perangkat dengan spesifikasi berbeda, dengan himpunan masukan dan hasil yang diharapkan. Aplikasi dapat dijalankan dengan sempurna pada aplikasi tertentu, dapat dijalankan dengan masukan tertentu pada perangkat lain, dan tidak dapat dijalankan dalam perangkat yang terakhir. Laju dari jarak tempuh pengguna dalam aplikasi dapat terlihat dengan baik pada perangkat yang dapat menjalankan aplikasi.

**Kata-kata kunci:** Google VR, *Google Cardboard*, *Google StreetView API*, *Google Directions API*, sensor *step detector*



## ABSTRACT

Exercising has been a need of a human being, people finds that monotonous and people are lazy doing it. The laziness and the monotonous feeling prevents people from doing even the simplest form of exercise, which is jogging. If an mobile application can bring the outdoor environment while running indoor, a lot of people may be encouraged to exercise because it feels fun.

To display a world in front of the eyes, Google VR and Google Cardboard can be used. To get the picture of outdoor scenery, Google StreetView API can be used to get picture of a specific location. The jogging route can be acquired from the Google Directions API, then the change of the scenery is managed by motion sensor, according to the user's footstep. By integrating all of these, the mobile application can bring the outdoor jogging environment indoor

Testing is done on three devices with different specifications, by preparing a set of inputs and expected results. The application can be run perfectly on one device, with some inputs on another, and cannot be run at all on the last one. The distance that the user covered in the application can be observed well enough on the devices that can run the application.

**Keywords:** Google VR, Google Cardboard, Google StreetView API, Google Directions API, step detector sensor



*Teknik Informatika UNPAR*



## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena telah memberikan anugerah dan berkat-Nya kepada penulis untuk menyelesaikan skripsi dengan judul ***Virtual Jogging App untuk Google Cardboard*** dengan cukup baik. Penulis berterimakasih kepada pihak-pihak yang mendukung dan membantu agar skripsi dapat diselesaikan, di antaranya:

1. Keluarga yang mendoakan dan mendukung penulis secara jasmani dan rohani.
2. Bapak Pascal Alfadian selaku dosen pembimbing yang telah membimbing penulis sampai penulis menyelesaikan skripsi ini.
3. Bapak Chandra Wijaya dan Ibu Natalia selaku tim penguji yang telah membantu menguji dan memperbaiki skripsi ini.
4. Teman-teman Teknik Informatika UNPAR angkatan 2016 yang telah belajar dan berbagi kepada penulis.
5. Pihak-pihak lain yang belum disebutkan, yang membantu penulis menyelesaikan skripsi ini.

Akhir kata, penulis berharap agar skripsi ini berguna bagi pembaca, baik untuk kebutuhan penelitian maupun kebutuhan positif yang lain.

Bandung, «bulan» 2020

Penulis



## DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR TABEL</b>	<b>xxi</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	1
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi Penelitian . . . . .	2
1.6 Sistematika Pembahasan . . . . .	2
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 Google VR . . . . .	5
2.1.1 Google VR SDK . . . . .	5
2.1.2 Aplikasi HelloVR . . . . .	9
2.2 Google <i>StreetView API</i> . . . . .	13
2.2.1 <i>API Key</i> . . . . .	13
2.2.2 Penggunaan <i>StreetView API</i> . . . . .	13
2.2.3 Atribut Parameter <i>StreetView API</i> . . . . .	14
2.3 Google <i>Directions API</i> . . . . .	15
2.3.1 Penggunaan <i>Directions API</i> . . . . .	16
2.3.2 Hasil Pemanggilan <i>Directions API</i> . . . . .	17
2.4 <i>Motion Sensor</i> . . . . .	22
2.4.1 Deskripsi <i>Motion Sensor</i> . . . . .	22
2.4.2 Deskripsi <i>Step Detector</i> . . . . .	22
<b>3 ANALISIS</b>	<b>23</b>
3.1 Masalah yang akan Diselesaikan . . . . .	23
3.2 Analisis Google VR SDK . . . . .	23
3.3 Analisis Pembuatan Bangun Ruang Tiga Dimensi . . . . .	24
3.4 Menampilkan Gambar <i>StreetView API</i> pada Bangun Ruang Silinder . . . . .	24
3.5 Analisis Google <i>Directions API</i> . . . . .	25
3.5.1 Menentukan Atribut yang akan Digunakan dari JSON <i>Directions</i> . . . . .	25
3.5.2 Cara Memanfaatkan Atribut . . . . .	28
3.6 Analisis <i>Step Detector Sensor</i> . . . . .	29
3.6.1 Eksperimen Pengujian <i>Step Detector Sensor</i> . . . . .	29
3.6.2 Cara Memanfaatkan <i>Step Detector Sensor</i> . . . . .	29

<b>4 RANCANGAN</b>	<b>31</b>
4.1 Rancangan Antarmuka . . . . .	31
4.1.1 <i>Activity</i> Utama . . . . .	31
4.1.2 <i>Activity</i> VR . . . . .	31
4.2 Rancangan Program . . . . .	32
4.2.1 Rancangan Kelas . . . . .	32
4.2.2 Algoritma-Algoritma yang Digunakan . . . . .	35
<b>5 IMPLEMENTASI DAN PENGUJIAN</b>	<b>41</b>
5.1 Implementasi . . . . .	41
5.1.1 Lingkungan Implementasi . . . . .	41
5.1.2 Hasil Implementasi . . . . .	41
5.1.3 Tampilan Ketika Aplikasi Pertama Kali Dibuka . . . . .	41
5.1.4 Aplikasi Setelah Pengguna Menyelesaikan Perjalannya . . . . .	44
5.2 Pengujian . . . . .	44
5.2.1 Pengujian Fungsional . . . . .	46
5.2.2 Pengujian Eksperimental . . . . .	46
5.3 Masalah yang Dihadapi . . . . .	50
<b>6 KESIMPULAN DAN SARAN</b>	<b>53</b>
6.1 Kesimpulan . . . . .	53
6.2 Saran . . . . .	53
<b>DAFTAR REFERENSI</b>	<b>55</b>
<b>A KODE PROGRAM APLIKASI PENGUJIAN <i>Step Detector</i></b>	<b>57</b>
<b>B KODE PROGRAM APLIKASI <i>Jogging</i></b>	<b>59</b>

## DAFTAR GAMBAR

2.1	Struktur Direktori Google VR SDK . . . . .	6
2.2	Tampilan <i>UI</i> permainan <i>treasure hunt</i> pada aplikasi HelloVR . . . . .	9
2.3	Isi <i>folder assets</i> aplikasi HelloVR . . . . .	10
2.4	Struktur Direktori Aplikasi HelloVR . . . . .	10
2.5	Gambar-gambar <i>assets</i> aplikasi HelloVR . . . . .	11
2.6	Tampilan <i>UI Google Cloud</i> saat mengakses <i>API Key (API Key disamaraskan)</i> . . . . .	13
2.7	Pemanggilan <i>StreetView API</i> yang berhasil . . . . .	15
3.1	Tampilan <i>UI Blender</i> Blender versi 2.81 . . . . .	24
3.2	Gambar dari <i>StreetView API</i> berukuran $600 \times 300$ dengan parameter <i>heading</i> yang berbeda-beda . . . . .	25
3.3	Contoh hasil penggabungan empat gambar <i>StreetView</i> . . . . .	26
3.4	Sumbu Sensor Perangkat Bergerak . . . . .	29
3.5	<i>Log console</i> dari aplikasi perangkat bergerak pengujian <i>step detector sensor</i> ketika mendeteksi rangsang . . . . .	30
4.1	Rancangan <i>Activity</i> utama aplikasi . . . . .	37
4.2	Rancangan <i>Activity</i> VR . . . . .	37
4.3	Diagram <i>class</i> dari Aplikasi . . . . .	38
4.4	<i>Flowchart</i> dari Proses Memperoleh dan Menyatukan Gambar dari <i>StreetView API</i> . . . . .	39
5.1	<i>Activity</i> utama aplikasi . . . . .	42
5.2	Ilustrasi Pengguna saat Memasukkan Masukan Lokasi Asal dan Tujuan . . . . .	42
5.3	Tampilan Aplikasi ketika tombol " <i>Start Running</i> " Ditekan saat <i>textbox origin</i> atau <i>destination</i> kosong . . . . .	43
5.4	Tampilan <i>Google Cardboard</i> sebelum gawai diputar . . . . .	44
5.5	Ilustrasi Pengguna saat Melihat Tampilan Aplikasi Tersebut . . . . .	44
5.6	Tampilan VR saat Pengguna Berlari . . . . .	45
5.7	Ilustrasi Pengguna saat Berlari . . . . .	45
5.8	Tampilan VR saat Pengguna Mencapai Tujuan . . . . .	45
5.9	<i>Log console</i> yang Menampilkan Jarak Tempuh Pengguna pada Perangkat 1 . . . . .	51
5.10	<i>Log console</i> yang Menampilkan Jarak Tempuh Pengguna pada Perangkat 2 . . . . .	52



## **DAFTAR TABEL**

3.1 Hasil Eksperimen Pengujian <i>Step Detector Sensor</i> . . . . .	30
5.1 Daftar Perangkat yang digunakan dalam Pengujian . . . . .	46
5.2 Himpunan Masukan untuk Pengujian Aplikasi . . . . .	47
5.3 Hasil Pengujian pada Perangkat 1 . . . . .	48
5.4 Hasil Pengujian pada Perangkat 2 . . . . .	49
5.5 Hasil Pengujian pada Perangkat 3 . . . . .	50



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Di zaman modern, ada banyak bidang profesi yang ditekuni masyarakat di berbagai negara di dunia. Mayoritas orang menekuni bidang-bidang profesi yang tak terhitung banyaknya sehingga menyebabkan kesulitan pengaturan waktu untuk berolahraga, yang merupakan salah satu kebutuhan manusia untuk menjaga kesehatan. Salah satu aktivitas olahraga yang paling mudah dan tidak memerlukan gerakan yang sulit adalah berlari. Metode dilakukannya olahraga ini berkembang, mulai dari dilakukan di luar ruangan hingga dilakukan di rumah sejak ditemukannya *treadmill*. Dua metode ini memiliki kelebihan dan kekurangannya masing-masing. Berlari di luar ruangan memberikan suasana dinamis dan tidak membosankan saat melakukannya, tetapi pelaku kegiatan ini harus berada di tempat dengan lingkungan eksternal yang aman seperti ketidakhadiran kendaraan yang bergerak dan udara yang rendah polusi. Di sisi yang lain, berlari menggunakan *treadmill* bisa dilakukan di dalam ruangan sehingga masalah terganggu oleh kendaraan dan polusi udara, tetapi lingkungan sekitar saat berlari monoton sehingga membuat pelaku kegiatan berlari bosan dan jemu. Bila suasana dunia luar dapat dibawa ke dalam rumah saat menggunakan *treadmill*, aktivitas berlari menggunakan *treadmill* dapat terasa menyenangkan.

Untuk memungkinkan menampilkan pemandangan tepat di depan mata pelari, teknologi *virtual reality* (VR), teknologi yang membuat pengguna merasa berada dalam lingkungan maya tertentu yang biasanya ada pada perangkat bergerak, dapat digunakan [1]. Google VR adalah teknologi VR yang sudah umum digunakan dengan *cost* yang cukup bersahabat, terutama dalam hal *viewer*, alat yang digunakan untuk melihat pemandangan VR, yang menggunakan kardus. *Viewer* itu adalah *Google Cardboard*, *VR viewer* yang dirancang Google untuk melihat pemandangan VR.

Untuk menampilkan gambar dan rute perjalanan, diperlukan *application programming interface* (API), yang merupakan antarmuka yang menghubungkan dua atau lebih perangkat lunak. API yang dapat dimanfaatkan untuk membentuk aplikasi ini adalah *Google StreetView API* yang berfungsi untuk menampilkan gambar dari suatu lokasi tertentu dari satu arah pandang dan *Google Directions API* yang digunakan untuk mendapatkan rute perjalanan antara dua lokasi [2] [3]. Agar pemandangan yang ditampilkan dapat berubah sesuai dengan langkah kaki saat berlari, sensor gerak pada perangkat bergerak dapat dimanfaatkan [4].

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat menampilkan pemandangan saat berlari pada lingkungan yang diinginkan saat berlari di *treadmill*, memanfaatkan beberapa hal seperti *Google VR*, *Google StreetView API*, *Google Directions API*, dan sensor gerak. Dengan menggunakan perangkat lunak tersebut, orang yang berlari dapat menikmati pemandangan yang dipilih saat berlari di dalam rumah sehingga merasa seperti berlari di lingkungan yang dipilih tersebut.

### 1.2 Rumusan Masalah

Rumusan masalah yang ada pada skripsi ini adalah:

- Bagaimana memanfaatkan Google VR SDK for Android untuk menampilkan gambar dengan perangkat VR?
- Bagaimana menampilkan hasil dari *Google StreetView API* dalam bentuk VR?
- Bagaimana mengintegrasikan *Google Directions API*, gambar *Google StreetView*, sensor gerak, dan Google VR dalam perangkat lunak *virtual jogging*?

### 1.3 Tujuan

Pada skripsi ini, hal-hal yang coba untuk dicapai adalah:

- Menggunakan Google VR SDK for Android untuk menampilkan gambar dengan *Google Cardboard*.
- Menampilkan hasil gambar dari *Google StreetView API* pada *Google Cardboard*.
- Mengintegrasikan *Google Directions API*, gambar dari *Google StreetView*, sensor gerak, dan Google VR (*Cardboard*) dalam perangkat lunak *virtual jogging*.

### 1.4 Batasan Masalah

1. Aplikasi dapat menampilkan lingkungan dari lokasi asal dan tujuan yang dapat memiliki jalur darat.
2. Aplikasi dapat berfungsi pada perangkat dengan *random access memory* (RAM) minimal 4 GB.

### 1.5 Metodologi Penelitian

Metodologi penelitian yang akan digunakan adalah sebagai berikut:

- Melakukan studi literatur dari situs-situs web tentang Google VR SDK, *StreetView API*, *Directions*, sensor tentang langkah, baik melalui media tulisan maupun video.
- Menampilkan pemandangan *StreetView* pada *Google Cardboard*.
- Mengintegrasikan *Google Directions API* dengan pemandangan *StreetView* yang telah ditampilkan pada *Google Cardboard*.
- Menganalisis sensor langkah dan menyinkornisasikannya dengan perubahan pemandangan *StreetView*.

Setelah mempelajari semua komponen dari aplikasi yang akan dibuat, peneliti akan melakukan implementasi.

### 1.6 Sistematika Pembahasan

Dokumen dibagi ke dalam beberapa bab dengan sistematika pembahasan sebagai berikut:

1. Bab 1: Pendahuluan, yang menjelaskan gambaran umum penelitian. Mengandung latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, serta sistematika pembahasan.

2. Bab 2: Dasar Teori, berisi landasan dari teori-teori yang berhubungan serta mendukung penelitian. Mengandung Google VR, Google *StreetView API*, Google *Directions API*, dan sensor.
3. Bab 3: Analisis, menjelaskan mengenai proses analisis masalah untuk menemukan solusi untuk menyelesaikan masalah. Mengandung cara membuat dunia VR, cara memanfaatkan *Google StreetView*, cara memanfaatkan *Google Directions API*, dan cara memanfaatkan sensor *step-detector*.
4. Bab 4: Rancangan, menjelaskan tentang rancangan antarmuka dan rancangan program dari aplikasi.
5. Bab 5: Implementasi dan Pengujian, menjelaskan tentang hasil implementasi, hasil pengujian aplikasi, serta masalah-masalah yang dihadapi saat implementasi.
6. Bab 6: Kesimpulan dan Saran, menjelaskan kesimpulan yang diperoleh dari penelitian serta saran untuk penelitian selanjutnya.



## BAB 2

### LANDASAN TEORI

Pada bab ini akan dijelaskan mengenai Google VR, Google StreetView API, Google Directions API, dan *motion sensor*.

#### 2.1 Google VR

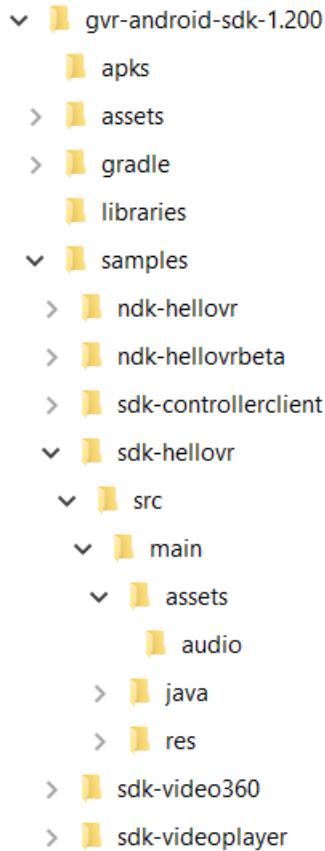
*Virtual reality* (VR) adalah teknologi yang menyajikan tampilan sebuah dunia maya yang diciptakan yang membuat pengguna merasa ada di dalam dunia tersebut menggunakan *smartphone* dan alat yang bernama *viewer*. *Viewer* adalah alat untuk melihat dunia VR pada aplikasi VR di *smartphone*. Pengguna harus memasukkan *smartphone* ke dalam *viewer* untuk dapat merasakan sensasi berada dalam dunia maya tersebut.

Teknologi VR ini telah diimplementasikan oleh Google yang disebut dengan Google VR. Google menciptakan dua jenis VR *viewer*: *Google Daydream* dan *Cardboard* [1]. Perbedaan dari *Google Daydream* dan *Google Cardboard* adalah pada bahan dan alat penerima masukan (*input*). *Google Daydream* memiliki alat penerima masukan pengguna seperti *remote control* dengan beberapa tombol serta berbahan utama plastik, sedangkan *Google Cardboard* memiliki penerima masukan berupa satu tombol (biasanya berupa magnet) yang akan memicu terjadinya suatu *event* dan berbahan dasar kardus. Google menyediakan sebuah alat bantu bagi pengembang perangkat lunak untuk memudahkan pengembangan aplikasi VR yang disebut Google VR SDK.

##### 2.1.1 Google VR SDK

Google VR SDK adalah alat bantu berlisensi Apache 2.0 yang disediakan Google pada *Github repository* yang berisi kode program dari aplikasi *virtual reality* (VR) yang tersedia untuk Android (Java), Android NDK, Unity, dan iOS [5]. Secara umum, SDK ini dibuat agar pengembang perangkat lunak dapat mempelajari serta memanfaatkan teknologi VR yang disediakan Google [1]. Ada beberapa aplikasi yang tersedia pada SDK tersebut seperti aplikasi demo bernama HelloVR dan pemutar video dalam VR, tetapi penulis akan memanfaatkan bagian aplikasi demo HelloVR untuk Android Java pada Google VR SDK. Untuk menggunakan SDK ini, dibutuhkan perangkat lunak Android Studio 2.3.3 dan lebih tinggi, dengan Android SDK versi 7.1.1 (API Level 25) atau lebih tinggi. *Folder* yang paling penting di dalam Google VR SDK adalah *samples*, yang merupakan contoh-contoh program yang memanfaatkan teknologi Google VR. Di dalam *folder* tersebut, terkandung beberapa *folder* seperti *ndk-hellovr*, *ndk-hellovrbeta*, *sdk-controllerclient*, *sdk-hellovr,sdk-video360*, dan *sdk-videoplayer*. *Folder* *sdk-hellovr* dan *ndk-hellovr* merupakan aplikasi permainan VR. Perbedaannya adalah *sdk-hellovr* ditulis murni dalam bahasa Java, sedangkan *ndk-hellovr* menggunakan bahasa C++ juga. *Folder* *sdk-controllerclient* adalah aplikasi yang menunjukkan bentuk yang berputar sesuai rangsang dari *gyroscope*. *Folder* *sdk-videoplayer* dan *sdk-video360* adalah aplikasi pemutar video 360. Gambar 2.1 menunjukkan struktur *folder* dari Google VR SDK untuk Android.

Google menyediakan *library* untuk bahasa pemrograman Java pada *package com.google.vr.sdk.base* agar dapat digunakan pengembang perangkat lunak untuk membuat aplikasi VR [6]. *Library* ini



Gambar 2.1: Struktur Direktori Google VR SDK

juga memanfaatkan *OpenGL*, yaitu *API* untuk mengolah grafika komputer. Berikut adalah beberapa *class* dan *interface* dari *package com.google.vr.sdk.base*:

#### 1. GvrView.StereoRenderere

*Interface* untuk *renderer* yang menyerahkan seluruh penyajian pemandangan stereo pada pemandangan (*view*). *Method-emethod* abstrak yang dimiliki *interface* ini adalah:

- **public abstract void onDrawEye (Eye eye)**  
*Method* yang menggambar pemandangan untuk satu mata. *Method* ini tidak memiliki nilai yang dikembalikan ataupun *exception*.
- **public abstract void onFinishFrame (Viewport viewport)** *Method* yang dipanggil sebelum *frame* selesai dibuat. Parameter yang dimiliki *method* ini adalah:
  - **Viewport viewport**: *Object* *viewport* yang dari *GL surface*.  
*Method* ini tidak memiliki nilai yang dikembalikan ataupun *exception*.
- **public abstract void onNewFrame (HeadTransform headTransform)**  
*Method* untuk menggambar perubahan *frame* dari pemandangan. Parameter yang dimiliki *method* ini adalah:
  - **HeadTransform headTransform**: *Object* dari kelas *headtransform*, yang mendefinisikan perubahan posisi kepala pengguna.  
*Method* ini tidak memiliki nilai yang dikembalikan, juga tidak memiliki *exception*.
- **public abstract void onRendererShutdown()**  
*Method* yang dipanggil ketika *thread renderer* dari pemandangan berhenti atau dimatikan, ketika method *onSurfaceCreated* dipanggil. *Method* ini tidak memiliki parameter, nilai yang dikembalikan, maupun *exception*.

- **public abstract void onSurfaceChanged (int width, int height)**  
*Method* yang terpanggil ketika ada perubahan dimensi pada permukaan dunia VR. Parameter yang dimiliki *method* ini adalah:
  - **int width:** Nilai dari lebar pemandangan satu *eye* dalam satuan *pixel*.
  - **int height:** Nilai dari tinggi pemandangan satu *eye* dalam satuan *pixel*.
- **public abstract void onSurfaceCreated (EGLConfig config)**  
*Method* untuk membuat dunia VR. Parameter yang dimiliki *method* ini adalah:
  - **EGLConfig config:** Konfigurasi EGL yang digunakan untuk membuat permukaan dunia VR.

## 2. AndroidCompat

*Utility class* yang disediakan Android untuk menggunakan fitur-fitur VR.

- **public static void setSustainedPerformanceMode (Activity activity, boolean enabled)**  
*Method* yang mengatur `android.view.Window` untuk menjaga performanya agar tetap stabil.
- **public static boolean setVrModeEnabled (Activity activity, boolean enabled)**  
*Method* ini menyetel pengaturan VR yang sesuai untuk suatu *Activity*. Parameter yang dimiliki *method* ini adalah:
  - **Activity activity**  
*Activity* yang akan disetel dengan mode VR.
  - **boolean enabled**  
Nilai *boolean* sesuai dengan apakah mode VR akan diaktifkan atau tidak.

## 3. Eye

*Class* yang mendefinisikan rincian *rendering* stereo dari satu *eye*, yang merupakan tampilan dunia VR dalam satu mata.

- **public float[] getEyeView()**  
*Method* untuk menghasilkan matriks dari kamera ke *eye*.
- **public FieldOfView getFov()**  
*Method* yang mengembalikan pemandangan untuk satu *eye*.
- **public float[] getPerspective (float zNear, float zFar)**  
*Method* untuk mengembalikan matriks proyeksi dari sudut pandang untuk *eye* tertentu.

## 4. GvrActivity

*Activity* dasar yang mudah diintegrasikan dengan perangkat Google VR.

- **public void onBackPressed()**  
*Method* yang dipanggil ketika tombol kembali ditekan.
- **public void onCardboardTrigger()**  
*Method* yang dipanggil ketika pemicu *Cardboard* ditekan.
- **public void setGvrView(GvrView gvrView)**  
*Method* untuk menentukan *GvrView* pada *activity*. Parameter yang diterima oleh *method* ini adalah:
  - **GvrView gvrView**  
Objek *GvrView* yang akan digunakan *activity*.

## 5. GvrView

*Class* dari *view* yang mendukung VR *rendering* dari Google. *Method-method* yang dimiliki *class* ini adalah:

- **public void setTransitionViewEnabled(boolean enabled)**  
*Method* yang menentukan apakah *view* transisi yang menginformasikan pengguna untuk memasukkan gawai ke dalam VR *viewer* dimunculkan atau tidak. Parameter yang dimiliki *method* ini adalah:
  - **boolean enabled**  
 Nilai *boolean* yang menyatakan apakah *view* transisi dimunculkan atau tidak.
- **public void setRenderer(GvrView.StereoRenderer renderer)**  
*Method* yang menyatakan *StereoRenderer* dari objek *GvrView*.
- **public void enableCardboardTriggerEmulation()**  
*Method* yang membuat *Daydream headset* menjadi mampu memberikan masukan *Cardboard trigger*.

## 6. HeadTransform

*Class* yang mendefinisikan perubahan posisi kepala pengguna terhadap lingkungan VR.

- **public void getHeadView (float[] headView, int offset)** *Method* yang mengacu pada matriks transformasi dari *camera* ke kepala. Parameter yang dimiliki *method* ini adalah:
  - **float[] headView** Matriks  $4 \times 4$  yang menyatakan matriks transformasi dari *camera* menuju kepala.
  - **int offset**  
*Offset* dari *array* tempat data disimpan.
- **public void getQuaternion (float[] quaternion, int offset).**  
*Method* yang mengacu pada matriks *quaternion* yang merepresentasikan transformasi dari putaran kepala. Parameter yang dimiliki *method* ini adalah:
  - **float[] quaternion**  
 Matriks quartenion yang menyatakan transformasi.
  - **int offset** *Offset* dari *array* tempat data disimpan.

## 7. Viewport

*Class* yang mendefinisikan *viewport* berbentuk persegi panjang. *Viewport* merupakan area untuk menampilkan objek grafis.

Atribut-atribut yang dimiliki kelas ini adalah sebagai berikut:

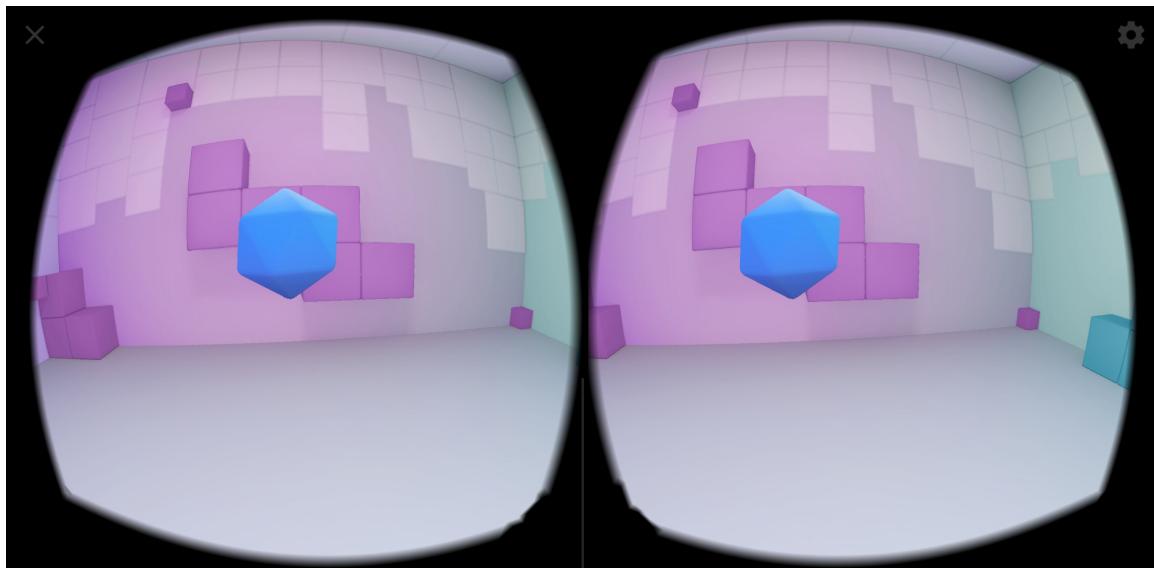
- **public int height**  
*Tinggi* dari *viewport*.
- **public int width** Lebar dari *viewport*.
- **public int x**  
*Koordinat sumbu x* titik pada *viewport*, dengan koordinat (0,0) berada pada sudut kiri bawah.
- **public int y**  
*Koordinat sumbu y* titik pada *viewport*, dengan koordinat (0,0) berada pada sudut kiri bawah.

*Method-method* yang dimiliki kelas ini adalah sebagai berikut:

- `public void setGLViewport()`  
*Method* untuk menentukan *viewport* dari OpenGL.
- `public void setViewport(int x, int y, int width, int height)`  
*Setter* dari objek *viewport* sesuai parameter posisi dan dimensinya. Parameter yang dimiliki *method* ini:
  - `int x`  
 Koordinat *viewport* dari sumbu x.
  - `int y`  
 Koordinat *viewport* dari sumbu y.
  - `int width`  
 Tinggi dari *viewport*
  - `int height`  
 Lebar dari *viewport*.

### 2.1.2 Aplikasi HelloVR

HelloVR, aplikasi yang diperoleh dari Google VR SDK pada direktori `./samples/sdk-hellovr`, adalah sebuah aplikasi demo permainan *treasure hunt*, yaitu sejenis permainan mencari bentuk yang mengapung di dunia VR dengan melihat tepat pada bentuk tersebut dan menyalakan pemicu pada Google Cardboard. Setelah kondisi untuk menangkap bentuk yang ada, bentuk tersebut akan menghilang, lalu bentuk yang lain akan muncul di tempat lain. Gambar 2.2 menunjukkan tampilan UI permainan *treasure hunt* HelloVR.

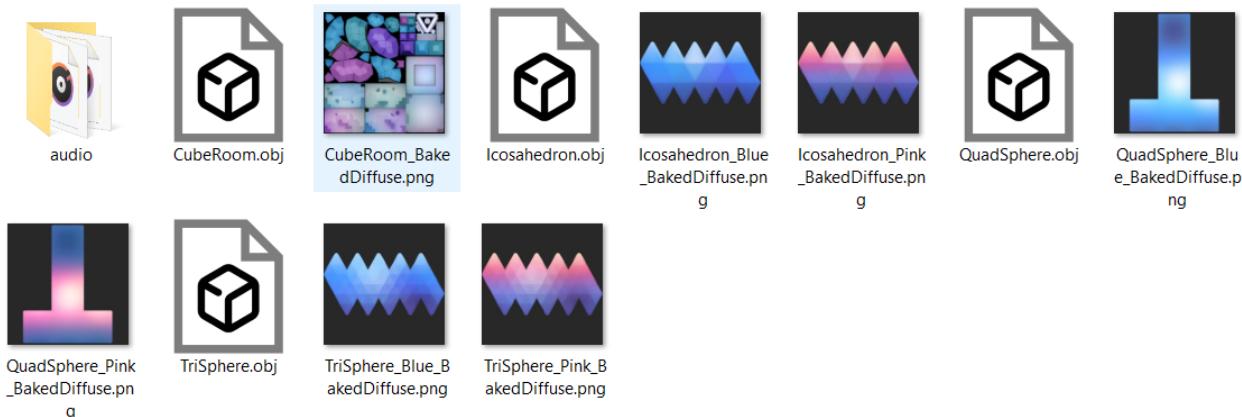


Gambar 2.2: Tampilan UI permainan *treasure hunt* pada aplikasi HelloVR

### Komponen Aplikasi HelloVR

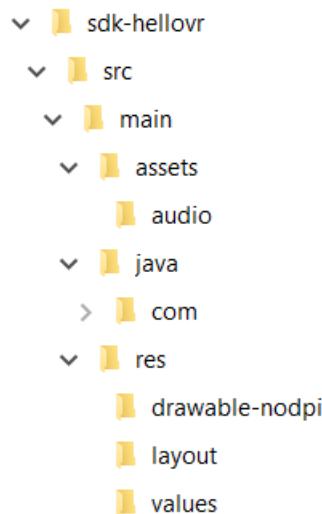
Aplikasi HelloVR terdiri dari beberapa *folder* dan komponen-komponen. Gambar 2.4 menunjukkan struktur direktori aplikasi HelloVR. Ada beberapa *folder* penting yang terkandung dalam program, yaitu *assets*, *java*, dan *res*. *Folder assets* mengandung *asset-asset* audiovisual yang akan digunakan dalam aplikasi seperti gambar, suara, atau video, *folder java* mengandung *file-file* kode program, sementara *folder res* mengandung komponen-komponen dari antarmuka aplikasi seperti gambar

logo aplikasi, gambar latar belakang tampilan aplikasi, sampai nilai-nilai alfabetik seperti nama aplikasi. Gambar 2.3 menunjukkan rincian isi *folder assets* dari aplikasi HelloVR.



Gambar 2.3: Isi *folder assets* aplikasi HelloVR

Dunia VR pada aplikasi ini terbentuk dari komponen-komponen yang ada dalam *folder assets*. Bentuk ruangan VR ditentukan oleh *file Wavefront Object* (OBJ) dan tampilannya oleh tekstur *file Portable Network Graphics* (PNG) (Gambar 2.5a) yang telah dengan sangat tepat dipetakan pada *file OBJ* yang ada sehingga dunia VR terlihat sangat nyata. Bentuk-bentuk yang akan dicari pengguna dibuat dari tiga file OBJ yang merepresentasikan tiga macam bentuk yang akan muncul. Masing-masing file OBJ memiliki dua tekstur yang telah dipetakan pada masing-masing file OBJ dalam file PNG. Satu tekstur (Gambar 2.5b) digunakan ketika bentuk sedang tidak ada di tengah-tengah titik pengelihatan pengguna, sedangkan satu tekstur yang lain (Gambar 2.5c) digunakan ketika pengguna sedang melihat bentuk tepat di titik tengah pengelihatan pengguna.



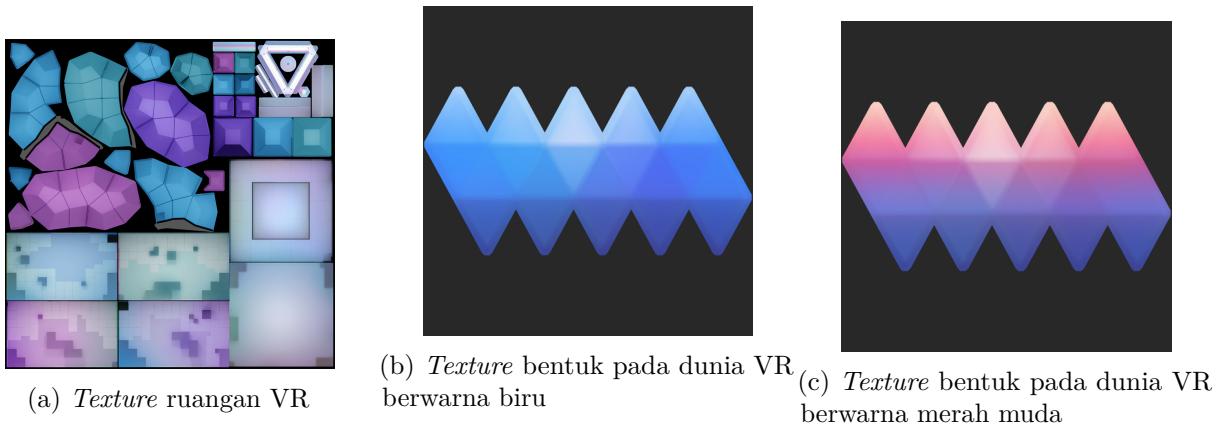
Gambar 2.4: Struktur Direktori Aplikasi HelloVR

## Rancangan kelas Aplikasi HelloVR

Aplikasi HelloVR memiliki empat kelas pada programnya, di antaranya:

### 1. Texture

Kelas yang memuat tekstur yang akan digunakan. Atribut yang dimiliki kelas ini adalah:



Gambar 2.5: Gambar-gambar *assets* aplikasi HelloVR

- `int[] textureId` - Atribut yang menyimpan representasi tekstur ruangan yang dapat digunakan dalam kode program.

*Method-method* yang dimiliki kelas ini di antaranya:

- `public void bind()`

*Method* ini adalah *method* mengikat tekstur ke `GL_TEXTURE0` dari `GLES20`, yang adalah penyaji (*renderer*) dari dunia VR.

### 2. TexturedMesh

Kelas ini memuat sebuah bentuk tiga dimensi yang sudah diberi tekstur sehingga terlihat indah dan berwarna. Atribut-atribut yang dimiliki kelas ini adalah:

- `private final FloatBuffer vertices`

Atribut ini adalah atribut dari sudut-sudut ruang tiga dimensi.

- `private final FloatBuffer uv`

Atribut ini adalah atribut dari koordinat tekstur yang digunakan.

- `private final ShortBuffer indices`

Atribut ini adalah atribut indeks sudut-sudut dari permukaan ruang tiga dimensi.

- `private final int positionAttrib`

Atribut ini adalah atribut dari posisi ruang tiga dimensi pada *shader*.

- `private final int uvAttrib`

Atribut ini adalah atribut dari koordinat tekstur pada *shader*. *Shader* adalah program yang mewarnai ruang.

Method yang dimiliki kelas ini adalah: `public void draw()`

Method untuk menggambar ruang dengan tekstur.

### 3. Util

Kelas yang digunakan untuk menghitung vektor dan sudut yang dibentuk antara mata pengguna dan bentuk yang akan dicari, serta mengatur pengaturan yang tepat untuk *OpenGL*, yang adalah *renderer* yang digunakan untuk menggambar bentuk dan ruangan. Atribut-atribut yang dimiliki kelas ini adalah:

- `private static final boolean HALT_ON_GL_ERROR`

Atribut ini menentukan apakah proses *build* program dihentikan jika ada masalah atau tidak.

*Method-method* yang dimiliki kelas ini di antaranya:

- `public static void checkGlError(String label)`

*Method* ini digunakan untuk menjalankan GLES20.

**Parameter:**

- `String label`

Parameter ini adalah nilai *label* yang akan diteruskan saat galat terjadi.

**Return Value:** Tidak ada

**Exception:** Tidak ada

- `public static int compileProgram(String[] vertexCode, String[] fragmentCode)`

*Method* ini digunakan untuk meng-*compile* program *shader* GLES20.

**Parameter:**

- `String[] vertexCode`

Parameter ini adalah nilai kumpulan sudut dari program *shader* GLES20

- `String[] fragmentCode`

Parameter ini adalah nilai pecahan-pecahan program *shader* GLES20.

**Return Value:** *id* dari program GLES20.

**Exception:** Tidak ada

#### 4. HelloVrActivity

Kelas ini merupakan kelas *activity* Google VR. Berikut adalah diagram kelas untuk memperjelas hubungan antara semua kelas aplikasi HelloVR. Kelas ini akan menggunakan tiga kelas lainnya untuk mendapat ruangan dan bentuk yang akan digambar, serta keadaan (*state*) dari permainan, seperti sedang menatap pada bentuk atau tidak dan bagian ruangan yang sedang dilihat. Atribut-atribut yang dimiliki kelas ini adalah sebagai berikut:

- `private float[] camera`

Atribut *camera* yang direpresentasikan dengan kumpulan bilangan *float*.

- `private int objectProgram`

Atribut dari *id* program *shader*.

- `private int objectPositionParam`

Atribut dari *id* parameter posisi dari objek-objek dalam dunia tiga dimensi.

- `private int objectUvParam`

Atribut dari *id* parameter koordinat tekstur dari objek-objek dunia tiga dimensi.

- `private int objectModelViewProjectionParam`

Atribut dari *id* parameter model view dari objek-objek dunia tiga dimensi.

- `private TexturedMesh room`

Atribut dari ruang tiga dimensi yang telah diberi tekstur.

- `private Texture roomTex`

Atribut dari tekstur yang akan digunakan untuk ruang tiga dimensi.

- `private ArrayList<TexturedMesh> targetObjectMeshes`

Atribut yang memuat kumpulan bentuk tiga dimensi dari target.

- `private ArrayList<Texture> targetObjectNotSelectedTextures`

Atribut yang memuat tekstur dari objek yang sedang tidak dilihat.

- `private ArrayList<Texture> targetObjectSelectedTextures`

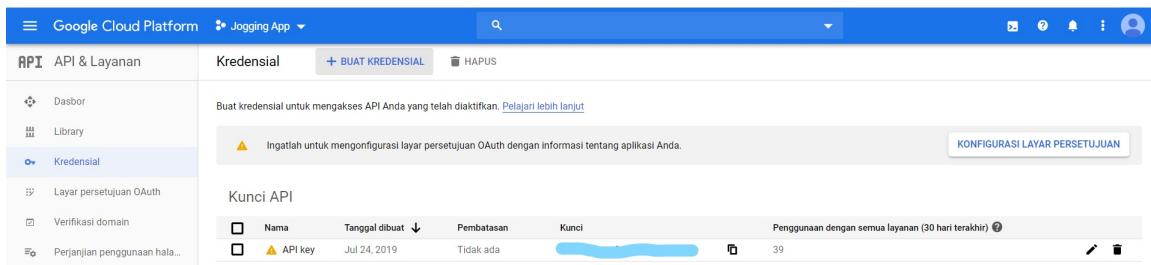
Atribut yang memuat tekstur dari objek yang sedang dilihat.

*Method-method* yang dimiliki kelas ini adalah:

- `public void initializeGvrView()`  
*Method* yang digunakan untuk menginisialisasi pemandangan VR. **Parameter:** Tidak ada  
**Return Value:** Tidak ada  
**Exception:** Tidak ada
- `public void onSurfaceCreated(EGLConfig config)`  
**Parameter:**
  - `EGLConfig config`  
*Konfigurasi* dari OpenGL yang akan digunakan. **Return Value:** Tidak ada **Exception:** Tidak ada

## 2.2 Google StreetView API

Google StreetView API adalah API yang disediakan Google untuk mendapatkan gambar dari suatu lokasi sesuai masukan pengguna melalui *HTTP request* [2]. Gambar yang dihasilkan dari pemanggilan *StreetView API* ini berbentuk persegi panjang yang merupakan gambar dari satu arah pandang. Ada dua jenis *StreetView API* yang disediakan Google, yaitu *static* dan *dynamic*. *StreetView API* yang statis akan menampilkan pemandangan yang tetap tanpa pergerakan pada pemandangannya, sedangkan yang dinamis menampilkan pemandangan yang berubah-ubah seperti *video*.



Gambar 2.6: Tampilan UI Google Cloud saat mengakses API Key (API Key disamarkan)

### 2.2.1 API Key

Agar dapat menggunakan *StreetView API* (dan *Directions API* pada Subbab 2.3), ada *API key* yang harus diperoleh pada Google Cloud Platform Console dengan memasukkan nomor kartu kredit. Gambar 2.6 menunjukkan tampilan *Google Cloud* setelah mendapatkan API key [7]. API key yang diberikan terdiri atas dua puluh dan delapan belas karakter alfanumerik (bisa huruf kapital dan huruf kecil) yang dihubungkan dengan tanda "-". API key yang telah diperoleh akan digunakan sebagai salah satu parameter masukan agar Google API dapat diakses.

### 2.2.2 Penggunaan StreetView API

Secara umum, API diakses menggunakan URL Web sebagai berikut:

`https://maps.googleapis.com/maps/api/streetview?parameters`

"Parameters" pada URL Web adalah atribut-atribut dengan parameter yang diterima StreetView. Sintaks parameter tersebut adalah:

$$X = Y$$

X adalah atribut dari StreetView, sedangkan Y adalah nilainya, dan nilai tersebut harus sesuai dengan tipe dan rentang nilai masing-masing atribut. Untuk atribut kedua dan seterusnya yang akan dimasukkan dalam parameter (jika ada), dapat diteruskan dengan tanda "&", lalu diikuti dengan pola seperti rumus di atas. Saat mengakses *StreetView API*, ada dua kemungkinan hasil yang diperoleh, yaitu berhasil dan gagal. Pemanggilan *API* yang berhasil akan menghasilkan gambar pemandangan dari lokasi sesuai masukan pengguna, sementara pemanggilan yang gagal menghasilkan sebuah gambar dengan penjelasan bahwa gambar tidak tersedia.

### 2.2.3 Atribut Parameter *StreetView API*

Untuk menampilkan pemandangan yang sesuai keinginan pengguna, beberapa parameter masukan harus ditentukan. Ada dua jenis parameter masukan, di antaranya parameter wajib dan parameter opsional. Pengaksesan atau pemanggilan *StreetView API* yang berhasil akan mengembalikan sebuah gambar pemandangan dari lokasi sesuai parameter masukan.

#### Parameter Wajib

Parameter wajib adalah parameter yang harus dimasukkan oleh pengguna dan jika tidak dimasukkan akan mengakibatkan pemanggilan yang gagal. Beberapa parameter wajib pada *StreetView API* adalah:

- *size*

Ukuran dari gambar yang dihasilkan, dalam *pixel*. Format parameter adalah: banyak *pixel* secara horizontal × banyak *pixel* secara vertikal.

- *key*

*API key* yang dijelaskan pada Subbab 2.2.1.

- *location* atau *pano* (salah satu)

Lokasi dari pemandangan yang ingin ditampilkan. *location* menerima dua jenis parameter garis lintang dan garis bujur (*longitude* dan *latitude*) atau *String* nama lokasi, sementara *pano* menerima *panorama id* dari lokasi atau panorama.

#### Parameter Opsional

Selain parameter wajib, ada parameter opsional, yaitu parameter yang tidak perlu diisi agar pengaksesan *API* berhasil dan biasanya atribut parameter tersebut sudah memiliki nilai bawaan (*default*). Ada beberapa parameter opsional yang dapat digunakan sebagai parameter untuk mengubah pengaturan dari pemandangan yang diambil:

- *signature*

Atribut untuk memastikan bahwa *request* dikirim dengan *API key* sesuai jenis *signature* yang diatur pemilik *API key*. Nilai dari *signature* bertipe alfabetik.

- *heading*

Atribut yang menyatakan arah pandangan secara horizontal, nilai atribut menyatakan sudut yang dibentuk dari arah utara dengan arah pandang yang diinginkan (sudut yang dibentuk dari arah berlawanan jarum jam), dengan rentang bilangan bulat positif.

- *fov (field of view)*

Atribut yang menyatakan seberapa perbesaran pemandangan (nilai dalam satuan derajat dengan rentang nilai 10 sampai 120).

- *pitch*

Atribut yang menyatakan pandangan pengguna secara vertikal, satuan nilai dalam derajat, dengan rentang nilai bilangan bulat dari -90 sampai 90.

- *radius*

Atribut yang menyatakan jarak dalam meter, yang adalah titik pengambilan pemandangan, dengan rentang nilai bilangan bulat positif dan nol.

- *source*

Atribut yang menyatakan jenis pemandangan, *default* atau *outdoor* (bertipe data alfabetik).

### Hasil Pemanggilan *Street View API*

Pemanggilan *StreetView* yang berhasil akan menghasilkan sebuah gambar dari pemandangan sesuai lokasi. Gambar 2.7 memperlihatkan pemanggilan *StreetView API* yang berhasil dengan URL [https://maps.googleapis.com/maps/api/streetview?size=600x300&location=-6.8746537,107.6046282&key=\(disamarkan\)](https://maps.googleapis.com/maps/api/streetview?size=600x300&location=-6.8746537,107.6046282&key=(disamarkan)).



Gambar 2.7: Pemanggilan *StreetView API* yang berhasil

Gambar yang dihasilkan merupakan gambar dari satu arah pandang dengan ukuran sesuai ukuran yang dimasukkan pengguna (jika parameter opsional tidak diisi, akan diisi dengan nilai *default*). Parameter yang digunakan untuk memperoleh Gambar 2.7 dari *StreetView API* adalah:

- *size* =  $600 \times 300$
- *location* =  $-6.8746537, 107.6046282$
- *key* disamarkan.

## 2.3 Google Directions API

Google *Directions API* adalah layanan berbasis *HTTP/HTTPS* dari Google yang membantu mencari jalur terdekat yang harus dilewati untuk pergi dari lokasi asal ke lokasi tujuan sesuai parameter

pengguna [3]. Ada beberapa *mode* dari arah yang dapat dicari seperti *driving*, *transit*, *walking*, dan *cycling*. Pengaksesan *Directions API* sangat mirip dengan *StreetView API*, yaitu membutuhkan *API Key*, seperti yang dijelaskan pada Subbab 2.2.1, sebagai salah satu atribut wajib, juga memiliki atribut wajib dan opsional yang dapat diatur lewat parameter.

### 2.3.1 Penggunaan *Directions API*

Sintaks untuk mengaksesnya pun mirip dengan *StreetView API* dengan *URL* sebagai berikut:

`https://maps.googleapis.com/maps/api/directions/filetype?parameter`

Bagian "filetype" pada *URL* diganti dengan tipe *file* keluaran atau hasil (xml atau json). Bagian "parameter" diganti dengan parameter-parameter dari *Directions API*. *Directions API* juga memiliki dua jenis parameter seperti *StreetView API*: wajib dan opsional. Parameter-parameter tersebut di antaranya:

- *origin*

Parameter wajib bertipe alfabetik yang menyatakan lokasi asal yang dimasukkan pengguna. Parameter ini diisi *string* lokasi yang *sah*.

- *destination*

Parameter wajib bertipe string alfabetik yang menyatakan lokasi yang valid dari lokasi tujuan yang dimasukkan pengguna.

- *key*

Parameter wajib yang bertipe String yang menyatakan *API key* milik pengguna (seperti pada Subbab 2.2.1).

- *mode*

Parameter opsional yang bertipe String yang menyatakan *mode* perjalanan yang akan ditempuh, seperti "*driving*", "*walking*", "*bicycling*", atau "*transit*"

- *waypoints*

Parameter opsional yang menyatakan lokasi yang ingin ditempuh dalam perjalanan. Parameter ini dapat diisi dengan beberapa jenis parameter yang menyatakan lokasi seperti garis lintang dan garis bujur dan *string* alamat lokasi.

- *alternatives*

Parameter opsional *bit (boolean)* yang menyatakan apakah rute yang disediakan *Directions API* menyediakan beberapa pilihan rute atau tidak.

- *avoid*

Parameter opsional bertipe *String* (alfabetik) yang menyatakan jenis-jenis jalan yang harus dihindari. Nilai-nilai yang sah untuk parameter ini adalah "*tolls*", "*highways*", "*ferries*", dan/atau "*indoor*" (nilai dipisahkan dengan "|" jika ada beberapa).

- *language*

Parameter opsional bertipe *string* (alfabetik) yang menyatakan bahasa yang digunakan untuk menyajikan rute perjalanan sesuai bahasa yang disediakan Google.

- *units*

Parameter opsional bertipe *String* (alfabetik) yang menyatakan jenis satuan yang akan digunakan, seperti "*metrics*" atau "*imperial*".

- *region*

Parameter opsional terdiri dari dua karakter yang menyatakan kode daerah.

- *arrival\_time*

Parameter opsional bertipe bilangan bulat yang menandakan waktu yang diinginkan untuk sampai di tujuan.

- *departure\_time*

Parameter opsional bertipe bilangan bulat yang menyatakan waktu keberangkatan yang diinginkan.

- *traffic\_model*

Parameter opsional *string* (alfabetik) yang menyatakan jenis perjalanan yang disajikan sesuai waktu tempuh, seperti perjalanan dengan waktu paling tepat ("*best\_guess*"), yang paling optimis ("*optimistic*"), dan yang paling pesimis ("*pessimistic*").

### 2.3.2 Hasil Pemanggilan *Directions API*

Hal yang dihasilkan oleh pemanggilan *Directions API* adalah *script Javascript Notation Object (JSON)* yang menyatakan arah sesuai lokasi asal dan tujuan, serta mode perjalanan yang adalah masukan pengguna [3]. Selain arah, *script JSON* yang dikembalikan juga mengandung informasi mengenai jarak jalan yang ditempuh serta waktu tempuh perjalanan. Listing 2.1 menunjukkan contoh *script JSON* dengan URL [https://maps.googleapis.com/maps/api/directions/json?origin=unpar&destination=Rumah+Sakit+Santo+Borromeus&mode=walking&key=\(disamarkan\)](https://maps.googleapis.com/maps/api/directions/json?origin=unpar&destination=Rumah+Sakit+Santo+Borromeus&mode=walking&key=(disamarkan)) (parameter *key* tidak dicantumkan). Pemanggilan yang gagal akan mengembalikan *script JSON* yang menyatakan jalan di antara dua lokasi masukan tidak dapat ditemukan.

Listing 2.1: Hasil Pemanggilan *Directions API* yang Berhasil

```
{
  "geocoded_waypoints" : [
    {
      "geocoder_status" : "OK",
      "place_id" : "ChIJbYmcEu7maC4RRijB2oKhHLA",
      "types" : [ "establishment", "point_of_interest", "university" ]
    },
    {
      "geocoder_status" : "OK",
      "place_id" : "ChIJU8k7DlHmaC4RQ2mUo1ERm1k",
      "types" : [ "establishment", "health", "hospital", "point_of_interest" ]
    }
  ],
  "routes" : [
    {
      "bounds" : {
        "northeast" : {
          "lat" : -6.8746719,
          "lng" : 107.6137497
        },
        "southwest" : {
          "lat" : -6.893148,
          "lng" : 107.6034915
        }
      }
    }
  ]
}
```

```

},
"copyrights" : "Map\u2022data\u2022c2020",
"legs" : [
{
  "distance" : {
    "text" : "3.0\u2022km",
    "value" : 2980
  },
  "duration" : {
    "text" : "35\u2022mins",
    "value" : 2116
  },
  "end_address" : "Jl.\u2022Ir.\u2022H.\u2022Juanda\u2022No.100,\u2022Lebakgede . . .",
  "end_location" : {
    "lat" : -6.893148,
    "lng" : 107.6131791
  },
  "start_address" : "Jl.\u2022Ciumbuleuit\u2022No.94,\u2022Hegarmanah . . .",
  "start_location" : {
    "lat" : -6.8746719,
    "lng" : 107.6046127
  },
  "steps" : [
    {
      "distance" : {
        "text" : "1.0\u2022km",
        "value" : 1008
      },
      "duration" : {
        "text" : "11\u2022mins",
        "value" : 667
      },
      "end_location" : {
        "lat" : -6.8833328,
        "lng" : 107.6049108
      },
      "html_instructions" : "Head\u2022\u003cb\u003esouth . . .",
      "polyline" : {
        "points" : "tu}\u003ch@yowoSdAP|AL\u003Cb@dAHHBvC . . ."
      },
      "start_location" : {
        "lat" : -6.8746719,
        "lng" : 107.6046127
      },
      "travel_mode" : "WALKING"
    },
    {
      "distance" : {
        "text" : "1.1\u2022km",
        "value" : 1078
      },
      "duration" : {
        "text" : "12\u2022mins",
        "value" : 720
      },
      "end_location" : {
        "lat" : -6.8833328,
        "lng" : 107.6049108
      },
      "html_instructions" : "Head\u2022\u003cb\u003esouth . . .",
      "polyline" : {
        "points" : "tu}\u003ch@yowoSdAP|AL\u003Cb@dAHHBvC . . ."
      },
      "start_location" : {
        "lat" : -6.8746719,
        "lng" : 107.6046127
      },
      "travel_mode" : "WALKING"
    }
  ]
}

```

```
        "duration" : {
            "text" : "14\u00a9mins",
            "value" : 834
        },
        "end_location" : {
            "lat" : -6.88521839999999,
            "lng" : 107.6137497
        },
        "html_instructions" : "Turn\u20e3\u003cb\u003eleft\u003c/b...",
        "maneuver" : "turn-left",
        "polyline" : {
            "points" : "xk_i@uqwoSCEAECKAM?K?E?..."
        },
        "start_location" : {
            "lat" : -6.8833328,
            "lng" : 107.6049108
        },
        "travel_mode" : "WALKING"
    },
    {
        "distance" : {
            "text" : "0.9\u00a9km",
            "value" : 884
        },
        "duration" : {
            "text" : "10\u00a9mins",
            "value" : 603
        },
        "end_location" : {
            "lat" : -6.89314079999999,
            "lng" : 107.6130843
        },
        "html_instructions" : "Turn\u20e3\u003cb\u003eright...",
        "maneuver" : "turn-right",
        "polyline" : {
            "points" : "rw_i@}hyoSzCLlAHz@Bd@@fB@tBDfABR@L... "
        },
        "start_location" : {
            "lat" : -6.88521839999999,
            "lng" : 107.6137497
        },
        "travel_mode" : "WALKING"
    },
    {
        "distance" : {
            "text" : "10\u00a9m",
            "value" : 10
        },
        "duration" : {
            "text" : "1\u00a9min",
            "value" : 12
        }
    }
]
```

```

        },
        "end_location" : {
            "lat" : -6.893148,
            "lng" : 107.6131791
        },
        "html_instructions" : "Turn\u003cb\u003eleft\u003c...",
        "maneuver" : "turn-left",
        "polyline" : {
            "points" : "biai@wdyoS@S"
        },
        "start_location" : {
            "lat" : -6.89314079999999,
            "lng" : 107.6130843
        },
        "travel_mode" : "WALKING"
    }
],
"traffic_speed_entry" : [],
"via_waypoint" : []
}
],
"overview_polyline" : {
    "points" : "tu}h@yowoSdAP|AL'Cb@dAH'Dd@~Bd@hG..."
},
"summary" : "Jl. Ciumbuleuit , Jl. Siliwangi ...",
"warnings" : [
    "Walking directions are in beta. Use caution ..."
],
"waypoint_order" : []
}
],
"status" : "OK"
}

```

Penjelasan mengenai atribut-atribut hasil keluaran *Directions API* adalah sebagai berikut:

- *geocoded waypoints*

Berisi kumpulan rincian lokasi asal, tujuan, dan *waypoint-waypoint* yang ditentukan pengguna (rincian termasuk *id* dan jenis tempat). Masing-masing objek dalam atribut ini memiliki atribut-atribut:

- *geocoder\_status*

Status keabsahan tempat.

- *place\_id*

*String* alfanumerik dari *id* tempat.

- *types*

*Array* dari deskripsi tempat, seperti kantor, pusat kesehatan, bangunan, dan sebagainya.

- *routes*

*Array* dari jalan-jalan yang merupakan rute yang ditempuh dari lokasi asal ke lokasi tujuan. Setiap objek dari atribut ini memiliki atribut-atribut:

- *copyrights*  
*Copyright text* yang ditunjukkan untuk suatu jalan.
  - *legs[]*  
*Array* dari satu *leg*, yang adalah objek yang mengandung informasi mengenai jalan yang ditempuh. Setiap objek *leg* memiliki atribut-atribut:
    - \* *distance*  
Panjang dari *leg* yang dilewati, dalam meter.
    - \* *duration*  
Lama waktu yang dibutuhkan untuk menempuh jalan sesuai *mode*.
    - \* *start\_address*  
Alamat dari satu titik ujung dari jalan yang ditempuh terlebih dahulu pada perjalanan.
    - \* *start\_location*  
Koordinat garis lintang dan garis bujur dari *start\_address*.
    - \* *end\_address*  
Alamat dari satu titik ujung dari jalan yang ditempuh kemudian atau terakhir pada perjalanan.
    - \* *end\_location*  
Koordinat garis lintang dan garis bujur dari *end\_address*.
    - \* *steps*  
*Array* dari objek *step*, bagian dari *legs*. Beberapa atribut yang dimiliki objek *step* adalah:
      - *distance*  
Jarak dari *step* ini.
      - *duration*  
Lama waktu yang digunakan untuk menempuh *step* ini.
      - *start\_location*  
Koordinat garis lintang dan bujur dari titik awal *step* ini.
      - *end\_location*  
Koordinat garis lintang dan bujur dari titik akhir *step* ini.
      - *polyline*  
Deretan titik di dalam *steps* yang direpresentasikan dengan kumpulan karakter.
      - *travel\_mode*  
*Mode* perjalanan yang ditempuh
  - *overview\_polyline*  
Satu objek *point* yang merupakan representasi rute.
  - *bounds*  
*Viewport* pembatas dari lokasi.
  - *summary*  
Deskripsi dari jalur seperti arah yang diambil.
  - *warnings[]*  
*Array* yang berisi peringatan saat rute perjalanan ditampilkan.
  - *waypoint\_order*  
*Array* dari urutan *waypoint* yang dilewati.
- *status*  
Berisi status dari *request* pengaksesan *Directions API*.

## 2.4 Motion Sensor

### 2.4.1 Deskripsi Motion Sensor

*Motion sensor* adalah sensor pada *smartphone* yang mendeteksi pergerakan gawai *smartphone* [4]. Pergerakan yang dapat dideteksi termasuk saat gawai dimiringkan, digoyangkan, diayunkan, atau diputar (sumber). Beberapa contoh *motion sensor* pada *smartphone* adalah:

- *accelerometer*

Sensor yang mendeteksi gerakan *smartphone* terhadap sumbu *x*, *y*, dan *z*, termasuk gaya gravitasi terhadap masing-masing sumbu.

- *gravity sensor*

Sensor yang mendeteksi gaya gravitasi terhadap sumbu *x*, *y*, dan *z*.

- *gyroscope*

Sensor yang mendeteksi putaran gawai terhadap sumbu *x*, *y*, dan *z*.

- *linear acceleration sensor*

Sensor yang mendeteksi pergerakan linier, terhadap sumbu *x*, *y*, dan *z* tanpa gaya gravitasi pada masing-masing sumbu.

- *rotation vector sensor*

Sensor yang mendeteksi vektor putaran pada gawai terhadap sumbu *x*, *y*, dan *z*.

- *step counter*

Sensor yang menghitung jumlah langkah saat berjalan yang pengguna gawai ambil.

- *step detector*

Sensor yang men-trigger sebuah *event* saat pengguna mengambil langkah saat berjalan.

Sintaks untuk menggunakan sensor ini tertera pada Listing 2.2.

Listing 2.2: Sintaks menggunakan sensor *step detector*

```
private SensorManager sensorManager;
private Sensor sensor;
...
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);
```

### 2.4.2 Deskripsi Step Detector

*Step detector* adalah sensor yang digunakan untuk mendeteksi langkah kaki pengguna. Sensor ini dapat memicu suatu *event* setiap kali langkah kaki pengguna diambil. Nilai yang dikembalikan adalah 1.0 dan *timestamp* dari saat langkah kaki diambil pengguna (nilai 1.0 menunjukkan bahwa ada langkah yang telah diambil). *Permission* yang dibutuhkan untuk mengaktifkan sensor ini adalah `android.permission.ACTIVITY_RECOGNITION`. Latensi dari sensor ini sekitar kurang dari dua detik.

## BAB 3

# ANALISIS

Pada bab ini dijelaskan mengenai analisis Google VR SDK, *Google StreetView API*, *Google Directions API*, dan sensor *step detector*, juga cara memanfaatkan mengintegrasikan semua komponen tersebut secara bertahap untuk membentuk aplikasi *jogging* virtual.

### 3.1 Masalah yang akan Diselesaikan

Ada tiga hal yang harus diperhatikan untuk membangun *virtual jogging app*, yaitu pemandangan, rute perjalanan, dan perubahan pemandangan sesuai dengan rute perjalanan. Pemandangan dapat diciptakan dengan menggunakan Google VR SDK dan *Google StreetView API*, rute perjalanan dari pelari dapat diperoleh menggunakan *Google Directions API*, dan perubahan dari pemandangan diatur dengan sensor gerak.

### 3.2 Analisis Google VR SDK

Dari aplikasi HelloVR yang disediakan di Google VR SDK, ada beberapa kebutuhan yang dapat dipelajari, terutama dari *folder assets*. *Folder assets* berisi bangun ruang tiga dimensi dari ruangan dan objek-objek beserta tekstur masing-masing ruangan dan objek [1]. *Folder assets* berisi *file-file* OBJ seperti CubeRoom.obj, Icosahedron.obj, QuadSphere.obj, dan TriSphere.obj. CubeRoom.obj berfungsi sebagai ruang dalam dunia VR itu, sementara *file-file* OBJ lain adalah objek-objek yang ada di dalam dunia aplikasi HelloVR. Masing-masing *file-file* OBJ dipasangkan dengan *file-file* PNG yang adalah tekstur-teksitur masing-masing *file-file* OBJ. Dari analisis *folder assets* aplikasi HelloVR, dibutuhkan dunia VR yang terdefinisi dengan tampilan seperti di dunia nyata. Dunia VR ini dapat dibentuk lewat dua komponen utama: bangun ruang tiga dimensi dan gambar dari *Google StreetView API*. Bangun ruang tiga dimensi berfungsi sebagai batasan dunia VR tersebut, sementara gambar *Google StreetView API* memberi pemandangan dari tempat dunia nyata. Dengan menciptakan bangun ruang tiga dimensi yang ditambahkan tekstur gambar *Google StreetView API*, pemandangan seperti di dunia nyata dapat direalisasikan pada aplikasi VR.

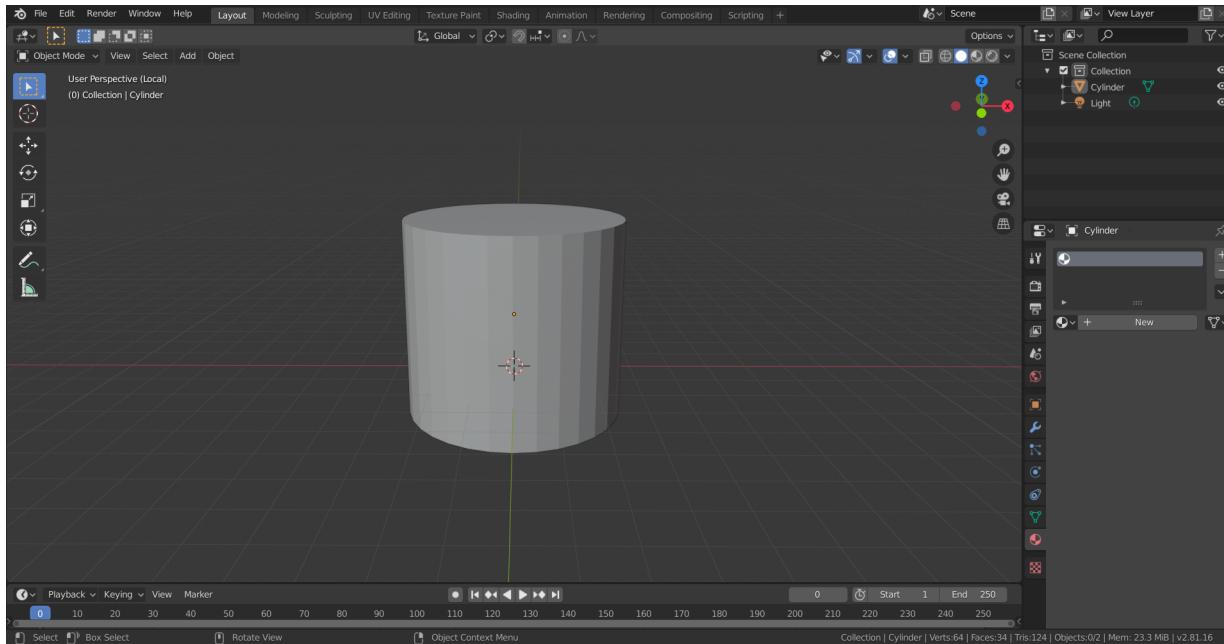
Agar dapat membuat pemandangan VR seperti dunia nyata yang bergerak, tekstur dari bangun ruang tiga dimensi yang sudah diciptakan harus berubah-ubah. Perubahan dari tekstur yang diperoleh *Google StreetView API* haruslah sesuai dengan rute perjalanan. *Google Directions API*lah yang dapat digunakan untuk menentukan rute perjalanan dari asal dan tujuan yang dapat digunakan untuk perubahan gambar tekstur untuk bangun ruang tiga dimensi.

Komponen terakhir dari aplikasi *jogging* virtual yang harus dimanfaatkan adalah sensor *step detector*. Sensor *step detector* ini digunakan untuk menentukan ritme perubahan dari tekstur bangun ruang tiga dimensi. Saat langkah kaki dari pengguna terdeteksi, barulah gambar tekstur dari bangun ruang tiga dimensi diubah seturut rute perjalanan dari *Google Directions API*.

### 3.3 Analisis Pembuatan Bangun Ruang Tiga Dimensi

Bangun ruang tiga dimensi yang tepat untuk dibuat adalah silinder atau tabung. Alasan dipilihnya bentuk silinder ini adalah bentuk silinder cukup merepresentasikan pandangan lateral atau samping dari manusia di dunia nyata. Permukaan samping inilah yang nantinya akan diisi tekstur dari gambar *Google StreetView API*.

Untuk membuat dunia berbentuk silinder, kertas yang digunakan adalah *Blender* versi 2.81. Gambar menunjukkan tampilan *UI Blender* versi 2.81. Setelah bangun ruang silinder dibentuk dari kertas *Blender*, bangun ruang itu disimpan dalam sebuah file OBJ.



Gambar 3.1: Tampilan *UI Blender* Blender versi 2.81

### 3.4 Menampilkan Gambar *StreetView API* pada Bangun Ruang Silinder

Untuk membuat tampilan dunia VR yang sempurna, bagian kedua yang harus dipenuhi adalah pemandangan yang dihasilkan lewat gambar *Google StreetView API* di permukaan samping silinder. *Google StreetView API* menghasilkan gambar pemandangan dari satu arah pandang, maka haruslah dikumpulkan gambar-gambar dari arah-arah yang lain untuk membentuk pemandangan sekeliling yang sempurna dan terlihat seperti dunia nyata [2]. Sebagai contoh, gambar dari empat arah (utara, selatan, timur, dan barat) harus digabungkan untuk menghasilkan pemandangan seperti dunia nyata. Hal yang harus dilakukan untuk mencapai hal tersebut adalah menggabungkan gambar-gambar dari atribut *heading* dari empat arah, dengan satu arah yang berlawanan dengan arah yang lain, lalu dua arah lain yang tegak lurus dengan arah pertama. Dengan kata lain, selisih setiap dua nilai *heading* yang berurutan bernilai 90 (misalnya *heading* dengan nilai 0, 90, 180 dan 270). Gambar 3.2 memperlihatkan empat gambar berukuran  $600 \times 300$  *StreetView* dengan berbagai macam *StreetView* dengan syarat tersebut.

Setelah mendapatkan gambar *StreetView* dari semua arah, gambar-gambar tersebut digabungkan sehingga membentuk pemandangan seperti ada di lokasi tersebut. Gambar 3.3 menunjukkan hasil penggabungan empat gambar dengan deskripsi di atas.

Setelah mendapatkan gambar *StreetView* yang sudah digabungkan tersebut, gambar tersebut dapat dimanfaatkan sebagai tekstur untuk ruang pada file OBJ yang berbentuk silinder sehingga



Gambar 3.2: Gambar dari *StreetView API* berukuran  $600 \times 300$  dengan parameter *heading* yang berbeda-beda

pemandangan *StreetView* dapat ditampilkan pada dunia VR. Gambar tersebut dapat di-*bind* dengan bangun ruang silinder yang akan dibuat.

## 3.5 Analisis *Google Directions API*

Untuk mendapatkan rute perjalanan, *Directions API* dapat dimanfaatkan [3]. File yang diperoleh lewat *Directions API* adalah file JSON yang memiliki *key* dan *value*. Ada beberapa atribut (*key*) dengan nilai (*value*) yang ada pada file JSON dari hasil pemanggilan *Directions API* yang dapat dimanfaatkan.

### 3.5.1 Menentukan Atribut yang akan Digunakan dari JSON *Directions*

File JSON yang dihasilkan memiliki beberapa tingkat *key* dan *value*. Pada tingkat pertama, ada tiga *key*: *geocoded\_waypoints*, *status* dan *routes*. *Key* yang dapat digunakan adalah *routes* yang menunjukkan jalan yang akan ditempuh. Pada tingkat berikutnya, *key routes* memiliki *value* seperti *bounds*, *copyrights*, dan *legs*. Jika melihat bagian *legs* yang menampung atribut-atribut jalan yang ditempuh, ada *distance*, *duration*, *end\_location*, *html\_instructions*, *maneuver*, *polyline*, *start\_location*, dan *travel\_mode*. Dari beberapa atribut dari *legs*, yang dapat digunakan adalah *end\_location* dan *start\_location*, yang menunjuk kepada posisi garis lintang dan garis bujur titik ujung dari jalan yang sedang ditempuh.

Listing 3.1: Atribut *legs* dari *Directions API*

```
"legs" : [
  {
    "distance" : {
      "text" : "3.0 km",
      "value" : 2980
    },
    "duration" : {
```



Gambar 3.3: Contoh hasil penggabungan empat gambar *Street View*

```

    "text" : "35\u00a9mins",
    "value" : 2116
},
"end_address" : "Jl.\u2022Ir.\u2022H.\u2022Juanda\u2022No.100,\u2022Lebakgede . . .",
"end_location" : {
    "lat" : -6.893148,
    "lng" : 107.6131791
},
"start_address" : "Jl.\u2022Ciumbuleuit\u2022No.94,\u2022Hegarmanah . . .",
"start_location" : {
    "lat" : -6.8746719,
    "lng" : 107.6046127
},
"steps" : [
    {
        "distance" : {
            "text" : "1.0\u00a9km",
            "value" : 1008
        },
        "duration" : {
            "text" : "11\u00a9mins",
            "value" : 667
        },
        "end_location" : {
            "lat" : -6.8833328,
            "lng" : 107.6049108
        },
        "html_instructions" : "Head\u2022\u0003cb\u0003esouth . . .",
        "polyline" : {
            "points" : "tu}h@yowoSdAP|AL'Cb@dAHHBvC . . ."
        },
        "start_location" : {

```

```
        "lat" : -6.8746719,
        "lng" : 107.6046127
    },
    "travel_mode" : "WALKING"
},
{
    "distance" : {
        "text" : "1.1\u00a5km",
        "value" : 1078
    },
    "duration" : {
        "text" : "14\u00a5mins",
        "value" : 834
    },
    "end_location" : {
        "lat" : -6.885218399999999,
        "lng" : 107.6137497
    },
    "html_instructions" : "Turn\u2022\u003cb\u003eleft\u003c/b\u2022...",
    "maneuver" : "turn-left",
    "polyline" : {
        "points" : "xk_i@uqwoSCEAECKAM?K?E?..."
    },
    "start_location" : {
        "lat" : -6.8833328,
        "lng" : 107.6049108
    },
    "travel_mode" : "WALKING"
},
{
    "distance" : {
        "text" : "0.9\u00a5km",
        "value" : 884
    },
    "duration" : {
        "text" : "10\u00a5mins",
        "value" : 603
    },
    "end_location" : {
        "lat" : -6.89314079999999,
        "lng" : 107.6130843
    },
    "html_instructions" : "Turn\u2022\u003cb\u003eright...",
    "maneuver" : "turn-right",
    "polyline" : {
        "points" : "rw_i@}hyoSzCLlAHz@Bd@@fB@tBDfABR@L... "
    },
    "start_location" : {
        "lat" : -6.885218399999999,
        "lng" : 107.6137497
    },
}
```

```

        "travel_mode" : "WALKING"
    } ,
    {
        "distance" : {
            "text" : "10\u00a0m",
            "value" : 10
        },
        "duration" : {
            "text" : "1\u00a0min",
            "value" : 12
        },
        "end_location" : {
            "lat" : -6.893148,
            "lng" : 107.6131791
        },
        "html_instructions" : "Turn\u2022\u003cb\u003eleft\u003c...",
        "maneuver" : "turn-left",
        "polyline" : {
            "points" : "biai@wdyoS@S"
        },
        "start_location" : {
            "lat" : -6.893140799999999,
            "lng" : 107.6130843
        },
        "travel_mode" : "WALKING"
    }
],
"traffic_speed_entry" : [] ,
"via_waypoint" : []
}
],
"overview_polyline" : {
    "points" : "tu}h@yowoSdAP|AL'Cb@dAH'Dd@~Bd@hG..."
},
"summary" : "Jl.\u2022Ciumbuleuit,\u2022Jl.\u2022Siliwangi...",
"warnings" : [
    "Walking\u2022directions\u2022are\u2022in\u2022beta.\u2022Use\u2022caution..."
],
"waypoint_order" : []
}
],
"status" : "OK"
}

```

Listing 3.1 menunjukkan atribut *legs* dari hasil JSON *Directions*.

### 3.5.2 Cara Memanfaatkan Atribut

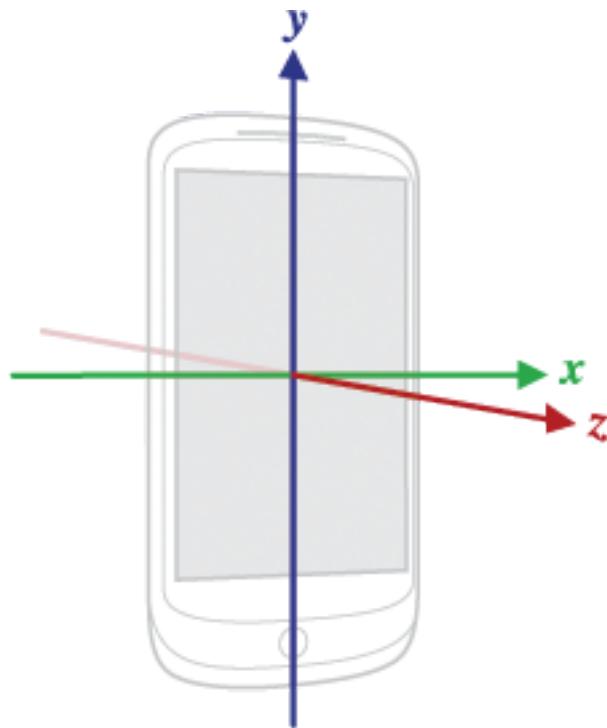
Setelah memperoleh nilai *end\_location* dan *start\_location*, jalur tempuh pada jalan yang sedang ditempuh dapat ditentukan lewat posisi garis lintang dan garis bujur kedua titik ujung jalan. Cara untuk membentuk jalan secara matematis adalah menggunakan selisih antara garis lintang dan garis bujur satu titik ujung jalan ke titik ujung jalan lain.

## 3.6 Analisis Step Detector Sensor

Subbab ini akan menunjukkan hasil eksperimen pengujian *step detector sensor* dan menjelaskan cara memanfaatkannya pada aplikasi.

### 3.6.1 Eksperimen Pengujian Step Detector Sensor

*Step detector sensor* adalah sensor yang mendeteksi langkah kaki, lalu merangsang suatu aksi di dalam aplikasi. Oleh karena itu, eksperimen untuk melihat gerakan-gerakan yang membuat perangkat bergerak mendeteksi langkah kaki dilakukan. Metode melakukan eksperimen ini adalah dengan memilih beberapa jenis gerakan pada sumbu-sumbu sensor perangkat bergerak, lalu melihat gerakan mana saja yang mendapat rangsang untuk memicu aksi pada aplikasi. Gambar 3.4 menunjukkan sumbu-sumbu yang dibaca sensor pada perangkat bergerak [4].



Gambar 3.4: Sumbu Sensor Perangkat Bergerak

Setelah mengetahui sumbu-sumbu sensor perangkat bergerak, langkah selanjutnya adalah menentukan gerakan-gerakan yang dilakukan pada eksperimen ini. Tabel 3.1 menunjukkan gerakan-gerakan yang dilakukan pada perangkat bergerak serta hasil deteksi *step detector sensor*.

Gambar 3.5 menunjukkan *log* dari aplikasi perangkat bergerak pengujian *step detector sensor* ketika *step detector sensor* mendeteksi rangsang.

Dari eksperimen, dapat disimpulkan bahwa gerakan perangkat ke sana dan ke mari (gerakan vibrasi) adalah gerakan yang memberikan rangsang pada *step detector sensor* sehingga memicu aksi pada aplikasi perangkat bergerak.

### 3.6.2 Cara Memanfaatkan Step Detector Sensor

Untuk membuat animasi atau perubahan pemandangan sesuai rute tempuh yang sudah diperoleh, harus ada perubahan gambar sesuai langkah kaki yang diambil pengguna. Jadi, setiap kali sensor mendapat rangsang, *event* yang dipicu agar terjadi adalah gambar berubah seperti yang dijelaskan pada Subbab 3.5.2, tetapi perubahan gambar *StreetView* yang sesuai *Directions API* harus berubah dengan tahap yang benar agar animasi pemandangan terlihat baik sesuai dengan rute.

Tabel 3.1: Hasil Eksperimen Pengujian *Step Detector Sensor*

No	Gerakan	Hasil (terdeteksi atau tidak)
1	Satu kali bergerak pada sumbu <i>x</i>	Tidak terdeteksi
2	Satu kali bergerak sepanjang sumbu <i>y</i>	Tidak terdeteksi
3	Satu kali bergerak pada sumbu <i>z</i>	Tidak terdeteksi
4	Gerakan sana ke mari pada sumbu <i>x</i>	Terdeteksi
5	Gerakan sana ke mari pada sumbu <i>y</i>	Terdeteksi
6	Gerakan sana ke mari pada sumbu <i>z</i>	Terdeteksi

```
2021-02-09 23:54:47.292 3985-3985/com.example.stepdetectorsensor D/Step: Step Taken
2021-02-09 23:54:48.012 3985-3985/com.example.stepdetectorsensor D/Step: Step Taken
2021-02-09 23:54:48.569 3985-3985/com.example.stepdetectorsensor D/Step: Step Taken
2021-02-09 23:54:49.253 3985-3985/com.example.stepdetectorsensor D/Step: Step Taken
2021-02-09 23:54:49.891 3985-3985/com.example.stepdetectorsensor D/Step: Step Taken|
2021-02-09 23:54:50.531 3985-3985/com.example.stepdetectorsensor D/Step: Step Taken
```

Gambar 3.5: *Log console* dari aplikasi perangkat bergerak pengujian *step detector sensor* ketika mendeteksi rangsang

## BAB 4

# RANCANGAN

Bab ini menjelaskan perancangan aplikasi, termasuk algoritma-algoritma untuk mengolah *Google StreetView API*, *Google Directions API*, serta modifikasi yang dilakukan pada aplikasi HelloVR untuk membangun aplikasi *jogging* virtual.

### 4.1 Rancangan Antarmuka

Aplikasi yang akan dibangun terdiri atas dua *activity*, yaitu *activity* utama dan *activity* VR.

#### 4.1.1 *Activity* Utama

*Activity* utama adalah *activity* yang ditampilkan pertama kali saat pengguna membuka aplikasi dengan *portrait layout*. Fungsi *activity* ini adalah menerima masukan pengguna, yaitu lokasi asal dan lokasi tujuan saat berlari, serta memicu *activity* kedua, yaitu *activity* VR. Ada tiga komponen utama dari *activity* ini, yaitu dua buah *textbox* dan sebuah tombol. Satu *textbox* adalah *textbox* "origin", dan *textbox* yang lain adalah *textbox* "destination". Gambar 4.1 menggambarkan tampilan *activity* utama.

##### *Textbox* Origin

*Textbox* adalah komponen pada antarmuka yang menerima masukan pengguna yang berupa tulisan atau *string*. *Textbox* "origin" akan menerima masukan pengguna yang merupakan lokasi asal dari rute lari.

##### *Textbox* Destination

*Textbox* Destination adalah *textbox* kedua dan berada di bawah *textbox* origin akan menerima masukan berupa lokasi tujuan dari rute lari yang dimasukkan pengguna.

##### Tombol "Start Running"

Tombol "Start Running" adalah tombol yang akan memicu *activity* VR yang sesuai dengan informasi dari dua *textbox* di atas ketika ditekan. Tombol ini berada di bawah *destination* *textbox*.

#### 4.1.2 *Activity* VR

*Activity* VR adalah *activity* yang menampilkan pemandangan VR secara VR ketika berlari. Untuk memunculkan *activity* ini, pengguna harus menekan tombol dari *activity* utama (dijelaskan pada Subbab 4.1.1) terlebih dahulu.

*Activity* ini dimunculkan ketika mengakses *Google Cardboard*. Untuk menuju ke *activity* VR, pengguna harus memutar perangkat sehingga perangkat berada dalam posisi *landscape*. Setelah perangkat berada dalam posisi *landscape*, *activity* VR akan dimunculkan seperti yang ditunjukkan pada Gambar 4.2.

Pada tampilan VR tersebut, gambar akan berubah-ubah sesuai dengan langkah kaki pengguna. Perubahan tersebut akan terjadi ketika mencapai pengguna mencapai jarak 100 meter. Jika pengguna sudah mencapai tujuan sesuai dengan jarak yang ditempuh, *activity* VR akan berhenti mengubah gambar.

## 4.2 Rancangan Program

Subbab ini akan menjelaskan rancangan program, mulai dari rancangan kelas dan algoritma-algoritma yang digunakan pada *method-method* yang penting.

### 4.2.1 Rancangan Kelas

Rancangan kelas dari aplikasi akan menggunakan seluruh bagian pada aplikasi HelloVR dan beberapa tambahan kelas. Gambar 4.3 menunjukkan atribut-atribut dan *method-method* dari masing-masing kelas, serta hubungan antara satu kelas dan kelas-kelas lain.

Beberapa kelas di Gambar 4.3 tidak memiliki deskripsi lebih karena berasal dari *library* Java ataupun *Google VR SDK*. Kelas-kelas yang ditambahkan adalah:

#### 1. MainActivity

Kelas yang mengelola *activity* utama. Atribut yang dimiliki kelas ini:

- **protected EditText originET**  
Bagian dari antarmuka yang dapat menerima masukan lokasi asal dari rute lari.
- **protected EditText destET**  
Bagian antarmuka yang dapat menerima masukan lokasi tujuan dari rute lari.
- **protected Button startButton**  
Tombol yang digunakan untuk memuat rute perjalanan dan gambar pemandangan, serta memicu *activity* VR.

*Method-method* yang dimiliki kelas ini adalah:

- **protected void onClick(View view)**  
*Method* yang dijalankan ketika komponen yang sudah dipasangkan *onClickListener* ditekan.

#### 2. VRActivity

*Class* yang mengelola *activity* VR. Atribut-atribut yang dimiliki *class* ini adalah:

- **private SensorManager sensorManager**  
Atribut yang mengatur sensor dari perangkat Android.
- **private Sensor stepDetector**  
Sensor pendekripsi langkah kaki.
- **protected TexturedMesh room**  
Bangun ruang VR yang sudah dipasangkan dengan objek *Texture*.
- **protected ArrayList<Texture> roomTex**  
*ArrayList* dari gambar-gambar *StreetView* yang merupakan tekstur dari bangun ruang VR.
- **protected Texture finishedTexture**  
Gambar dari tekstur bangun ruang VR yang menandakan perjalanan pengguna sudah selesai.

- **private int[] stepDistances**  
Array dari jarak setiap *steps* dari rute lari.
- **private int distanceElapsed**  
Atribut ini menyimpan jarak yang sudah ditempuh pengguna.
- **private int curStepIndex**  
Atribut yang menyimpan *index* dari *step* yang sedang ditempuh saat ini.
- **private int curStepDistance**  
Atribut yang menyimpan penjumlahan jarak dari *step* saat ini dan jarak dari setiap *steps* yang sudah ditempuh.

*Method-method* yang dimiliki *class* ini adalah:

- **public void drawRoom()**  
*Method* untuk menggambar ruangan tiga dimensi.
- **public void onSurfaceCreated(EGLConfig eglConfig)**  
*Method* yang dipanggil ketika permukaan VR diciptakan. Parameter yang dimiliki *method* ini adalah:
  - **EGLConfig eglConfig**  
Konfigurasi dari *OpenGL renderer*.*Method* ini tidak memiliki nilai kembali.
- **public void onSensorChanged(SensorEvent sensorEvent)**  
*Method* yang dipanggil ketika sensor *step detector* mendeteksi *event*.

### 3. StreetViewLoader

*Class* yang berfungsi untuk memuat gambar *StreetView* dan menyatukan semua gambar itu, membentuk gambar pemandangan yang utuh. Atribut-atribut yang dimiliki *class* ini adalah:

- **protected Bitmap streetViewBitmap**  
Atribut yang menampung *Bitmap* dari *StreetView* yang telah diunduh dari *StreetView API*.
- **protected Activity activity**  
*Activity* yang dihubungkan dengan *class* ini sehingga komunikasi antara objek dua kelas ini dapat terjadi.
- **protected Bitmap streetViewBitmap**  
Atribut untuk menyimpan *bitmap* dari gambar *StreetView*.
- **protected Activity activity**  
Atribut dari *activity* yang memanggil *method* dari *class* ini sehingga data dan nilai yang sudah diperoleh dari *class* ini dapat diserahkan pada *activity* ini. *Method* ini tidak memiliki nilai kembali.
- **protected Intent intent**  
*Intent* dari *activity* yang memicu *activity* untuk dijalankan.
- **protected int imgCount**  
Atribut yang menyimpan angka perhitungan setiap kali gambar dimuat dari *StreetView API*.
- **protected ArrayList<JSONObject> arrSteps**  
*ArrayList* dari *JSON Object* dengan *key "steps"* dari *JSON* yang diperoleh dari *Directions API*.

*Method-method* yang dimiliki *class* ini adalah:

- **protected doInBackground(Void... aVoid)**  
*Method* yang memuat dan menyatukan gambar-gambar *StreetView API*, yang dijalankan pada *AsyncTask* berbeda.
- **protected Void onPostExecute(Void aVoid)**  
*Method* yang dipanggil setelah *doInBackground()* selesai dijalankan, untuk memicu *VrActivity*.
- **public String[] generateStreetViewURL(int svUrlLength, boolean isStart)**  
*Method* ini akan men-generate URL untuk mengakses *StreetView API*.

Parameter:

- **int svUrlLength**  
 Parameter ini menyatakan berapa banyak *string* URL untuk mendapatkan gambar-gambar *StreetView*.
- **boolean isStart**  
 Nilai *boolean* yang menandakan apakah *key* dari *JSONObject start\_location* atau *end\_location*.

Nilai Kembalian:

- **String[] urlArr**  
*Array* dari URL *StreetView API* yang sudah di-generate.

#### 4. DirectionsLoader

*Class* yang memuat JSON dari rute lari. Atribut-atribut yang dimiliki kelas ini adalah:

- **protected Activity activity**  
*Activity* yang diacu agar dapat berhubungan dengan objek dari *class* ini.
- **protected String jsonText**  
 Atribut yang menyimpan *text* yang diperoleh dari *Directions API*.

*Method-method* yang dimiliki *class* ini adalah:

- **protected Void doInBackground(Strings... strings)**  
*Method* yang dijalankan di *AsyncTask* berbeda, yaitu untuk memuat file JSON *Directions API*.
- **protected Void onPostExecute(Void aVoid)**  
*Method* yang dijalankan setelah *method doInBackground()* selesai dijalankan, untuk mem-parse JSON lewat objek dari *class DirectionsExtractor Directions API*, lalu memanggil menginstansiasi objek dari *class StreetViewLoader*.

#### 5. DirectionsExtractor

*Class* yang berfungsi sebagai *parser* JSON yang diperoleh dari *Directions API*. Atribut yang ada pada *class* ini adalah:

- **protected ArrayList<JSONObject> arrSteps**  
*ArrayList* dari objek-objek *steps* yang diperoleh dari *Directions API*.
- **protected String jsonText**  
*Text* JSON yang diperoleh dari *Directions API*.

*Method* yang ada pada *class* ini adalah:

- **protected void extractJSONDir()**  
*Method* yang akan mengambil bagian *steps* dari JSON *Directions API*.

### 4.2.2 Algoritma-Algoritma yang Digunakan

Ada beberapa algoritma yang digunakan untuk melakukan beberapa proses seperti mengolah *Google StreetView API*, *Google Directions API*, dan *step detector*.

#### Pemanfaatan *Google Directions API*

*Directions API* adalah *API* yang menggunakan protokol HTTPS dan menghasilkan keluaran dalam bentuk JSON. Untuk memuat JSON berisi rute perjalanan dari asal sampai tujuan, pemanggilan *Directions API* harus dilakukan melalui HTTP/HTTPS.

---

#### Algorithm 1 Algoritma Mengunduh JSON dari *Directions API* dan *Parsing*

---

```

1: function PROCESSJSONDIRECTIONS(originLoc,destLoc)
2:   dirText  $\leftarrow$  getJSONFromDirAPI(originLoc,destLoc)
3:   dirJSON  $\leftarrow$  toJSON(dirText)
4:   jsonRoute  $\leftarrow$  dirJSON.getJSONObject("routes")
5:   jsonArrLegs  $\leftarrow$  jsonRoute.getJSONArray("legs")
6:   Declare Array processedJSONObj
7:   for elements in jsonArrLegs do
8:     jsonLegs  $\leftarrow$  element.getJSONObject("legs")
9:     jsonArrSteps  $\leftarrow$  jsonLegs.getJSONArray("steps")
10:    Insert all elements from jsonArrSteps to processedJSONObj

```

---

Setelah JSON rute diperoleh, JSON rute dapat digunakan dengan membaca atribut dan nilai-nilainya sesuai kebutuhan. Beberapa nilai atribut dalam JSON ini berupa JSON *object*, ada yang berupa JSON *array*. Karena itu, *JSON parsing* harus dilakukan sesuai dengan ketentuan tersebut. Algoritma *parsing* dari JSON *Directions* dipaparkan pada Algoritma 1.

Setelah JSON dari *Directions API* diunduh, *parsing* adalah langkah selanjutnya untuk mengambil informasi yang dibutuhkan. Karena bagian *steps* adalah bagian yang harus diperoleh, haruslah *value* dari *key* *legs* harus diperoleh terlebih dahulu. Selanjutnya, *JSON Array* yang merupakan *value* dari *key* "legs", dan objek-objek dari *JSON Array* *steps* itu dapat dimasukkan ke dalam sebuah struktur data seperti *array* agar dapat digunakan.

#### Algoritma Memperoleh dan Mengolah *Google StreetView API*

Pemandangan yang harus dibentuk adalah pemandangan dari lokasi asal dan tujuan dari pengguna untuk membentuk perjalanan dari pengguna, gambar-gambar dari lokasi asal dan setiap *steps* sampai lokasi tujuan harus diperoleh. Artinya, semua gambar dari setiap *steps* dari rute *Directions API* haruslah dimuat terlebih dahulu seluruhnya. Jika sudah dimuat semuanya, gambar-gambar tersebut akan dapat dimanfaatkan untuk membentuk lingkungan lari virtual.

*StreetView API* berfungsi untuk memuat gambar pemandangan dan menggunakan menggunakan menggunakan HTTPS. Karena satu kali pemanggilan hanya menghasilkan gambar dari satu *heading* atau arah, beberapa gambar dari beberapa arah pandang (*heading*) haruslah diunduh terlebih dahulu. Setelah semua gambar itu terunduh, barulah semua gambar itu digabungkan menjadi satu gambar untuk menjadi tekstur bangun ruang silinder, lalu dapat disimpan dalam memori perangkat.

Gambar 4.4 menunjukkan langkah-langkah yang dilakukan untuk memuat semua gambar lokasi dari setiap *steps* rute lari pengguna.

Peubah *i* pada *flowchart* digunakan untuk mengiterasi semua elemen *array arrSteps*. Untuk setiap iterasi, URL untuk mengakses *StreetView API* dibuat sebanyak jumlah gambar yang ingin dihasilkan. Arah atau *heading* dari gambar-gambar yang dihasilkan ditentukan dengan peubah *x* pada *flowchart*. Peubah *y* merupakan selisih *heading* dari satu iterasi ke iterasi yang berikutnya. Nilai 360 menunjuk pada 360 derajat dalam satu putaran. Untuk mendapatkan pemandangan

yang sekeliling yang penuh, gambar dari setiap derajat ke- $x$  selama nilai  $x$  masih bernilai lebih kecil dari 360. Setelah gambar-gambar itu sudah diperoleh dari *StreetView API*, gambar-gambar tersebut harus disatukan, lalu gambar itu dapat disimpan dalam *file*, lalu gambar dalam *file* itu dapat digunakan.

### Pemanfaatan Sensor *Step Detector*

Sensor *step detector* digunakan untuk mendeteksi langkah kaki pengguna. Untuk membentuk skenario yang tepat untuk memungkinkan perubahan pemandangan yang sudah ditangani oleh *StreetView* dan *Directions API*, waktu perubahan dari pemandangan itulah yang harus ditangani. Sensor *step detector*lah yang akan menangani bagian waktu perubahan pemandangan.

Dari *JSON Object* dengan key "*steps*", dapat diperoleh *JSON Object* dengan key "*distance*" yang memiliki nilai "*value*". "*Value*" dari "*distance*" menunjuk pada jarak dari "*step*" yang diacu. Untuk berlari sesuai dengan jarak itu, harus ada sebuah peubah bertipe numerik (*integer*) yang menandakan *progress* perjalanan pelari, lalu harus ada sebuah konstan yang menyatakan jarak dari satu langkah. Peubah yang mencatat *progress* akan bertambah setiap kali sensor mendeteksi langkah kaki. Jika *progress* sudah mencapai nilai jarak dari *step* saat ini, pemandangan VR berulah dapat diubah dengan gambar pemandangan dari *step* berikutnya. Algoritma 2 menunjukkan hal-hal yang terjadi saat sensor *step detector* menerima rangsang.

---

#### Algorithm 2 Algoritma Saat *Step Detector* Menerima Rangsang

---

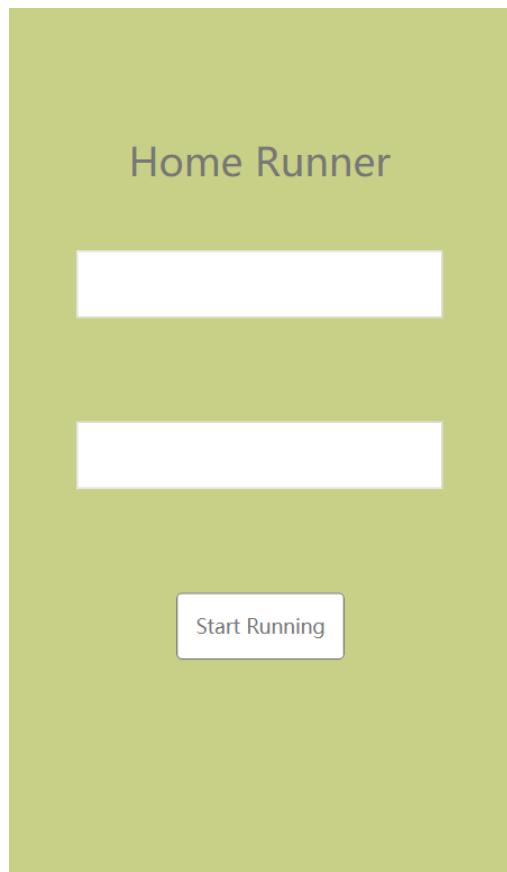
```

1: function ONSTEPDETECTORCHANGED(stepsDist)
2:   distanceElapsed  $\leftarrow$  0
3:   currentStepDist  $\leftarrow$  first element of stepsDist.value
4:   curBenchMark  $\leftarrow$  currentStepDist.value
5:   if sensor detects footstep then
6:     distanceElapsed  $\leftarrow$  distanceElapsed + DISTANCE_PER_STEP
7:     if currentStepDist.NextISNOTNULL then
8:       if distanceElapsed  $\geq$  curBenchMark then
9:         currentStepDist  $\leftarrow$  currentStepDist.next
10:        curBenchMark  $\leftarrow$  curBenchMark + currentStepDist.value

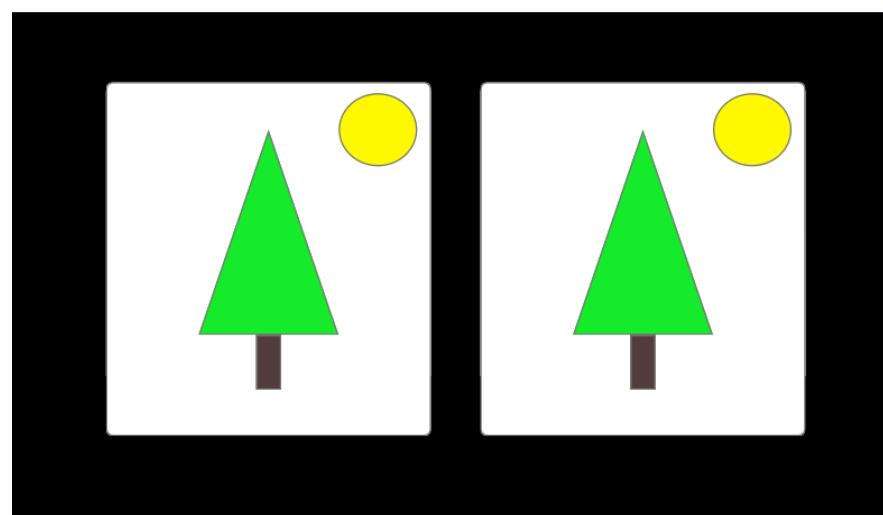
```

---

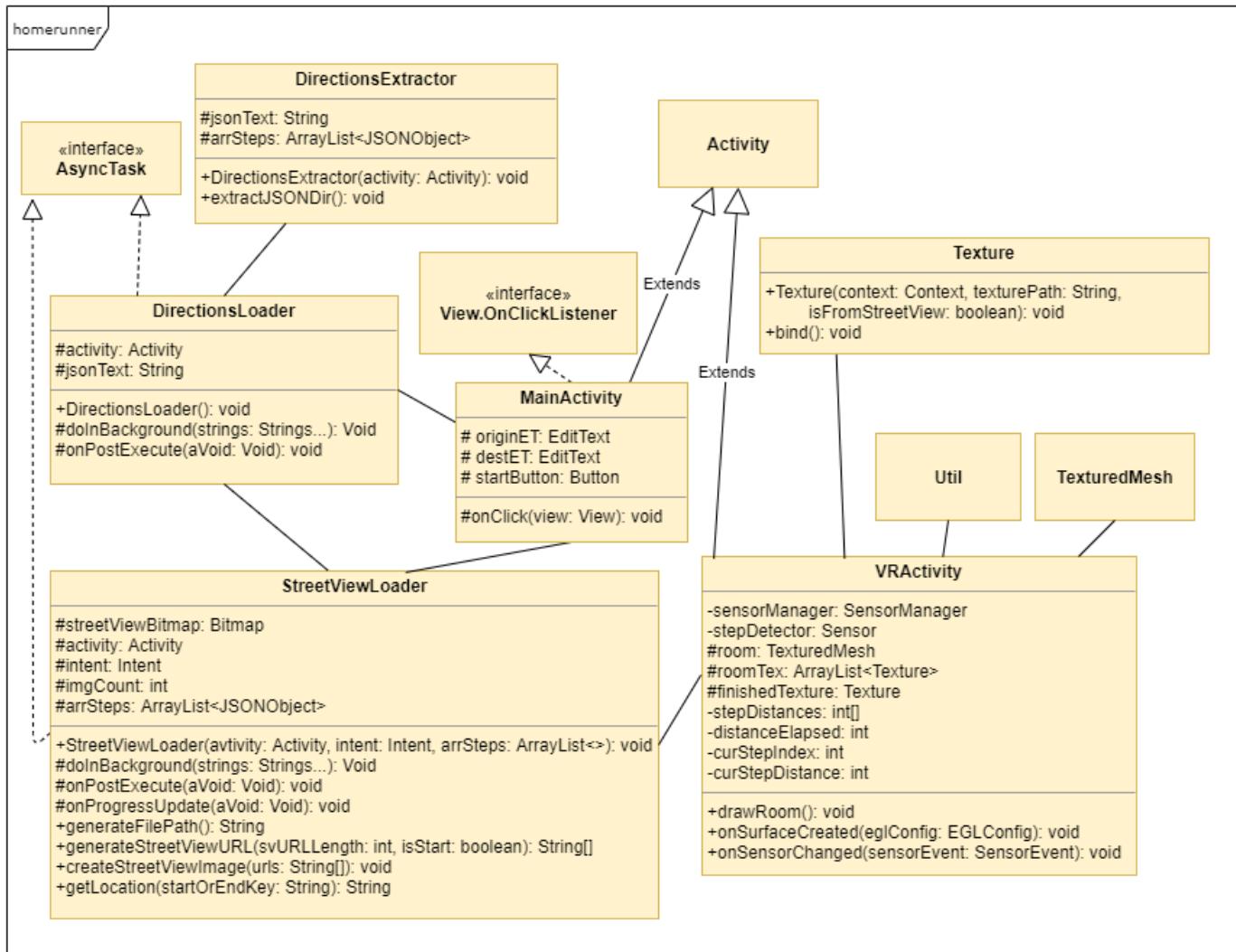
*StepsDist* merupakan *array* dari *steps distance*. *DistanceElapsed* merupakan peubah yang mencatat *progress* langkah dari pengguna, sementara *currentStepDist* merupakan peubah yang mencatat jarak yang harus ditempuh untuk mencapai *steps distance* berikutnya. Peubah *distanceElapsed* akan ditambah dengan konstanta DISTANCE\_PER\_STEP setiap kali ada langkah kaki yang terdeteksi. Jika *distanceElapsed* sudah mencapai *curStepDistance*, *curStepDistances* akan ditambah dengan nilai *step distance* selanjutnya.

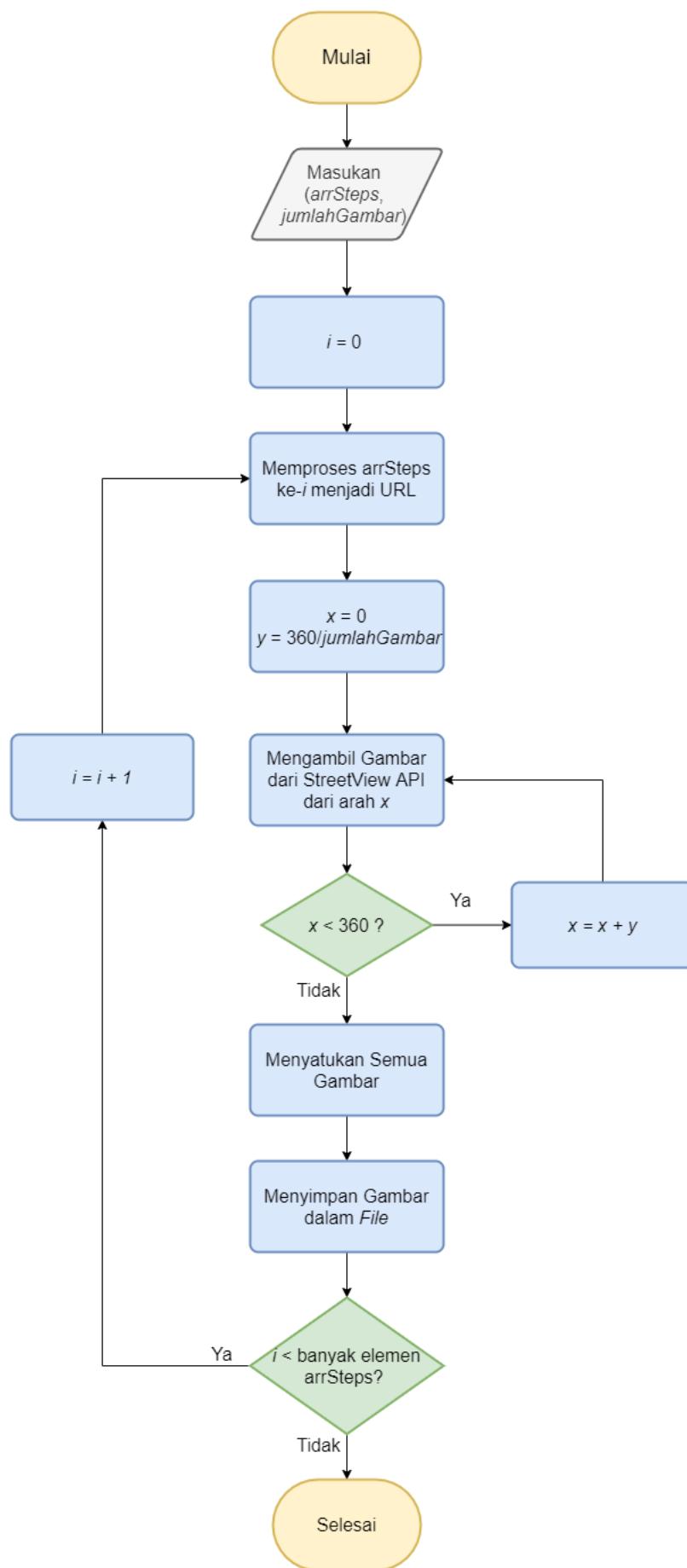


Gambar 4.1: Rancangan *Activity* utama aplikasi



Gambar 4.2: Rancangan *Activity* VR

Gambar 4.3: Diagram *class* dari Aplikasi



Gambar 4.4: Flowchart dari Proses Memperoleh dan Menyatukan Gambar dari *StreetView API*



## BAB 5

# IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dijelaskan mengenai implementasi, pengujian, dan masalah yang dihadapi saat implementasi.

### 5.1 Implementasi

Aplikasi dikembangkan dengan Android Studio, dengan sebuah *laptop*. Aplikasi dijalankan pada tiga buah ponsel Android.

#### 5.1.1 Lingkungan Implementasi

Berikut adalah spesifikasi *laptop* dan kakas yang digunakan untuk implementasi:

1. Processor : Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.
2. RAM : 16.0 GB (15.9 GB usable).
3. Versi Android Studio : 4.1.1
4. Google VR for Android : 1.190.0

Berikut adalah spesifikasi ponsel Android yang digunakan untuk implementasi:

1. Processor : Qualcomm Snapdragon 835
2. Sistem Operasi : Android 9.0 (Pie)
3. Sensor yang dimiliki : Accelerometer, gyroscope, magnetometer, step detector.

#### 5.1.2 Hasil Implementasi

Yang dihasilkan dari implementasi ini adalah aplikasi VR untuk berlari. Aplikasi dapat diperoleh di Google Play Store dan dapat di-*install* pada gawai dengan sistem operasi Android.

#### 5.1.3 Tampilan Ketika Aplikasi Pertama Kali Dibuka

Ketika pengguna membuka aplikasi pertama kali, akan ada judul aplikasi, dua *text box*, dan satu buah tombol bertuliskan "Start Running". Gambar 5.1 menggambarkan tampilan aplikasi dan Gambar 5.2 mengilustrasikan *state* pengguna saat itu.

#### Tampilan ketika Tombol Ditekan dengan *Textbox* dalam Keadaan Kosong

Ketika pengguna menekan tombol "Start Running" dan keadaan salah satu *textbox* kosong, akan ada *Toast* yang memberikan pesan bahwa *textbox* tidak boleh ada dalam keadaan kosong. Gambar 5.3 menunjukkan tampilan *activity* ini.



Gambar 5.1: *Activity* utama aplikasi



Gambar 5.2: Ilustrasi Pengguna saat Memasukkan Masukan Lokasi Asal dan Tujuan



Gambar 5.3: Tampilan Aplikasi ketika tombol "*Start Running*" Ditekan saat *textbox origin* atau *destination* kosong



Gambar 5.5: Ilustrasi Pengguna saat Melihat Tampilan Aplikasi Tersebut

Gambar 5.4: Tampilan *Google Cardboard* sebelum gawai diputar

### Aplikasi ketika Masukan Lokasi yang Sah Dimasukkan, Saat Gawai dalam Posisi *Portrait*

Saat masukan yang sah dimasukkan pengguna, pengguna akan langsung diarahkan ke halaman yang memberitahu untuk mempersiapkan *Cardboard viewer*, memutar gawai agar menjadi *landscape*. Gambar 5.4 menunjukkan tampilan peringatan ini dan Gambar 5.5 mengilustrasikan *state* dari pengguna.

#### Aplikasi saat Pengguna Berlari

Setelah memasukkan masukan lokasi asal dan tujuan yang benar lewat dua *textbox*, siap dengan *Cardboard viewer* dan gawai berada dalam posisi *landscape*, aplikasi akan menampilkan dunia VR dengan pemandangan sesuai dengan lokasi asal, dan pemandangan akan berubah sesuai jarak yang pengguna telah tempuh. *Activity VR* saat pengguna berlari digambarkan pada Gambar 5.6 dan Gambar 5.7 menunjukkan *state* pengguna.

#### 5.1.4 Aplikasi Setelah Pengguna Menyelesaikan Perjalannya

Ketika pengguna sudah berlari sampai tujuan, tampilan dengan tulisan "*Congratulations! You finished your run*". Gambar 5.8 menunjukkan halaman ini.

## 5.2 Pengujian

Pengujian dilakukan dalam dua metode: fungsional dan eksperimental. Pengujian fungsional dilakukan untuk menguji apakah aplikasi sudah berfungsi dengan seharusnya, sedangkan pengujian eksperimental ditujukan untuk mengetahui fungsi sensor *step detector* dalam aplikasi pada beberapa perangkat yang digunakan dalam pengujian fungsional.



Gambar 5.6: Tampilan VR saat Pengguna Berlari



Gambar 5.7: Ilustrasi Pengguna saat Berlari



Gambar 5.8: Tampilan VR saat Pengguna Mencapai Tujuan

Tabel 5.1: Daftar Perangkat yang digunakan dalam Pengujian

No	Processor	Sistem Operasi	RAM
1	Qualcomm Snapdragon 835 2.45 GHz	Android OS v9.0 (Pie)	6GB
2	Quad-Core 2.3 GHz	Android v5.0 (Lollipop)	4GB
3	1.6GHz octa-core	Android OS v8.1 (Oreo)	2GB

### 5.2.1 Pengujian Fungsional

Hal pertama yang dilakukan pada pengujian fungsional ini adalah mencoba meng-*install* aplikasi pada tiga perangkat dengan spesifikasi yang dirincikan pada Tabel 5.1.

Setelah memasang aplikasi pada masing-masing perangkat, pengujian fungsional dapat dilakukan dengan mempersiapkan himpunan masukan dengan keluaran yang diharapkan, lalu memasukkan setiap masukan pada himpunan masukan pada setiap perangkat saat menjalankan aplikasi. Tabel 5.2 memperlihatkan masukan pengujian beserta hasil yang diharapkan.

Masukan 1 sampai 4 ditujukan untuk menguji *string* lokasi yang dieja dengan lengkap dan disingkat pada masing-masing masukan *origin* dan *destination*. Masukan 5 dan 6 ditujukan untuk menguji masukan dari nama-nama tempat yang umum didengar dan dibicarakan orang, namun jarang ditulis orang sehingga penulisan nama sering kali salah (kekurangan satu huruf dalam pengejaannya).

Setelah himpunan masukan untuk pengujian telah ditentukan, pengujian dilakukan pada masing-masing perangkat. Tabel 5.3 memperlihatkan hasil pengujian pada Perangkat 1, Tabel 5.4 memperlihatkan pengujian pada Perangkat 2, Tabel 5.5 memperlihatkan pengujian pada Perangkat 3.

Pada Perangkat 1, aplikasi berjalan dengan sangat lancar, mengeluarkan hasil yang sesuai dengan yang diharapkan pada semua masukan.

Pada Perangkat 2, aplikasi hanya berhasil pada Masukan 1 dan 7, tetapi masukan-masukan yang lain membuat aplikasi terhenti secara tiba-tiba, yaitu masukan yang memiliki karakter spasi (*whitespace*). Hal ini mungkin disebabkan *operating system* (OS) yang kurang tinggi sehingga *character coding* pada *textbox* masukan tidak dapat dilakukan dengan baik sehingga mengolah *string URL* menjadi sesuatu hal yang tidak dapat diproses aplikasi.

Pada Perangkat 3, aplikasi tidak dapat berfungsi dengan baik karena pada tampilan VR, aplikasi tidak dapat merespon terhadap pergerakan yang terjadi pada perangkat. Pemandangan VR langsung terhenti dan tidak bisa berlanjut sampai berlari ke tujuan. Hal ini diduga karena ukuran RAM yang kurang sehingga tidak dapat menjalankan aplikasi dengan seharusnya.

### 5.2.2 Pengujian Eksperimental

Pada pengujian eksperimental, akan dilakukan *logging* untuk melihat jarak yang ditempuh pada *virtual jogging app*. Perangkat yang digunakan untuk pengujian ini adalah Perangkat 1 dan 2 yang dijelaskan pada Subbab 5.2.1. Perangkat 3 tidak digunakan karena aplikasi tidak dapat berjalan dengan seharusnya pada perangkat ini. Masukan untuk aplikasi pada masing-masing aplikasi adalah "unpar" untuk lokasi asal dan "bip,bandung" untuk lokasi tujuannya.

Gambar 5.9 dan Gambar 5.10 menampilkan *logging* dari beberapa langkah terakhir perjalanan.

Dari *log console virtual jogging app*, dapat dilihat bahwa pada masing-masing perangkat, jarak tempuh pengguna terlihat bertambah setiap menerima rangsang yang adalah langkah kaki pengguna. Hasil yang ditampilkan pada masing-masing *log console* serupa.

Tabel 5.2: Himpunan Masukan untuk Pengujian Aplikasi

Nomor Masukan	Masukan <i>origin,destination</i>	Hasil yang Diharapkan
1	"unpar","bip,bandung"	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"
2	"unpar","bandung indah plaza"	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"
3	"universitas katolik parahyangan","bip,bandung"	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"
4	"universitas katolik parahyangan","bandung indah plaza"	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"
5	"rs borromeus","rs immanuel"	Berhasil menampilkan pemandangan di sekitar Rumah Sakit Santo Borromeus, dapat berjalan sampai Rumah Sakit Immanuel, lalu menampilkan tulisan "Congratulations"
6	"rs borromeus","rs immanuel"	Berhasil menampilkan pemandangan di sekitar Rumah Sakit Santo Borromeus, dapat berjalan sampai Rumah Sakit Immanuel, lalu menampilkan tulisan "Congratulations"
7	"ciwalk","boromeus"	Berhasil menampilkan pemandangan di sekitar <i>Cihampelas Walk</i> , dapat berjalan sampai Rumah Sakit Santo Borromeus, lalu menampilkan tulisan "Congratulations"

Tabel 5.3: Hasil Pengujian pada Perangkat 1

Nomor Masukan	Hasil	Status
1	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"	Berhasil
2	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"	Berhasil
3	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, secara virtual dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"	Berhasil
4	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"	Berhasil
5	Berhasil menampilkan pemandangan di sekitar Rumah Sakit Borromeus, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"	Berhasil
6	Berhasil menampilkan pemandangan di sekitar Rumah Sakit Santo Borromeus, dapat berjalan sampai Rumah Sakit Santo Immanuel, lalu menampilkan tulisan "Congratulations"	Berhasil
7	Berhasil menampilkan pemandangan di sekitar Cihampelas Walk, dapat berjalan sampai Rumah Sakit Santo Borromeus, lalu menampilkan tulisan "Congratulations"	Berhasil

Tabel 5.4: Hasil Pengujian pada Perangkat 2

Nomor Masukan	Hasil	Status
1	Berhasil menampilkan pemandangan di sekitar Universitas Katolik Parahyangan, dapat berjalan sampai Bandung Indah Plaza, lalu menampilkan tulisan "Congratulations"	Berhasil
2	Aplikasi terhenti tiba-tiba	Gagal
3	Aplikasi terhenti secara tiba-tiba	Gagal
4	Aplikasi terhenti secara tiba-tiba	Gagal
5	Aplikasi terhenti tiba-tiba	Gagal
6	Aplikasi terhenti tiba-tiba	Gagal
7	Berhasil menampilkan pemandangan di sekitar <i>Cihampelas Walk</i> , dapat berjalan sampai Rumah Sakit Santo Borromeus, lalu menampilkan tulisan "Congratulations"	Berhasil

Tabel 5.5: Hasil Pengujian pada Perangkat 3

Nomor Masukan	Hasil	Status
1	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal
2	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal
3	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal
4	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal
5	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal
6	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal
7	Aplikasi menampilkan lokasi asal, namun gambar berhenti	Gagal

### 5.3 Masalah yang Dihadapi

Adapun masalah-masalah yang dihadapi dalam proses pengembangan aplikasi adalah sebagai berikut:

1. Munculnya galat pada saat mencoba menggambar ulang bangun ruang silinder dengan tekstur baru. Galat tersebut adalah pemanggilan *OpenGL renderer* pada *thread* yang tidak memiliki *context*. Rencana awalnya adalah melakukan *load StreetView API* dari pemandangan selanjutnya pada saat sensor *step detector* mendeteksi langkah kaki, lalu mengubah tekstur bangun ruang silinder.
2. Peng-update-an *Gradle* di Android Studio membuat terjadinya *compile error* sehingga diperlukan waktu untuk memperbaiki galat ini dan menghambat perkembangan aplikasi.

```
2021-02-10 02:40:24.199 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2100
2021-02-10 02:40:24.820 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2200
2021-02-10 02:40:25.420 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2300
2021-02-10 02:40:26.041 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2400
2021-02-10 02:40:26.641 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2500
2021-02-10 02:40:27.241 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2600
2021-02-10 02:40:27.862 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2700
2021-02-10 02:40:28.502 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2800
2021-02-10 02:40:29.223 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2900
2021-02-10 02:40:29.803 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3000
2021-02-10 02:40:30.443 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3100
2021-02-10 02:40:31.064 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3200
2021-02-10 02:40:31.684 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3300
2021-02-10 02:40:32.265 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3400
2021-02-10 02:40:32.886 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3500
2021-02-10 02:40:33.486 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3600
2021-02-10 02:40:34.086 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3700
2021-02-10 02:40:34.706 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3800
2021-02-10 02:40:35.328 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3900
2021-02-10 02:40:35.907 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4000
2021-02-10 02:40:36.507 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4100
2021-02-10 02:40:37.168 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4200
2021-02-10 02:40:37.768 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4300
2021-02-10 02:40:38.389 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4400
2021-02-10 02:40:38.989 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4500
2021-02-10 02:40:39.569 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4600
2021-02-10 02:40:40.170 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4700
2021-02-10 02:40:40.730 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4800
2021-02-10 02:40:41.311 4497-4497/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4900
```

Gambar 5.9: *Log console* yang Menampilkan Jarak Tempuh Pengguna pada Perangkat 1

```
02-10 02:46:45.594 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2200
02-10 02:46:45.952 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2300
02-10 02:46:46.312 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2400
02-10 02:46:46.632 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2500
02-10 02:46:46.952 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2600
02-10 02:46:47.272 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2700
02-10 02:46:47.592 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2800
02-10 02:46:47.952 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 2900
02-10 02:46:48.272 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3000
02-10 02:46:48.632 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3100
02-10 02:46:48.953 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3200
02-10 02:46:49.272 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3300
02-10 02:46:49.552 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3400
02-10 02:46:49.912 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3500
02-10 02:46:50.232 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3600
02-10 02:46:50.552 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3700
02-10 02:46:50.872 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3800
02-10 02:46:51.193 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 3900
02-10 02:46:51.513 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4000
02-10 02:46:51.832 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4100
02-10 02:46:52.192 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4200
02-10 02:46:52.512 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4300
02-10 02:46:52.832 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4400
02-10 02:46:53.193 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4500
02-10 02:46:53.512 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4600
02-10 02:46:53.842 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4700
02-10 02:46:54.202 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4800
02-10 02:46:54.602 14984-14984/id.ac.unpar.informatika.homerunner D/Distance Elapsed: 4900
```

Gambar 5.10: *Log console* yang Menampilkan Jarak Tempuh Pengguna pada Perangkat 2

## BAB 6

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Dari penelitian yang telah dilakukan, kesimpulan-kesimpulan yang dapat ditarik adalah sebagai berikut:

1. Bagian Aplikasi HelloVR dari Google VR SDK dapat dimanfaatkan dengan mengubah bangun ruang dan tekstur gambar yang diinginkan. Cara memanfaatkan Aplikasi HelloVR adalah dengan mengubah bangun ruang pada file OBJ berbentuk silinder yang di-*bind* dengan tekstur gambar StreetView sekeliling.
2. Untuk menampilkan *Google StreetView API* pada *Google Cardboard*, beberapa gambar *StreetView* dari sekeliling (beberapa arah atau *heading*) harus disatukan, lalu ditampilkan pada bangun ruang yang terdapat pada Google VR SDK. Bab 4 menjelaskan langkah demi langkah mengenai pemanggilan *StreetView API* untuk mendapatkan gambar sekeliling dari lokasi tertentu. Setelah gambar itu selesai dibentuk, gambar sekeliling dari lokasi tertentu tersebut barulah di-*bind* dengan bangun ruang silinder yang dibuat.
3. Integrasi *Google Directions API*, *StreetView API*, *Google Cardboard*, dan sensor dapat dilakukan dengan memuat JSON *Directions API*, mem-parsenya, lalu menampilkan gambar *StreetView* dari sekeliling yang sudah digabungkan. Perubahan gambar *StreetView* pada Google VR bergantung pada rangsang yang diterima sensor *step detector*.

#### 6.2 Saran

Beberapa saran yang dapat diberikan untuk pengembangan:

1. Membedakan *StreetView VR* dengan metode yang sudah dipakai.
2. Mengimplementasi aplikasi dengan menggunakan *library* Google VR SDK yang tidak *obsolete*.
3. Mengimplementasi aplikasi tanpa harus memuat dan menyimpan gambar-gambar *StreetView* terlebih dahulu.



## **DAFTAR REFERENSI**

- [1] 2018-10-17 (2018) *Quickstart for Google VR SDK for Android / Google Developers*. Google. Mountain View, United States.
- [2] 2020-12-01 (2020) *Developer Guide / Street View Static API / Google Developers*. Google. Mountain View, United States.
- [3] 2020-11-19 (2020) *Developer Guide / Directions API / Google Developers*. Google. Mountain View, United States.
- [4] 2020-10-28 (2020) *Motion sensors / Android Developers*. Google. Mountain View, United States.
- [5] v1.190.0 (2019) *Releases - googlevr/gvr-android-sdk*. Github. Washington, United States.
- [6] 2018-10-29 (2020) *com.google.vr.sdk.base / Google VR / Google Developers*. Google. Mountain View, United States.
- [7] 2020-12-11 (2020) *Get an API Key and Signature / Street View Static API*. Google. Mountain View, United States.



## LAMPIRAN A

### KODE PROGRAM APLIKASI PENGUJIAN *STEP DETECTOR*

Listing A.1: StepDetectorSensor.java

```
1 package com.example.stepdetectorsensor;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.content.Context;
6 import android.hardware.Sensor;
7 import android.hardware.SensorEvent;
8 import android.hardware.SensorEventListener;
9 import android.hardware.SensorManager;
10 import android.os.Bundle;
11 import android.util.Log;
12
13 public class MainActivity extends AppCompatActivity implements SensorEventListener {
14
15     private SensorManager sensorManager;
16     private Sensor stepDetector;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         SensorManager sensorManager = sensorManager = (SensorManager) this.getSystemService(Context.SENSOR_SERVICE);
24         stepDetector = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);
25
26         sensorManager.registerListener(this, stepDetector, SensorManager.SENSOR_DELAY_FASTEST);
27     }
28
29     @Override
30     public void onSensorChanged(SensorEvent event) {
31         if (event.sensor == this.stepDetector)
32             Log.d("Step", "Step_Taken");
33     }
34
35     @Override
36     public void onAccuracyChanged(Sensor sensor, int accuracy) {
37
38     }
39 }
```



## LAMPIRAN B

### KODE PROGRAM APLIKASI JOGGING

Listing B.1: MainActivity.java

```
1 package id.ac.unpar.informatika.homerunner;
2
3
4 import android.app.Activity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9 import android.widget.TextView;
10 import android.widget.Toast;
11
12 public class MainActivity extends Activity implements View.OnClickListener {
13
14     protected EditText originET, destET;
15     protected TextView oriTV, destTV;
16     protected Button startButton;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         startButton = (Button) findViewById(R.id.startBtn);
24         originET = (EditText) findViewById(R.id.originET);
25         destET = (EditText) findViewById(R.id.destET);
26         oriTV = (TextView) findViewById(R.id.originTV);
27         destTV = (TextView) findViewById(R.id.destTV);
28
29         oriTV.setText("Origin");
30         destTV.setText("Destination");
31
32         startButton.setText("Start_Running");
33         startButton.setOnClickListener(this);
34     }
35
36     @Override
37     public void onClick(View view) {
38         if(view.getId() == startButton.getId()){
39
40             if (originET.getText().toString().equals("") || destET.getText().toString().equals(""))
41                 Toast.makeText(getApplicationContext(),"Origin_or_Destination_Text_Box_can't_be_Empty",
42                     Toast.LENGTH_SHORT).show();
43             else {
44
45                 DirectionsLoader dirLoader = new DirectionsLoader(this);
46
47                 String dirURL = "https://maps.googleapis.com/maps/api/directions/json?" +
48                     "mode=walking&origin=" + originET.getText().toString() + "&destination=" +
49                     destET.getText().toString() + "&key=" + getString(R.string.key);
50
51                 originET.setText("");
52                 destET.setText("");
53
54                 dirLoader.execute(dirURL);
55             }
56         }
57     }
58 }
59 }
```

Listing B.2: DirectionsLoader.java

```
1 package id.ac.unpar.informatika.homerunner;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.AsyncTask;
6 import java.io.BufferedReader;
7 import java.io.InputStream;
8 import java.io.InputStreamReader;
9 import java.net.HttpURLConnection;
10 import java.net.URL;
```

```
11 public class DirectionsLoader extends AsyncTask<String,Void(Void> {
12
13     protected String jsonText;
14     protected Activity activity;
15
16     public DirectionsLoader(Activity activity){
17         this.activity = activity;
18         jsonText = "";
19     }
20
21     @Override
22     protected Void doInBackground(String... strings) {
23         try {
24             URL url = new URL(strings[0]);
25             HttpURLConnection conn = (HttpURLConnection) url.openConnection();
26             conn.connect();
27             InputStream inputStream = conn.getInputStream();
28
29             InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
30             BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
31
32             String curLine = bufferedReader.readLine();
33
34             while(curLine != null){
35                 jsonText += curLine;
36                 curLine = bufferedReader.readLine();
37
38             }
39
40             conn.disconnect();
41
42         } catch (Exception ex) {}
43
44         return null;
45     }
46
47     @Override
48     protected void onPostExecute(Void aVoid) {
49         DirectionsExtractor dirExtractor = new DirectionsExtractor(jsonText);
50         dirExtractor.extractJSONDir();
51
52         Intent intent = new Intent(activity, VrActivity.class);
53
54         StreetViewLoader streetViewLoader = new StreetViewLoader(activity, intent, dirExtractor.arrSteps);
55         streetViewLoader.execute();
56     }
57 }
58 }
```

Listing B.3: DirectionsExtractor.java

```
1 package id.ac.unpar.informatika.homerunner;
2
3 import org.json.JSONArray;
4 import org.json.JSONObject;
5 import java.io.BufferedReader;
6 import java.io.StringReader;
7 import java.util.ArrayList;
8
9 public class DirectionsExtractor {
10
11     protected String jsonText;
12     protected ArrayList<JSONObject> arrSteps;
13
14     public DirectionsExtractor(String jsonText){
15         this.jsonText = jsonText;
16         arrSteps = new ArrayList<>();
17     }
18
19     public void extractJSONDir(){
20         String jsonTemp = "";
21         JSONObject jsonDir;
22
23         try {
24             BufferedReader bufferedReader = new BufferedReader(new StringReader(jsonText));
25             String curLine = bufferedReader.readLine();
26
27             while (curLine != null){
28                 jsonTemp += curLine;
29                 curLine = bufferedReader.readLine();
30             }
31
32             jsonDir = new JSONObject(jsonTemp);
33
34             JSONArray jsonArrRoute = jsonDir.getJSONArray("routes");
35
36             for(int i = 0; i < jsonArrRoute.length(); i++){
37                 JSONObject jsonRoute = jsonArrRoute.getJSONObject(i);
38
39                 JSONArray jsonArrLegs = jsonRoute.getJSONArray("legs");
40
41                 for(int j = 0; j < jsonArrLegs.length(); j++){
42                     JSONObject jsonLegs = jsonArrLegs.getJSONObject(j);
43
44
45                     JSONArray jsonArrSteps = jsonLegs.getJSONArray("steps");
46
47                     for(int k = 0; k < jsonArrSteps.length(); k++) {
48                         JSONObject jsonStep = jsonArrSteps.getJSONObject(k);
49
50                         arrSteps.add(jsonStep);
51                     }
52                 }
53             }
54         } catch (Exception e) {
55             e.printStackTrace();
56         }
57     }
58
59     public ArrayList<JSONObject> getArrSteps() {
60         return arrSteps;
61     }
62 }
```

```

48             arrSteps.add(jsonArrSteps.getJSONObject(k));
49         }
50     }
51 }catch (Exception ex){
52 }
53 }
54 }
55 }
56 }

```

Listing B.4: StreetViewLoader.java

```

1 package id.ac.unpar.informatika.homerunner;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.graphics.Bitmap;
6 import android.graphics.BitmapFactory;
7 import android.graphics.Canvas;
8 import android.os.AsyncTask;
9 import org.json.JSONObject;
10 import java.io.File;
11 import java.io.FileOutputStream;
12 import java.io.InputStream;
13 import java.net.URL;
14 import java.util.ArrayList;
15
16 public class StreetViewLoader extends AsyncTask<Void, Void, Void> {
17
18     protected Bitmap streetViewBitmap;
19     protected Activity activity;
20     protected Intent intent;
21     protected ArrayList<JSONObject> arrSteps;
22     protected int imgCount;
23
24     public StreetViewLoader(Activity activity, Intent intent, ArrayList<JSONObject> arrSteps) {
25         this.intent = intent;
26         this.activity = activity;
27         this.arrSteps = arrSteps;
28         this.imgCount = 0;
29     }
30
31     @Override
32     protected Void doInBackground(Void... aVoid) {
33         for(int i = 0; i < arrSteps.size(); i++){
34             int length = 4;
35
36             boolean isEnd = (i == arrSteps.size() - 1 ? true : false);
37
38             createStreetViewImage(generateStreetViewURL(length, true));
39             publishProgress();
40
41             if(isEnd) {
42                 createStreetViewImage(generateStreetViewURL(length, false));
43                 publishProgress();
44             }
45         }
46
47         return null;
48     }
49
50     @Override
51     protected void onProgressUpdate(Void... values) {
52         imgCount++;
53     }
54
55     @Override
56     protected void onPostExecute(Void aVoid) {
57         intent.putExtra("length", imgCount);
58
59         int[] stepsDistance = new int[arrSteps.size()];
60         for (int i = 0; i < arrSteps.size(); i++){
61             try{
62                 stepsDistance[i] = arrSteps.get(i).getJSONObject("distance").getInt("value");
63             } catch (Exception ex){}
64         }
65
66         intent.putExtra("stepDistances", stepsDistance);
67
68         activity.startActivity(intent);
69     }
70
71     protected String[] generateStreetViewURL(int svUrlLength, boolean isStart){
72         int heading = 0;
73
74         String[] urlArr = new String[svUrlLength];
75
76         String location = (isStart ? getLocation("start_location") : getLocation("end_location"));
77
78         String svTempURL = "https://maps.googleapis.com/maps/api/streetview?size=600x300&location=" +
79             location + "&key=" + activity.getString(R.string.key) + "&heading=";
80
81         for(int i = 0 ; i < svUrlLength ; i++){
82             urlArr[i] = svTempURL + heading;
83             heading += 90;
84         }
85
86         return urlArr;

```

```

87 }
88
89     public String generateFilePath() {
90         return ("streetview" + imgCount + ".png");
91     }
92
93     public String getLocation(String startOrEndKey){
94         double startLoc = 0.0;
95         double endLoc = 0.0;
96         try {
97             startLoc = arrSteps.get(imgCount).getJSONObject(startOrEndKey).getDouble("lat");
98             endLoc = arrSteps.get(imgCount).getJSONObject(startOrEndKey).getDouble("lng");
99         } catch (Exception ex){}
100
101        return startLoc + "," + endLoc;
102    }
103
104    public void createStreetViewImage(String[] urls){
105        Bitmap[] bitmaps = new Bitmap[url.length];
106        for (int j = 0; j < urls.length; j++) {
107            try {
108                URL url = new URL(urls[j]);
109                InputStream input = url.openStream();
110                bitmaps[j] = BitmapFactory.decodeStream(input);
111            } catch (Exception ex) {}
112        }
113
114        int width = bitmaps[0].getWidth();
115        int height = bitmaps[0].getHeight();
116
117        width *= bitmaps.length;
118
119        Bitmap allBitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
120        Canvas canvas = new Canvas(allBitmap);
121        int left = 0;
122        for (int j = 0; j < bitmaps.length; j++) {
123            left = (j == 0 ? 0 : left + bitmaps[j].getWidth());
124            canvas.drawBitmap(bitmaps[j], left, 0, null);
125        }
126        streetViewBitmap = allBitmap;
127
128        try {
129            File file = new File(activity.getCacheDir(), generateFilePath());
130            if (file.exists()) {
131                file.delete();
132            file = new File(activity.getCacheDir(), generateFilePath());
133            }
134            FileOutputStream fos = new FileOutputStream(file);
135            streetViewBitmap.compress(Bitmap.CompressFormat.PNG, 0, fos);
136            fos.flush();
137            fos.close();
138        } catch (Exception e) {}
139    }
140}
141

```

Listing B.5: VrActivity.java

```

1 package id.ac.unpar.informatika.homerunner;
2
3 import android.content.Context;
4 import android.hardware.Sensor;
5 import android.hardware.SensorEvent;
6 import android.hardware.SensorEventListener;
7 import android.hardware.SensorManager;
8 import android.opengl.GLES20;
9 import android.opengl.Matrix;
10 import android.os.Bundle;
11 import android.util.Log;
12
13 import com.google.vr.ndk.base.Properties;
14 import com.google.vr.ndk.base.Properties.PropertyType;
15 import com.google.vr.ndk.base.Value;
16 import com.google.vr.sdk.base.AndroidCompat;
17 import com.google.vr.sdk.base.Eye;
18 import com.google.vr.sdk.base.GvrActivity;
19 import com.google.vr.sdk.base.GvrView;
20 import com.google.vr.sdk.base.HeadTransform;
21 import com.google.vr.sdk.base.Viewport;
22
23 import java.io.File;
24 import java.io.IOException;
25 import java.util.ArrayList;
26
27 import javax.microedition.khronos.egl.EGLConfig;
28
29 public class VrActivity extends GvrActivity implements GvrView.StereoRenderer, SensorEventListener {
30
31     private static final float Z_NEAR = 0.01f;
32     private static final float Z_FAR = 10.0f;
33
34     private static final float[] POS_MATRIX_MULTIPLY_VEC = {0.0f, 0.0f, 0.0f, 1.0f};
35     private static final float[] FORWARD_VEC = {0.0f, 0.0f, -1.0f, 1.0f};
36
37     private static final float DEFAULT_FLOOR_HEIGHT = -1.6f;
38
39     private static final String[] OBJECT_VERTEX_SHADER_CODE =
40         new String[] {

```

```

41         "uniform_mat4_u_MVP;",
42         "attribute_vec4_a_Position;",
43         "attribute_vec2_a_UV;",
44         "varying_vec2_v_UV;",
45         "",
46         "void_main() {",
47         "    v_UV = a_UV;";
48         "    gl_Position = u_MVP * a_Position;",
49         "}",
50     };
51     private static final String[] OBJECT_FRAGMENT_SHADER_CODE =
52     new String[] {
53         "precision mediump float;",
54         "varying vec2 v_UV;",
55         "uniform sampler2D u_Texture;",
56         "",
57         "void_main() {",
58         "    // The y coordinate of this sample's textures is reversed compared to",
59         "    // what OpenGL expects, so we invert the y coordinate.",
60         "    gl_FragColor = texture2D(u_Texture, vec2(v_UV.x, 1.0 - v_UV.y));",
61         "}",
62     };
63 }
64 protected int objectProgram;
65
66 protected int objectPositionParam;
67 protected int objectUvParam;
68 protected int objectModelViewProjectionParam;
69
70 private float[] camera;
71 private float[] view;
72 private float[] headView;
73 private float[] modelViewProjection;
74 private float[] modelView;
75
76 private float[] modelRoom;
77
78 private float[] headRotation;
79
80 private Properties gvrProperties;
81
82 protected TexturedMesh room;
83 protected ArrayList<Texture> roomTex;
84 protected Texture finishedTexture;
85
86 private final Value floorHeight = new Value();
87
88 private SensorManager sensorManager;
89 private Sensor stepDetector;
90
91 private static final int DISTANCE_PER_STEP = 100;
92
93 private int distanceElapsed;
94 private int curStepIndex;
95 private int curStepDistance;
96 private int[] stepsDistance;
97
98 @Override
99 public void onCreate(Bundle savedInstanceState) {
100     super.onCreate(savedInstanceState);
101
102     sensorManager = (SensorManager) this.getSystemService(Context.SENSOR_SERVICE);
103     stepDetector = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR);
104
105     sensorManager.registerListener(this, stepDetector, SensorManager.SENSOR_DELAY_FASTEST);
106
107     distanceElapsed = 0;
108     curStepIndex = 0;
109
110     roomTex = new ArrayList<>();
111
112     stepsDistance = getIntent().getIntArrayExtra("stepDistances");
113
114     curStepDistance = stepsDistance[curStepIndex];
115
116     initializeGvrView();
117
118     camera = new float[16];
119     view = new float[16];
120     modelViewProjection = new float[16];
121     modelView = new float[16];
122
123     headRotation = new float[4];
124     modelRoom = new float[16];
125     headView = new float[16];
126
127     if (sensorManager.getDefaultSensor(Sensor.TYPE_STEP_DETECTOR) != null) {
128         Log.d("Step_Detector", "This_device_has_it");
129     } else {
130         Log.d("Step_Detector", "This_device_don't_have_it");
131     }
132 }
133
134 public void initializeGvrView() {
135     setContentView(R.layout.activity_vr);
136
137     GvrView gvrView = findViewById(R.id.gvr_view);
138
139     gvrView.setRenderer(this);

```

```

140 gvrView.setTransitionViewEnabled(true);
141 gvrView.enableCardboardTriggerEmulation();
142
143 if (gvrView.setAsyncReprojectionEnabled(true)) {
144     // Async reprojection decouples the app framerate from the display framerate,
145     // allowing immersive interaction even at the throttled clockrates set by
146     // sustained performance mode.
147     AndroidCompat.setSustainedPerformanceMode(this, true);
148 }
149
150 setGvrView(gvrView);
151 gvrProperties = gvrView.getGvrApi().getCurrentProperties();
152 }
153
154
155 @Override
156 public void onPause() {
157     super.onPause();
158 }
159
160 @Override
161 public void onResume() {
162     super.onResume();
163 }
164
165 @Override
166 public void onRendererShutdown() {
167     floorHeight.close();
168 }
169
170 @Override
171 public void onSurfaceChanged(int width, int height) {}
172
173
174 @Override
175 public void onSurfaceCreated(EGLConfig config) {
176     GLES20.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
177
178     objectProgram = Util.compileProgram(OBJECT_VERTEX_SHADER_CODE, OBJECT_FRAGMENT_SHADER_CODE);
179
180     objectPositionParam = GLES20.glGetAttribLocation(objectProgram, "a_Position");
181     objectUvParam = GLES20.glGetAttribLocation(objectProgram, "a_UV");
182     objectModelViewProjectionParam = GLES20.glGetUniformLocation(objectProgram, "u_MVP");
183
184     Util.checkGlError("Object_program_params");
185
186     Matrix.setIdentityM(modelRoom, 0);
187     Matrix.translateM(modelRoom, 0, 0, DEFAULT_FLOOR_HEIGHT, 0);
188
189     int imgLength = getIntent().getIntExtra("length", 0);
190
191     try {
192         finishedTexture = new Texture(this, "finish_image.png", false);
193     } catch (IOException e) {
194     }
195
196     for (int i = 0; i < imgLength; i++) {
197         try {
198             room = new TexturedMesh(this, "Room.obj", objectPositionParam, objectUvParam);
199             roomTex.add(new Texture(this, "streetview" + i + ".png", true));
200         } catch (IOException e) {}
201     }
202 }
203
204
205 @Override
206 public void onNewFrame(HeadTransform headTransform) {
207     // Build the camera matrix and apply it to the ModelView.
208     Matrix.setLookAtM(camera, 0, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f, -1.0f, 0.0f, 1.0f, 0.0f);
209
210     if (gvrProperties.getPropertyType(TRACKING_FLOOR_HEIGHT, floorHeight)) {
211         // The floor height can change each frame when tracking system detects a new floor position.
212         Matrix.setIdentityM(modelRoom, 0);
213         Matrix.translateM(modelRoom, 0, 0, floorHeight.asFloat(), 0);
214     } // else the device doesn't support floor height detection so DEFAULT_FLOOR_HEIGHT is used.
215
216     headTransform.getHeadView(headView, 0);
217
218     headTransform.getQuaternion(headRotation, 0);
219     Util.checkGlError("onNewFrame");
220 }
221
222
223 @Override
224 public void onDrawEye(Eye eye) {
225     GLES20.glEnable(GLES20.GL_DEPTH_TEST);
226     // The clear color doesn't matter here because it's completely obscured by
227     // the room. However, the color buffer is still cleared because it may
228     // improve performance.
229     GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT | GLES20.GL_DEPTH_BUFFER_BIT);
230
231     // Apply the eye transformation to the camera.
232     Matrix.multiplyMM(view, 0, eye.getEyeView(), 0, camera, 0);
233
234     // Build the ModelView and ModelViewProjection matrices
235     // for calculating the position of the target object.
236     float[] perspective = eye.getPerspective(Z_NEAR, Z_FAR);
237
238     Matrix.multiplyMM(modelViewProjection, 0, perspective, 0, modelView, 0);

```

```

239 // Set modelView for the room, so it's drawn in the correct location
240 Matrix.multiplyMM(modelView, 0, view, 0, modelRoom, 0);
241 Matrix.multiplyMM(modelViewProjection, 0, perspective, 0, modelView, 0);
242 drawRoom();
243 }
244
245 @Override
246 public void onFinishFrame(Viewport viewport) {
247     File cacheDir = getCacheDir();
248     emptyDir(cacheDir);
249 }
250
251 public static boolean emptyDir(File dir) {
252     if (dir.isDirectory()) {
253         String[] children = dir.list();
254         for (int i = 0; i < children.length; i++) {
255             if (!emptyDir(new File(dir, children[i])))
256                 return false;
257         }
258     }
259     return dir.delete();
260 }
261
262 public void drawRoom() {
263     GLES20.glUseProgram(objectProgram);
264     GLES20.glUniformMatrix4fv(objectModelViewProjectionParam, 1, false, modelViewProjection, 0);
265     if (curStepIndex >= roomTex.size())
266         finishedTexture.bind();
267     else
268         roomTex.get(curStepIndex).bind();
269     room.draw();
270     Util.checkGLError("drawRoom");
271 }
272
273 @Override
274 public void onCardboardTrigger() {}
275
276 @Override
277 public void onSensorChanged(SensorEvent sensorEvent) {
278     if(sensorEvent.sensor == this.stepDetector) {
279         if (curStepIndex <= stepsDistance.length) {
280             distanceElapsed += DISTANCE_PER_STEP;
281             if (distanceElapsed >= curStepDistance) {
282                 curStepIndex++;
283                 if (curStepIndex < stepsDistance.length-1)
284                     curStepDistance += stepsDistance[curStepIndex];
285             }
286         }
287         Log.d("Distance_Elapsed", ""+distanceElapsed);
288     }
289 }
290
291 @Override
292 public void onAccuracyChanged(Sensor sensor, int i) {}
293 }
294
295 //Portions of this page are modifications based on work created and shared by Google
296 //and used according to terms described in the Creative Commons 4.0 Attribution License.

```

Listing B.6: TexturedMesh.java

```

1 package id.ac.unpar.informatika.homerunner;
2
3 import android.content.Context;
4 import android.opengl.GLES20;
5 import de.javagl.obj.Obj;
6 import de.javagl.obj.ObjData;
7 import de.javagl.obj.ObjReader;
8 import de.javagl.obj.ObjUtils;
9 import java.io.IOException;
10 import java.io.InputStream;
11 import java.nio.ByteBuffer;
12 import java.nio.ByteOrder;
13 import java.nio.FloatBuffer;
14 import java.nio.IntBuffer;
15 import java.nio.ShortBuffer;
16
17 class TexturedMesh {
18     private static final String TAG = "TexturedMesh";
19
20     private final FloatBuffer vertices;
21     private final FloatBuffer uv;
22     private final ShortBuffer indices;
23     private final int positionAttrib;
24     private final int uvAttrib;
25
26     public TexturedMesh(Context context, String objFilePath, int positionAttrib, int uvAttrib)
27         throws IOException {
28         InputStream objInputStream = context.getAssets().open(objFilePath);
29         Obj obj = ObjUtils.convertToRenderable(ObjReader.read(objInputStream));
30         objInputStream.close();
31
32         IntBuffer intIndices = ObjData.getFaceVertexIndices(obj, 3);
33         vertices = ObjData.getVertices(obj);
34         uv = ObjData.getTexCoords(obj, 2);
35
36         indices =
37             ByteBuffer.allocateDirect(2 * intIndices.limit())

```

```

38     .order(ByteOrder.nativeOrder())
39     .asShortBuffer();
40   while (intIndices.hasRemaining()) {
41     indices.put((short) intIndices.get());
42   }
43   indices.rewind();
44
45   this.positionAttrib = positionAttrib;
46   this.uvAttrib = uvAttrib;
47 }
48
49 public void draw() {
50   GLES20.glEnableVertexAttribArray(positionAttrib);
51   GLES20.glVertexAttribPointer(positionAttrib, 3, GLES20.GL_FLOAT, false, 0, vertices);
52   GLES20.glEnableVertexAttribArray(uvAttrib);
53   GLES20.glVertexAttribPointer(uvAttrib, 2, GLES20.GL_FLOAT, false, 0, uv);
54   GLES20.glDrawElements(GLES20.GL_TRIANGLES, indices.limit(), GLES20.GL_UNSIGNED_SHORT, indices);
55 }
56 }
57 }

```

Listing B.7: Util.java

```

1 package id.ac.unpar.informatika.homerunner;
2
3 import static android.opengl.GLU.gluErrorString;
4
5 import android.opengl.GLES20;
6 import android.opengl.Matrix;
7 import android.text.TextUtils;
8 import android.util.Log;
9
10 class Util {
11   private static final String TAG = "Util";
12
13   private static final boolean HALT_ON_GL_ERROR = true;
14
15   private Util() {}
16
17   public static void checkGlError(String label) {
18     int error = GLES20.glGetError();
19     int lastError;
20     if (error != GLES20.GL_NO_ERROR) {
21       do {
22         lastError = error;
23         Log.e(TAG, label + ":_glError_" + gluErrorString(lastError));
24         error = GLES20.glGetError();
25       } while (error != GLES20.GL_NO_ERROR);
26
27       if (HALT_ON_GL_ERROR) {
28         throw new RuntimeException("glError_" + gluErrorString(lastError));
29       }
30     }
31   }
32 }
33
34 public static int compileProgram(String[] vertexCode, String[] fragmentCode) {
35   checkGlError("Start_of_compileProgram");
36
37   int vertexShader = GLES20.glCreateShader(GLES20.GL_VERTEX_SHADER);
38   GLES20.glShaderSource(vertexShader, TextUtils.join("\n", vertexCode));
39   GLES20.glCompileShader(vertexShader);
40   checkGlError("Compile_vertex_shader");
41
42   int fragmentShader = GLES20.glCreateShader(GLES20.GL_FRAGMENT_SHADER);
43   GLES20.glShaderSource(fragmentShader, TextUtils.join("\n", fragmentCode));
44   GLES20.glCompileShader(fragmentShader);
45   checkGlError("Compile_fragment_shader");
46
47   int program = GLES20.glCreateProgram();
48   GLES20.glAttachShader(program, vertexShader);
49   GLES20.glAttachShader(program, fragmentShader);
50
51   GLES20.glLinkProgram(program);
52   int[] linkStatus = new int[1];
53   GLES20.glGetProgramiv(program, GLES20.GL_LINK_STATUS, linkStatus, 0);
54   if (linkStatus[0] != GLES20.GL_TRUE) {
55     String errorMsg = "Unable_to_link_shader_program:_\n" + GLES20.glGetProgramInfoLog(program);
56     Log.e(TAG, errorMsg);
57     if (HALT_ON_GL_ERROR) {
58       throw new RuntimeException(errorMsg);
59     }
60   }
61   checkGlError("End_of_compileProgram");
62
63   return program;
64 }
65
66 public static float angleBetweenVectors(float[] vec1, float[] vec2) {
67   float cosOfAngle = dotProduct(vec1, vec2) / (vectorNorm(vec1) * vectorNorm(vec2));
68   return (float) Math.acos(Math.max(-1.0f, Math.min(1.0f, cosOfAngle)));
69 }
70
71 private static float dotProduct(float[] vec1, float[] vec2) {
72   return vec1[0] * vec2[0] + vec1[1] * vec2[1] + vec1[2] * vec2[2];
73 }
74
75 private static float vectorNorm(float[] vec) {

```

```
76|     return Matrix.length(vec[0], vec[1], vec[2]);  
77| }  
78| }
```