

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

«**penguji 1**»

«**penguji 2**»

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

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »

Bandung, «bulan» 2020

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
1 PENDAHULUAN	1
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
2 LANDASAN TEORI	5
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>	12
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>	13
2.3 Google <i>Directions API</i>	14
2.3.1 Penggunaan <i>Directions API</i>	14
2.3.2 Hasil Pemanggilan <i>Directions API</i>	16
2.4 Motion Sensor	21
2.4.1 Deskripsi <i>Motion Sensor</i>	21
2.4.2 Deskripsi <i>Step Detector</i>	21
3 ANALISIS	23
3.1 Analisis Google VR SDK	23
3.2 Analisis Pembuatan Bangun Ruang Tiga Dimensi	24
3.3 Menampilkan <i>StreetView API</i> pada Bangun Ruang Silinder	24
3.3.1 Membentuk Gambar StreetView untuk Tekstur Ruang Tiga Dimensi	24
3.4 Analisis Google <i>Directions API</i>	25
3.4.1 Menentukan Atribut yang akan Digunakan	25
3.4.2 Cara Memanfaatkan Atribut	29
3.5 Cara Memanfaatkan <i>Step Detector</i> Sensor	29
4 RANCANGAN	31
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
5	IMPLEMENTASI DAN PENGUJIAN	41
5.1	Implementasi	41
5.1.1	Lingkungan Implementasi	41
5.1.2	Hasil Implementasi	41
5.1.3	Aplikasi Setelah Pengguna Menyelesaikan Perjalannya	43
5.2	Pengujian	43
5.3	Masalah yang Dihadapi	43
6	KESIMPULAN DAN SARAN	45
6.1	Kesimpulan	45
6.2	Saran	45
DAFTAR REFERENSI		47

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	Struktur Direktori Aplikasi HelloVR	10
2.4	Gambar-gambar <i>assets</i> aplikasi HelloVR	10
2.5	Tampilan <i>UI Google Cloud</i> saat mengakses <i>API Key (API Key disamaraskan)</i>	12
2.6	Pemanggilan <i>StreetView API</i> yang berhasil	15
3.1	Isi <i>folder assets</i> aplikasi HelloVR	23
3.2	Tampilan <i>UI Blender</i> Blender versi 2.81	24
3.3	Pemanggilan <i>StreetView API</i> yang berhasil, dengan <i>URL https://maps.googleapis.com/maps/api/streetview?size=600x300&location=-6.8746537,107.6046282&key=(disamaraskan)</i> , dengan parameter <i>heading</i> yang berbeda-beda-jelaskan di bab 2 tentang keluaran directions.	25
3.4	Contoh hasil penggabungan empat gambar <i>StreetView</i>	26
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	Tampilan <i>Google Cardboard</i> sebelum gawai diputar	42
5.3	Tampilan VR saat Pengguna Berlari	43
5.4	Tampilan VR saat Pengguna Mencapai Tujuan	44

DAFTAR TABEL

5.1 Tabel Perangkat yang digunakan	43
--	----

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Zaman modern adalah zaman saat profesi sudah dan sedang berkembang sehingga ada banyak sekali jumlah bidangnya serta perkembangannya. Mayoritas orang menekuni bidang-bidang profesi yang tak terhitung banyaknya untuk mengembangkan setiap bidang profesi. Hal ini 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 melakuannya, 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.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat menampilkan pemandangan saat berlari pada lingkungan yang diinginkan saat berlari di *treadmill*. 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.

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, dan *Google Directions API* yang digunakan untuk menentukan 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 [7].

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* 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* dan Google VR (*Cardboard*) dalam perangkat lunak *virtual jogging*.

1.4 Batasan Masalah

Karena ada banyak tempat atau pemandangan di dunia ini, hanya beberapa lokasi saja yang dapat dipilih dari yang telah disediakan.

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

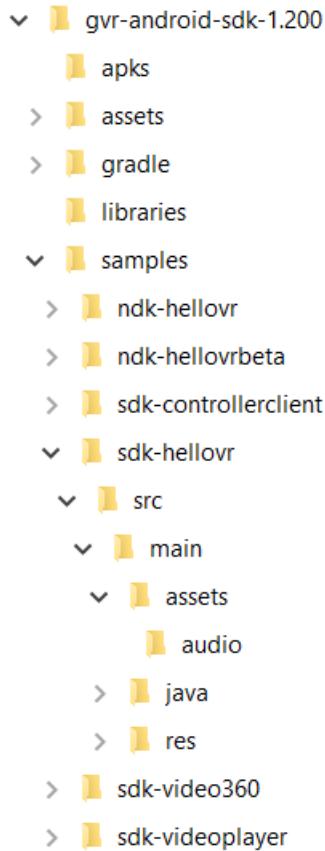
Google VR adalah teknologi yang diciptakan perusahaan Google untuk membuat pemandangan dunia maya yang terlihat nyata dengan menempatkan *smartphone* pada alat yang bernama *viewer* [1]. *Viewer* adalah alat untuk melihat dunia VR pada aplikasi VR di *smartphone*. Dua *viewer* yang diciptakan Google adalah *Google Daydream* dan *Cardboard*. 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 repository Github yang berisi kode program dari aplikasi *virtual reality* (VR) yang tersedia untuk Android (Java), Android NDK, Unity, dan iOS [4]. 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 [5]. *Library* ini juga memanfaatkan *OpenGL*, yaitu *API* untuk mengolah grafika komputer. Berikut adalah beberapa *class* dan *interface* dari *package com.google.vr.sdk.base*:

1. *GvrView.StereoRenderer*



Gambar 2.1: Struktur Direktori Google VR SDK

Interface untuk *renderer* yang menyerahkan seluruh penyajian pemandangan stereo pada pemandangan (*view*). *Method-emethod* abstrak yang dimiliki *interface* imi 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 menopang performanya.
- `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.
- `public static boolean trySetVrModeEnabled (Activity activity, boolean enabled)`
Sets the appropriate "VR mode" setting for an Activity. Parameter yang dimiliki

3. Eye

Class yang mendefinisikan rincian *rendering* stereo dari satu *eye*.

- `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 4×4 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.

Atribut-atribut yang dimiliki kelas ini adalah sebagai berikut:

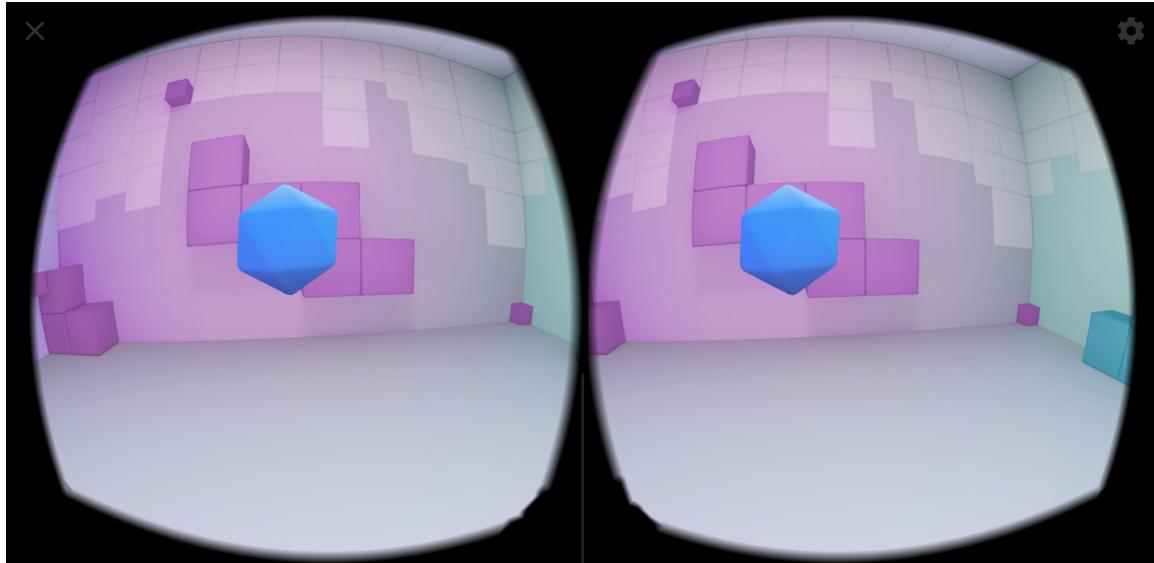
- **public int height**
 Tinggi dari *viewport*.
- **public int width** Lebar dari *viewport*.
- **public int x**
 Koordinat sumbu x titik pada *viewport*, dengan koordinat (0,0) berada pada sudut kiri bawah.
- **public int y**
 Koordinat sumbu y titik pada *viewport*, dengan koordinat (0,0) berada pada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

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 2.3 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.

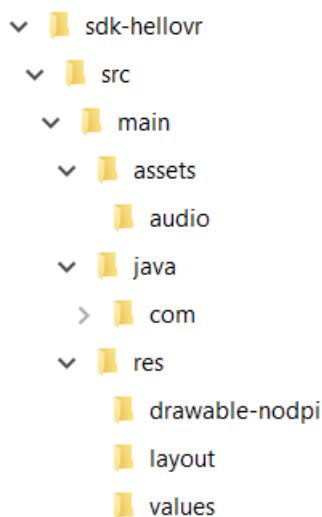
Dunia VR pada aplikasi ini terbentuk dari komponen-komponen yang ada dalam *folder assets*. Bentuk ruangan VR ditentukan oleh *file Wavefront Object* (OBJ) dan tampilannya oleh tekstur *file Portable Network Graphics* (PNG) (Gambar 2.4a) yang telah dengan sangat tepat dipetakan pada *file OBJ* yang ada 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 2.4b) digunakan ketika bentuk sedang tidak ada di tengah-tengah titik pengelihatan pengguna, sedangkan satu tekstur yang lain (Gambar 2.4c) digunakan ketika pengguna sedang melihat bentuk tepat di titik tengah pengelihatan pengguna.

Rancangan kelas Aplikasi HelloVR

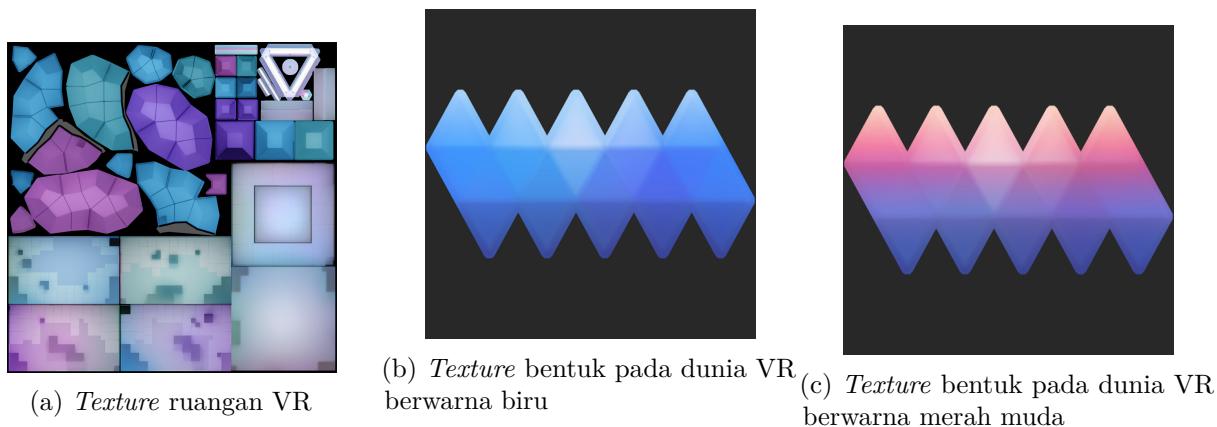
Aplikasi HelloVR memiliki empat kelas pada programnya, di antaranya:

1. Texture

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



Gambar 2.3: Struktur Direktori Aplikasi HelloVR



Gambar 2.4: 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*.

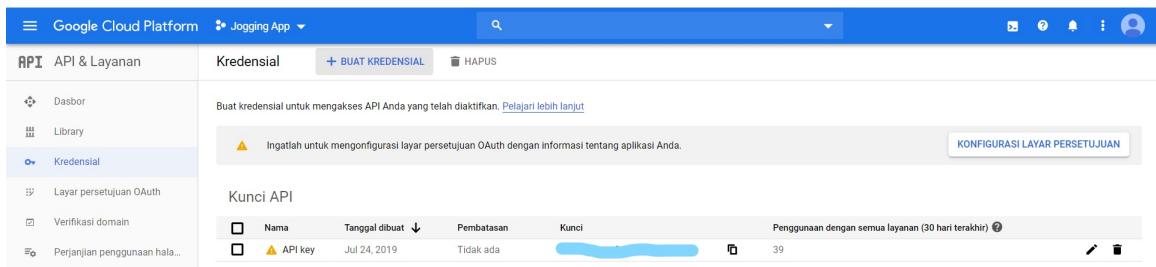
- `private int objectPositionParam`
Atribut dari *id* parameter posisi dari objek-objek dalam dunia tiga dimensi.
- `private int objectUvParam`
Atribut dari *id* parameter koordinat tekstur dari objek-objek dunia tiga dimensi.
- `private int objectModelViewProjectionParam`
Atribut dari *id* parameter 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 pemandangan sesuai masukan pengguna melalui *HTTP request* [2]. 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 2.5: 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 2.5 menunjukkan tampilan *Google Cloud* setelah mendapatkan *API key* [6]. *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 horisontal, 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 *StreetView API*

Pemanggilan *StreetView* yang berhasil akan menghasilkan sebuah gambar dari pemandangan sesuai lokasi. Gambar 2.6 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)).

2.3 Google Directions API

Google *Directions API* adalah layanan berbasis *HTTP/HTTPS* dari Google yang membantu mencari dan menghitung arah dari satu tempat ke tempat yang lain [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>



Gambar 2.6: Pemanggilan *StreetView API* yang berhasil

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" : [
    {
      "distance" : 1000,
      "duration" : 1000,
      "legs" : [
        {
          "distance" : 500,
          "duration" : 500,
          "end_address" : "Jl. Prof. Dr. Satrio No. 10, Bandung, Jawa Barat 40132, Indonesia",
          "end_location" : { "lat": -6.916667, "lng": 107.616667 },
          "start_address" : "Jl. Prof. Dr. Satrio No. 10, Bandung, Jawa Barat 40132, Indonesia",
          "start_location" : { "lat": -6.916667, "lng": 107.616667 }
        }
      ],
      "start_address" : "Jl. Prof. Dr. Satrio No. 10, Bandung, Jawa Barat 40132, Indonesia",
      "start_location" : { "lat": -6.916667, "lng": 107.616667 }
    }
  ]
}
```

```
        }
    ],
"routes" : [
{
    "bounds" : {
        "northeast" : {
            "lat" : -6.8746719,
            "lng" : 107.6137497
        },
        "southwest" : {
            "lat" : -6.893148,
            "lng" : 107.6034915
        }
    },
    "copyrights" : "Map\u00a9data\u00a92020",
    "legs" : [
{
        "distance" : {
            "text" : "3.0\u00a9km",
            "value" : 2980
        },
        "duration" : {
            "text" : "35\u00a9mins",
            "value" : 2116
        },
        "end_address" : "Jl.\u00a9Ir.\u00a9H.\u00a9Juanda\u00a9No.100,\u00a9Lebakgede . . .",
        "end_location" : {
            "lat" : -6.893148,
            "lng" : 107.6131791
        },
        "start_address" : "Jl.\u00a9Ciumbuleuit\u00a9No.94,\u00a9Hegarmanah . . .",
        "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\u00a9\u003cb\u003esouth . . .",
            "polyline" : {

```

```

        "points" : "tu}h@yowoSdAP|AL‘Cb@dAHHBvC... "
    },
    "start_location" : {
        "lat" : -6.8746719,
        "lng" : 107.6046127
    },
    "travel_mode" : "WALKING"
},
{
    "distance" : {
        "text" : "1.1\u00a0km",
        "value" : 1078
    },
    "duration" : {
        "text" : "14\u00a0mins",
        "value" : 834
    },
    "end_location" : {
        "lat" : -6.88521839999999,
        "lng" : 107.6137497
    },
    "html_instructions" : "Turn\u20a6\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\u00a0km",
        "value" : 884
    },
    "duration" : {
        "text" : "10\u00a0mins",
        "value" : 603
    },
    "end_location" : {
        "lat" : -6.89314079999999,
        "lng" : 107.6130843
    },
    "html_instructions" : "Turn\u20a6\u003cb\u003eright...",
    "maneuver" : "turn-right",
    "polyline" : {
        "points" : "rw_i@}hyoSzCLIAHz@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" : "bhai@wdyoS@S"
    },
    "start_location" : {
        "lat" : -6.893140799999999,
        "lng" : 107.6130843
    },
    "travel_mode" : "WALKING"
}
],
"traffic_speed_entry" : [],
"via_waypoint" : []
}
],
"overview_polyline" : {
    "points" : "tu}\u003ch@yowoSdAP|AL'Cb@dAH'Dd@~Bd@hG..."
},
"summary" : "Jl.\u2022 Ciumbuleuit ,\u2022 Jl.\u2022 Siliwangi ...",
"warnings" : [
    "Walking\u2022 directions\u2022 are\u2022 in\u2022 beta.\u2022 Use\u2022 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 jalan yang dilewati.
 - * *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*.
- *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* [7]. 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.

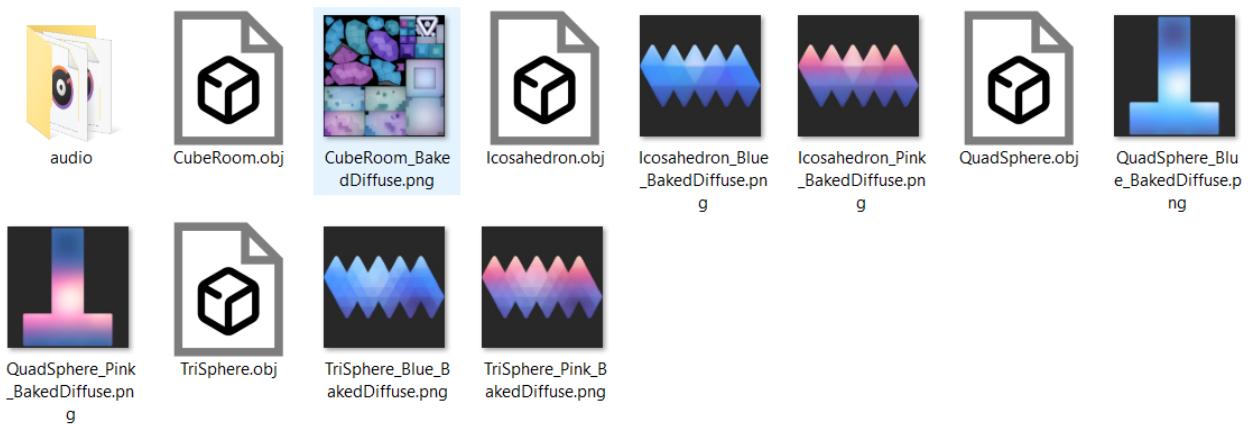
BAB 3

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 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]. Gambar 3.1 menunjukkan rincian isi *folder assets* dari aplikasi HelloVR.



Gambar 3.1: Isi *folder assets* aplikasi HelloVR

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-teksutur 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

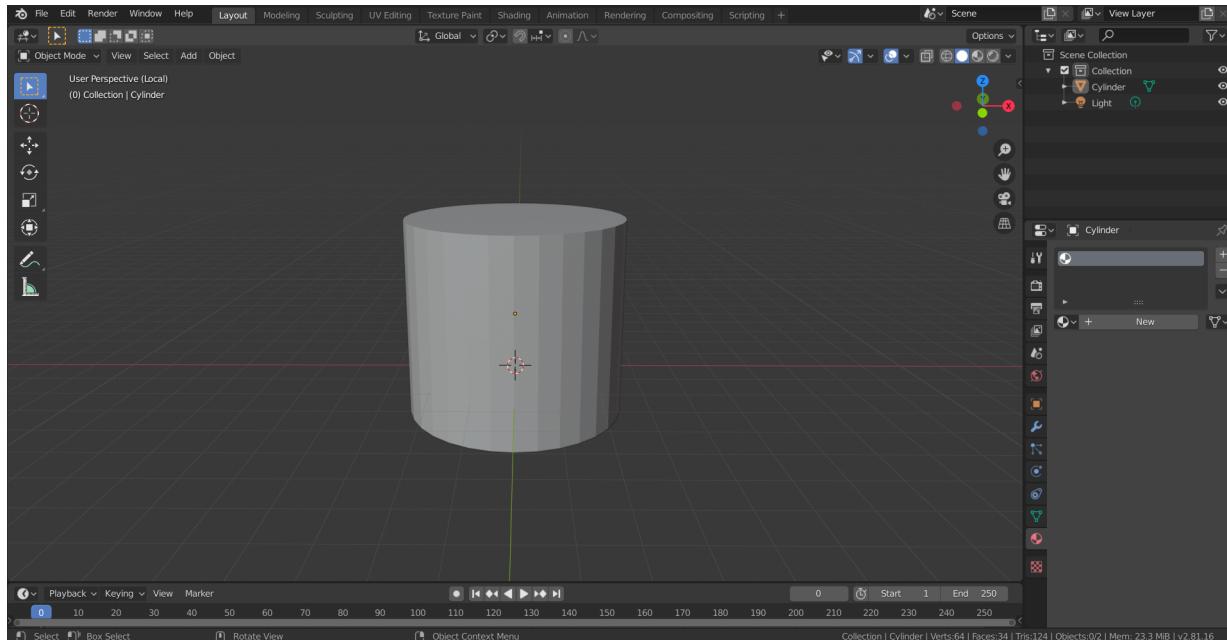
yang dapat digunakan untuk menentukan rute perjalanan dari asal dan tujuan yang dapat digunakan untuk perubahan gambar tekstur untuk bangun ruang tiga dimensi.

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*.

3.2 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, kakas yang digunakan adalah *Blender* versi 2.81. Gambar menunjukkan tampilan *UI Blender* versi 2.81. Setelah bangun ruang silinder dibentuk dari kakas *Blender*, bangun ruang itu disimpan dalam sebuah file OBJ.



Gambar 3.2: Tampilan *UI Blender* Blender versi 2.81

3.3 Menampilkan *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.

3.3.1 Membentuk Gambar *StreetView* untuk Tekstur Ruang Tiga Dimensi

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,



Gambar 3.3: Pemanggilan *StreetView API* yang berhasil, dengan URL [https://maps.googleapis.com/maps/api/streetview?size=600x300&location=-6.8746537,107.6046282&key=\(disamarikan\)](https://maps.googleapis.com/maps/api/streetview?size=600x300&location=-6.8746537,107.6046282&key=(disamarikan)), dengan parameter *heading* yang berbeda-beda-jelaskan di bab 2 tentang keluaran directions.

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 3.3 memperlihatkan empat gambar *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 3.4 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 pemandangan *StreetView* dapat ditampilkan pada dunia VR.

3.4 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.

3.4.1 Menentukan Atribut yang akan Digunakan

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



Gambar 3.4: Contoh hasil penggabungan empat gambar *Street View*

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" : {
      "text" : "35 mins",
      "value" : 2116
    },
    "end_address" : "Jl. Ir. H. Juanda No.100 , Lebakgede ...",
    "end_location" : {
      "lat" : -6.893148,
      "lng" : 107.6131791
    },
    "start_address" : "Jl. Ciumbuleuit No.94 , Hegarmanah ...",
    "start_location" : {
      "lat" : -6.8746719,
      "lng" : 107.6046127
    },
    "steps" : [
      {
        "distance" : {
          "text" : "1.0 km",
          "value" : 1008
        },
        "duration" : {
          "text" : "11 mins",
          "value" : 667
        }
      }
    ]
  }
]
```

```
        } ,
        "end_location" : {
            "lat" : -6.8833328,
            "lng" : 107.6049108
        },
        "html_instructions" : "Head\u202a\u003cb\u003esouth . . .",
        "polyline" : {
            "points" : "tu}h@yowoSdAP|AL‘Cb@dAHHBvC . . ."
        },
        "start_location" : {
            "lat" : -6.8746719,
            "lng" : 107.6046127
        },
        "travel_mode" : "WALKING"
    },
{
    "distance" : {
        "text" : "1.1\u00a1km",
        "value" : 1078
    },
    "duration" : {
        "text" : "14\u00a1mins",
        "value" : 834
    },
    "end_location" : {
        "lat" : -6.885218399999999,
        "lng" : 107.6137497
    },
    "html_instructions" : "Turn\u202a\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\u00a1km",
        "value" : 884
    },
    "duration" : {
        "text" : "10\u00a1mins",
        "value" : 603
    },
    "end_location" : {
        "lat" : -6.893140799999999,
        "lng" : 107.6130843
    }
}
```

```

        },
        "html_instructions" : "Turn\u2022\u003cb\u003eright ...",
        "maneuver" : "turn-right",
        "polyline" : {
            "points" : "rw_i@}hyoSzCLIAHz@Bd@@fB@tBDfABR@L...",
        },
        "start_location" : {
            "lat" : -6.88521839999999,
            "lng" : 107.6137497
        },
        "travel_mode" : "WALKING"
    },
    {
        "distance" : {
            "text" : "10\u00b3m",
            "value" : 10
        },
        "duration" : {
            "text" : "1\u00b3min",
            "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" : -6.89314079999999,
            "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.\u2022Ciumbuleuit,\u2022Jl.\u2022Siliwangi ...",
"warnings" : [
    "Walking\u00b2directions\u00b2are\u00b2in\u00b2beta.\u00b2Use\u00b2caution ..."
],
"waypoint_order" : []
}
],

```

```
    "status" : "OK"
}
```

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

3.4.2 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.

3.5 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 3.4.2, tetapi perubahan gambar *StreetView* yang sesuai *Directions API* harus berubah dengan tahap yang benar agar animasi pemandangan terlihat baik sesuai dengan rute.

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 5.1 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. Karena posisi gawai masih dalam keadaan *portrait*, ada satu *activity* terlebih dahulu yang muncul seperti di Gambar 5.2.

Activity ini dimunculkan ketika mengakses *Google Cardboard*. Untuk menuju ke *activity* VR, pengguna harus memutar gawai sehingga gawai berada dalam posisi *landscape*. Setelah gawai berada dalam posisi *landscape*, *activity* VR akan dimunculkan seperti yang ditunjukkan pada Gambar 5.3.

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 gawai 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 pemanfaatan sensor *step detector*.

Pemanfaatan *Google Directions API*

Directions API adalah *API* yang menggunakan protokol HTTP/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 lari 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 hal ini, *parsing* dari JSON ini harus dilakukan sesuai dengan ketentuan tersebut. Algoritma *parsing* dari JSON *Directions* dipaparkan pada 1.

Setelah JSON dari *Directions API* diunduh, *parsing* adalah langkah selanjutnya untuk mengambil informasi yang dibutuhkan. Karena bagian *steps* yang harus diambil, 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*

StreetView API berfungsi untuk memuat gambar pemandangan dan menggunakan menggunakan HTTP/HTTPS. Karena satu kali pemanggilan hanya menghasilkan gambar dari satu *heading* atau arah, beberapa gambar dari beberapa arah pandangan haruslah diunduh terlebih dahulu. Setelah semua gambar itu terunduh, semua gambar itu disatukan, menjadi satu gambar untuk menjadi tekstur bangun ruang silinder. Setelah gambar-gambar itu sudah disatukan, gambar tersebut harus disimpan di dalam *file*.

Karena pemandangan yang harus diunduh adalah gambar dari lokasi asal dan tujuan dari pengguna untuk membentuk perjalanan lari dari pengguna, gambar-gambar dari lokasi asal dan setiap *steps* sampai lokasi tujuan harus diperoleh. 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 iterasi 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.

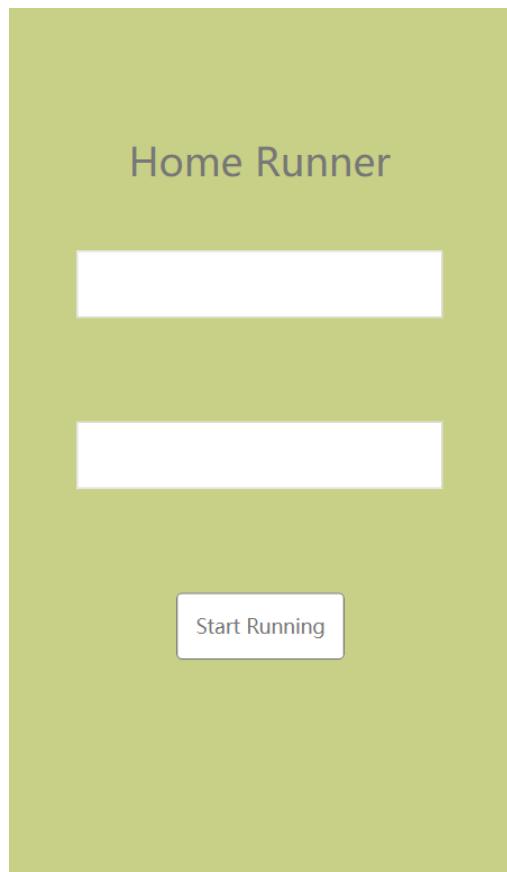
Algorithm 2 Algoritma Saat *Step Detector* Menerima Rangsang

```

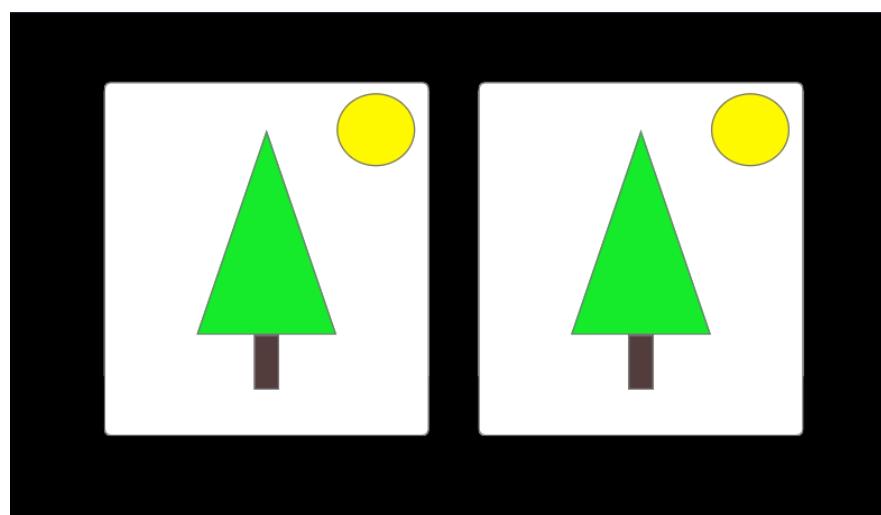
1: function ONSTEPDETECTORCHANGED(stepsDist)
2:   distanceElapsed  $\leftarrow$  0
3:   currentStepDist  $\leftarrow$  first element of stepsDist.value
4:   if sensor detects footstep then
5:     distanceElapsed  $\leftarrow$  distanceElapsed + DISTANCE_PER_STEP
6:     if distanceElapsed  $<$  currentStepDist.value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

```

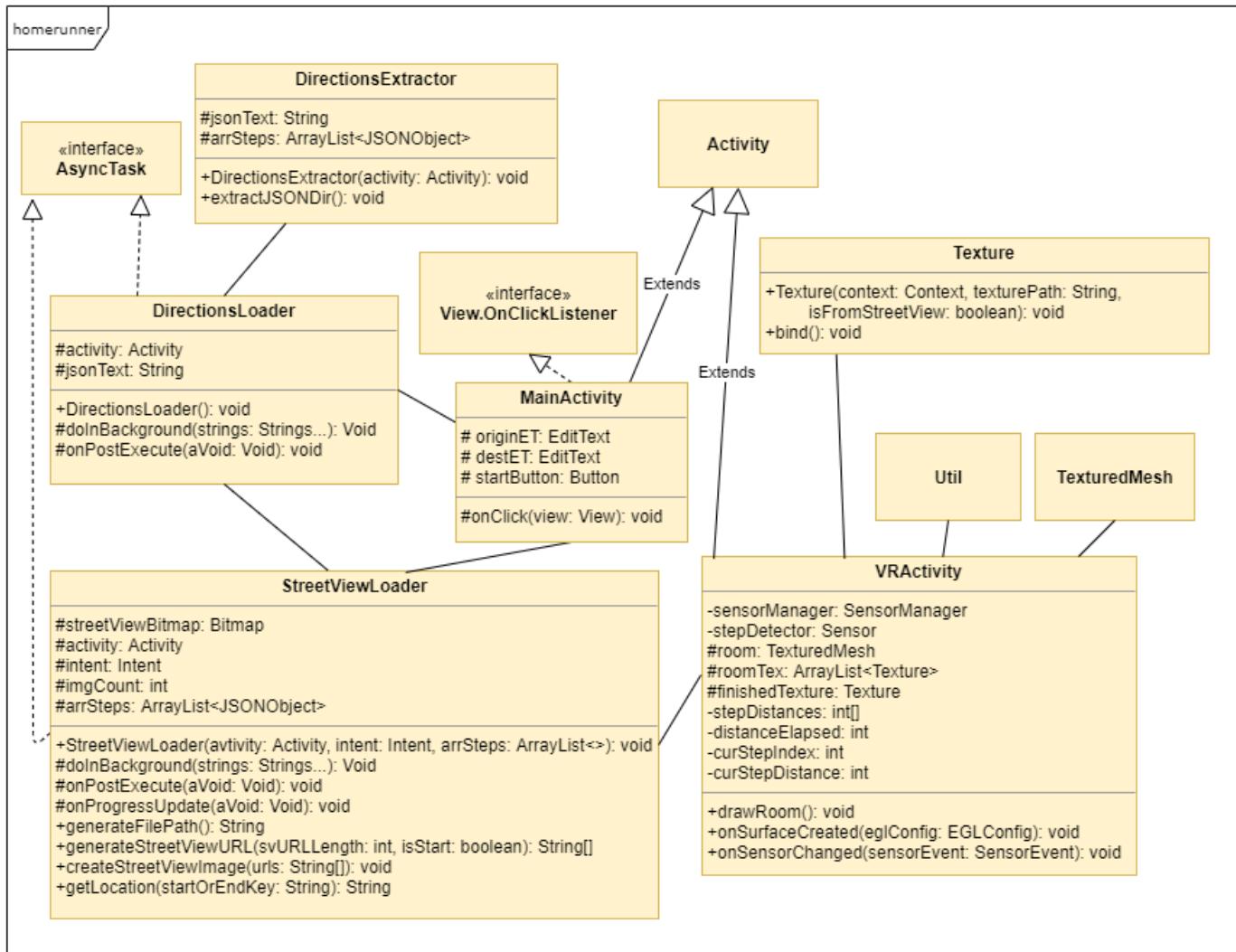
StepsDist merupakan array dari *steps distance*. *DistanceElapsed* merupakan peubah yang mencatat *progress* langkah dari pengguna, sementara *currentStepDist* merupakan peubah yang mencatat jarak yang harus ditempuh untuk mencapai *steps distance* berikutnya. Peubah *distanceElapsed* akan ditambah dengan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

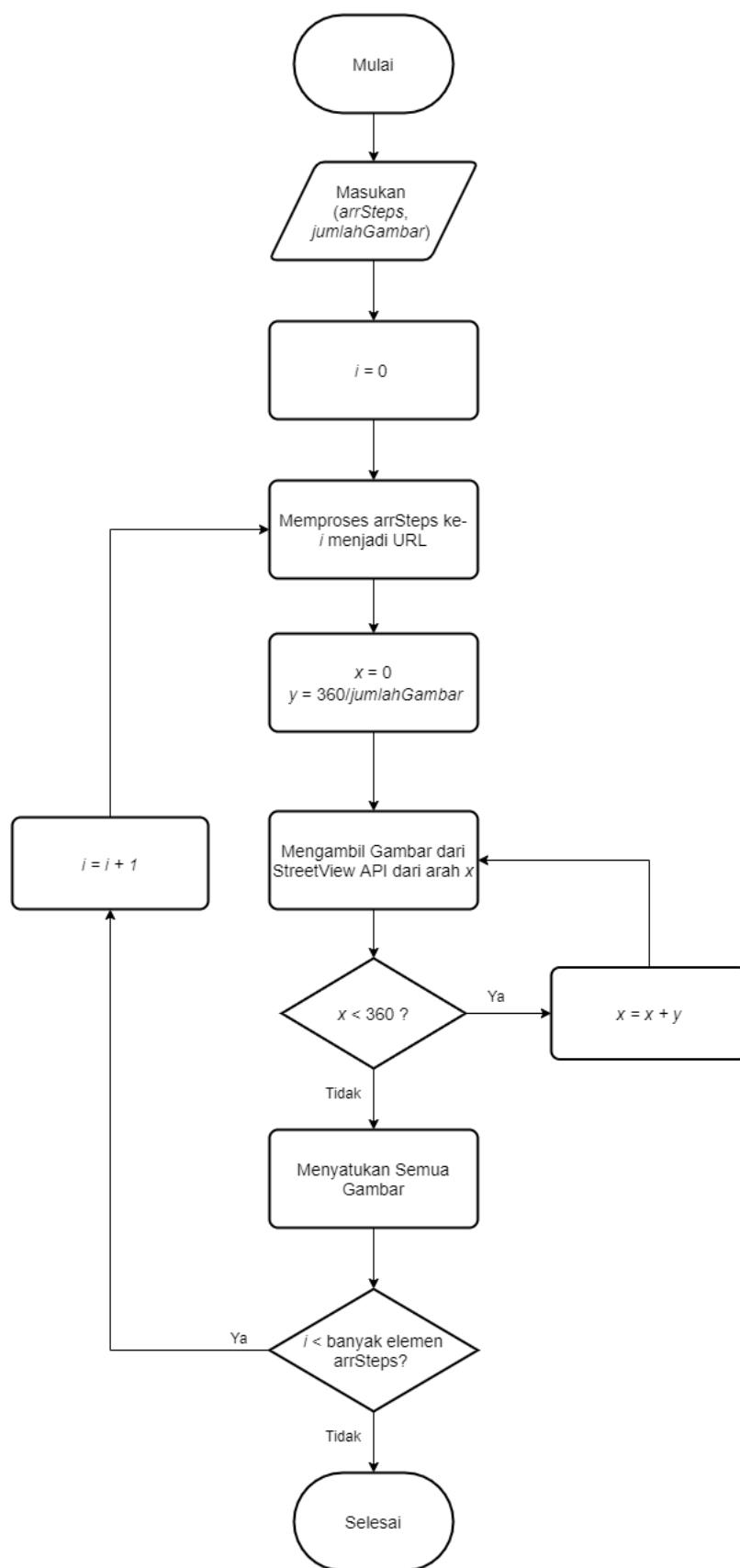


Gambar 4.1: Rancangan *Activity* utama aplikasi



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.

5.1 Implementasi

Aplikasi dikembangkan dengan Android Studio, dengan sebuah *laptop*. Aplikasi dijalankan pada x 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

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

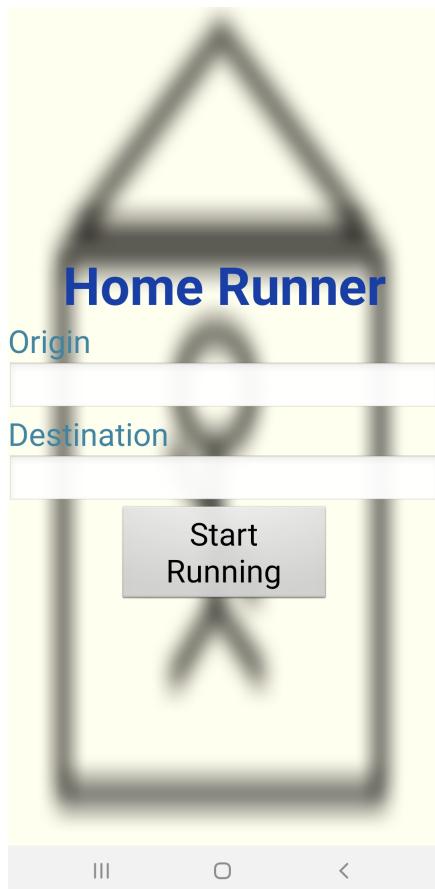
Yang dihasilkan dari implementasi ini adalah aplikasi VR untuk berlari. Aplikasi dapat diperoleh di Google Play Store dan dapat di-*install* pada gawai dengan sistem operasi Android.

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 5.1 menunjukkan tampilan *activity* ini.

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 5.2 menunjukkan tampilan peringatan ini.



Gambar 5.1: *Activity* utama aplikasi



Gambar 5.2: Tampilan *Google Cardboard* sebelum gawai diputar



Gambar 5.3: Tampilan VR saat Pengguna Berlari



Gambar 5.4: Tampilan VR saat Pengguna Mencapai Tujuan

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 5.3.

5.1.3 Aplikasi Setelah Pengguna Menyelesaikan Perjalanannya

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

5.2 Pengujian

Pengujian yang dilakukan adalah pengujian fungsional, yaitu menguji perangkat lunak pada tiga perangkat.

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 v9.0 (Pie)	4GB
3	1.6GHz octa-core	Android OS v8.1.0 (Oreo)	2GB

Tabel 5.1: Tabel Perangkat yang digunakan

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*.
2. Galat saat meng-*update* Gradle yang membuat kemajuan pengembangan aplikasi terhambat.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

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

6.2 Saran

Beberapa saran yang dapat diberikan untuk pengembangan:

1. Memanfaatkan fitur StreetView VR yang sudah diimplementasikan Google.

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] v1.190.0 (2019) *Releases - googlevr/gvr-android-sdk*. Github. Washington, United States.
- [5] 2018-10-29 (2020) *com.google.vr.sdk.base / Google VR / Google Developers*. Google. Mountain View, United States.
- [6] 2020-12-11 (2020) *Get an API Key and Signature / Street View Static API*. Google. Mountain View, United States.
- [7] 2020-10-28 (2020) *Motion sensors / Android Developers*. Google. Mountain View, United States.