

## 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

<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 . . . . .	1
1.4 Batasan Masalah . . . . .	2
1.5 Metodologi Penelitian . . . . .	2
1.6 Sistematika Pembahasan . . . . .	2
<b>2 LANDASAN TEORI</b>	<b>3</b>
2.1 Google VR . . . . .	3
2.1.1 Google VR SDK . . . . .	3
2.1.2 Aplikasi HelloVR . . . . .	6
2.2 Google <i>StreetView API</i> . . . . .	9
2.2.1 <i>API Key</i> . . . . .	10
2.2.2 Penggunaan <i>StreetView API</i> . . . . .	10
2.2.3 Atribut Parameter <i>StreetView API</i> . . . . .	10
2.3 Google <i>Directions API</i> . . . . .	11
2.3.1 Penggunaan <i>Directions API</i> . . . . .	11
2.3.2 Hasil Pemanggilan <i>Directions API</i> . . . . .	13
2.4 <i>Motion Sensor</i> . . . . .	18
2.4.1 Deskripsi <i>Motion Sensor</i> . . . . .	18
2.4.2 Deskripsi <i>Step Detector</i> . . . . .	18
<b>3 ANALISIS</b>	<b>19</b>
3.1 Membuat Dunia VR . . . . .	19
3.2 Menampilkan <i>StreetView API</i> pada Dunia VR . . . . .	20
3.3 Integrasi dengan <i>Directions API</i> . . . . .	20
3.3.1 Menentukan Atribut yang Bermanfaat . . . . .	21
3.3.2 Cara Memanfaatkan Atribut . . . . .	21
3.4 Cara Memanfaatkan <i>Step Detector</i> Sensor . . . . .	21
<b>DAFTAR REFERENSI</b>	<b>23</b>



## DAFTAR GAMBAR

2.1	Struktur Direktori Google VR SDK . . . . .	4
2.2	Tampilan <i>UI</i> permainan <i>treasure hunt</i> pada aplikasi HelloVR . . . . .	6
2.3	Struktur Direktori Aplikasi HelloVR . . . . .	7
2.4	Gambar-gambar <i>assets</i> aplikasi HelloVR . . . . .	7
2.5	Tampilan <i>UI Google Cloud</i> saat mengakses <i>API Key (API Key disamaraskan)</i> . . . . .	9
2.6	Pemanggilan <i>StreetView API</i> yang berhasil . . . . .	12
3.1	Tampilan <i>UI Blender</i> Blender versi 2.81 . . . . .	19
3.2	Pemanggilan <i>StreetView API</i> yang berhasil, dengan <i>URL https://maps.googleapis.com/maps/api/streetview?size=600x300&amp;location=Rumah+Sakit+Santo+Borromeus&amp;key=AIzaSyALPfhhnemi3xC4-FUthkWidaugsZTwJq4</i> , dengan parameter <i>heading</i> yang berbeda-beda-jelaskan di bab 2 tentang keluaran directions. . . . .	20
3.3	Contoh hasil penggabungan empat gambar <i>StreetView</i> . . . . .	21



## **DAFTAR TABEL**



# 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 adalah salah satu kebutuhan manusia untuk menjaga kesehatan. Salah satu aktivitas olahraga yang paling mudah dan tidak memerlukan gerakan yang sulit adalah berlari, namun kegiatan ini memerlukan lahan yang cukup besar agar dapat dilakukan dengan leluasa. Selain kebutuhan lahan, aktivitas berlari lebih menyenangkan jika dilakukan di luar rumah. Agar dapat dilakukan di dalam rumah, berlari dapat dilakukan di rumah adalah *treadmill*, akan tetapi masalah lingkungan yang monoton dan membosankan di dalam rumah membuat orang enggan untuk melakukan aktivitas berlari. Bila suasana dunia luar dapat dibawa ke dalam rumah, aktivitas ini dapat dilakukan di dalam rumah, tetapi suasana yang dirasakan adalah seperti di luar rumah.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat menampilkan simulasi aktivitas 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.

Teknologi yang dapat dimanfaatkan untuk membuat aplikasi VR untuk berlari adalah *Google Cardboard* dan sensor perangkat bergerak, dan untuk *Application Programming Interface (API)* yang digunakan adalah *Google Streetview API* dan *Google Directions API* [1] [2] [3].

### 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 Cardboards*.
- Mengintegrasikan *Google Directions API* dengan pemandangan *StreetView* yang telah ditampilkan pada *Google Cardboard*.
- Menganalisis sensor langkah dan menyinkronisasikannya dengan perubahan pemandangan *StreetView*.

Untuk membuat skripsi ini, peneliti . 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*.

## 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

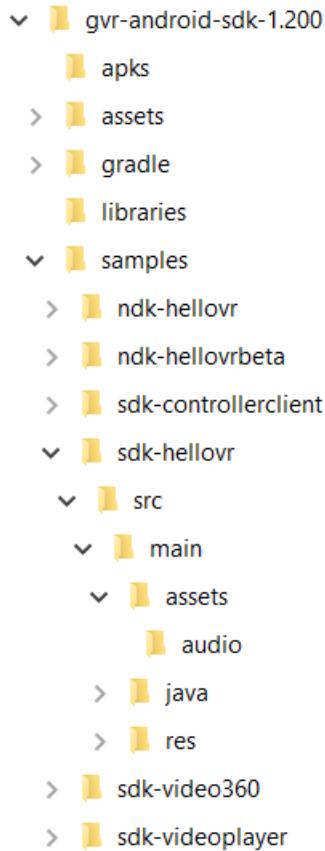
Google VR SDK adalah alat bantu berlisensi Apache 2.0 yang disediakan Google pada repository Github yang berisi kode program dari aplikasi *virtual reality* (VR) yang tersedia untuk Android (Java), Android NDK, Unity, dan iOS [4]. Secara umum, SDK ini dibuat agar pengembang perangkat lunak dapat mempelajari 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. 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]. Berikut adalah beberapa *class* dan *interface* dari *package com.google.vr.sdk.base*:

###### 1. *GvrView.StereoRenderer*

*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:



Gambar 2.1: Struktur Direktori Google VR SDK

– **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* dari Google. *Method-method* yang dimiliki *class* ini adalah:

- `public void setEGLConfigChooser (int redSize, int greenSize, int blueSize, int alphaSize, int depthSize, int stencilSize)`

*View* yang mendukung *rendering* *VR*.

## 6. HeadTransform

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

- public void getEulerAngles (float[] eulerAngles, int offset)
- public void getForwardVector (float[] forward, int offset)
- public void getHeadView (float[] headView, int offset)
- public void getQuaternion (float[] quaternion, int offset)
- public void getRightVector (float[] right, int offset)
- public void getTranslation (float[] translation, int offset)
- public void getUpVector (float[] up, int offset)

7. *Viewport Class* yang mendefinisikan *viewport* berbentuk persegi panjang.

Atribut-atribut yang dimiliki kelas ini adalah sebagai berikut:

- public int height
- public int width
- public int x
- public int y

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

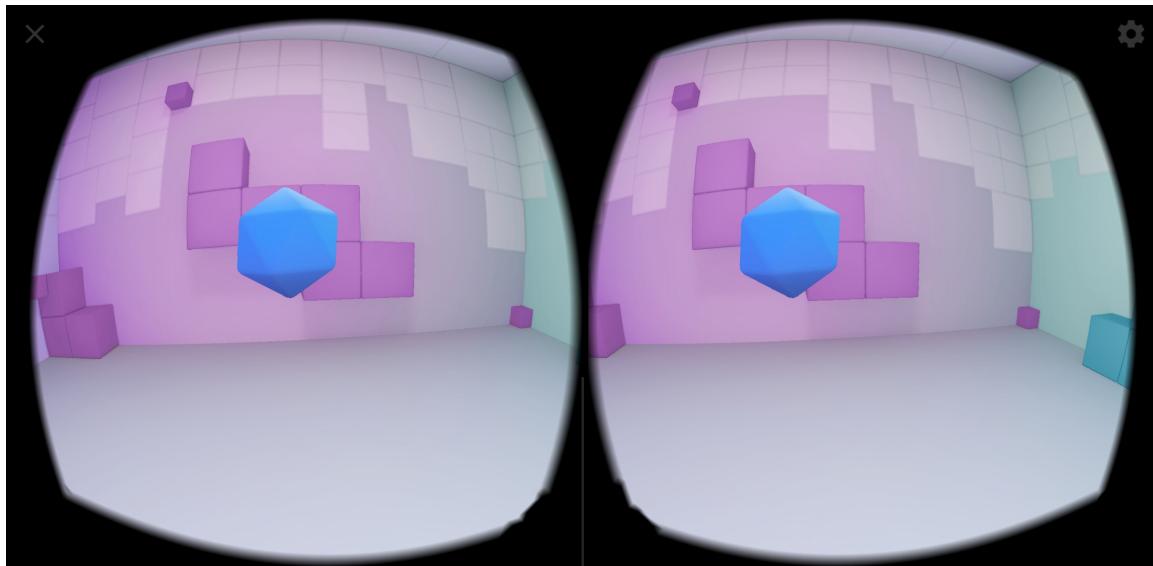
- public void getAsArray (int[] array, int offset)
- public void setGLScissor ()
- public void setGLViewport ()
- public void setViewport (int x, int y, int width, int height)

### 2.1.2 Aplikasi HelloVR

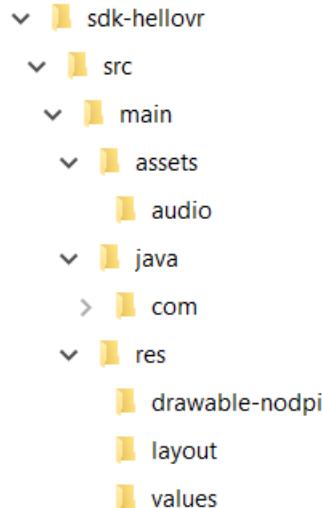
HelloVR, aplikasi yang diperoleh dari Google VR SDK pada direktori `./samples/sdk-hellovr` adalah sebuah aplikasi demo permainan *treasure hunt*, yaitu sejenis permainan mencari bentuk yang mengapung di dunia VR dengan melihat tepat pada bentuk tersebut dan menyalakan pemicu pada Google Cardboard. Setelah 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.

#### Komponen Aplikasi HelloVR

Aplikasi HelloVR terdiri dari beberapa *folder* dan komponen-komponen. Gambar 2.3 menunjukkan struktur direktori aplikasi HelloVR. Dunia VR pada aplikasi ini dibuat dari *file Wavefront Object* (OBJ) dengan 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.



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



Gambar 2.3: 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:

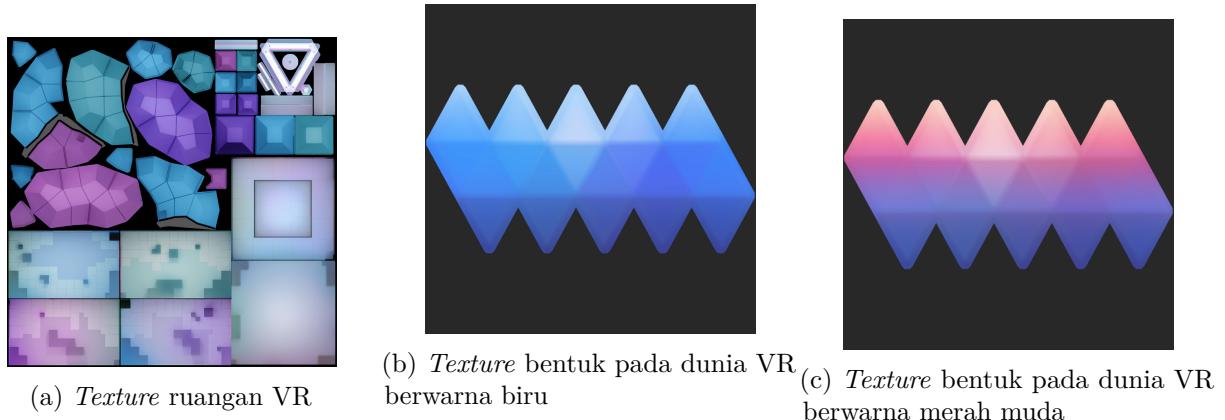
- `int[] textureId` - Atribut yang menyimpan representasi tekstur ruangan yang dapat digunakan dalam kode program.

*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



Gambar 2.4: Gambar-gambar *assets* aplikasi HelloVR

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 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 horisontal × 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. *locaction* 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)).



Gambar 2.6: Pemanggilan *StreetView API* yang berhasil

## 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>

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=...> (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 data ©2020",
      "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 \u2192 \u003cb\u003e south . . .",
                "polyline" : {
                    "points" : "tu}h@yowoSdAP|AL'Cb@dAHHBvC . . ."
                },
                "start_location" : {
                    "lat" : -6.8746719,
                    "lng" : 107.6046127
                },
                "travel_mode" : "WALKING"
            },
            {
                "distance" : {
                    "text" : "1.1 km",
                    "value" : 1078
                },
                "duration" : {
                    "text" : "14 mins",
                    "value" : 834
                },
                "end_location" : {
                    "lat" : -6.88521839999999,
                    "lng" : 107.6137497
                },
                "html_instructions" : "Turn \u2192 \u003cb\u003e left \u003c/b\u003e . . .",
                "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\u202a\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\u00a0m",
            "value" : 10
        },
        "duration" : {
            "text" : "1\u00a0min",
            "value" : 12
        },
        "end_location" : {
            "lat" : -6.893148,
            "lng" : 107.6131791
        },
        "html_instructions" : "Turn\u202a\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 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.



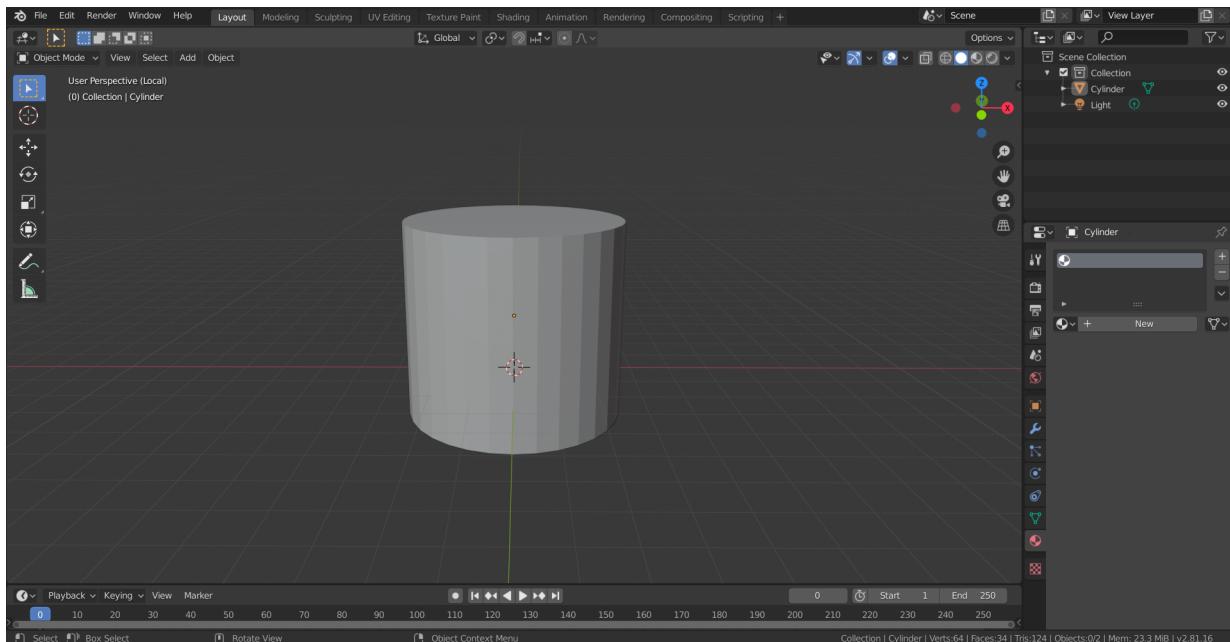
## BAB 3

# ANALISIS

Pada bab ini dijelaskan mengenai cara memanfaatkan Google VR SDK, *Google StreetView API*, *Google Directions API*, dan cara memanfaatkan sensor *step detector*, termasuk cara mengintegrasikan semua komponen tersebut secara bertahap.

### 3.1 Membuat Dunia VR

Dari komponen-komponen Google VR SDK, bagian *assets*-lah yang dapat dimodifikasi untuk menampilkan pemandangan VR yang ingin ditampilkan. Dunia VR terdiri atas dua *file*, yaitu *file* OBJ sebagai bentuk ruangan dan *file* PNG sebagai tekstur ruangan *file* OBJ. Dunia yang ingin ditampilkan pada dunia VR adalah dunia dengan bentuk silinder agar terlihat seperti dunia nyata, terutama bagian permukaan samping (yang melengkung). Untuk membuat dunia berbentuk silinder, kertas yang digunakan adalah *Blender* versi 2.81. Gambar menunjukkan tampilan *UI Blender* versi 2.81.



Gambar 3.1: Tampilan *UI Blender* Blender versi 2.81

Bentuk silinder dipilih sebagai bentuk dari dunia VR karena permukaan samping dapat menampilkan pemandangan seperti di dunia nyata. Permukaan samping berbentuk persegi panjang, dan sifat melingkar dari permukaan dapat membuat menampilkan pemandangan di sekitar.



Gambar 3.2: Pemanggilan *StreetView API* yang berhasil, dengan URL <https://maps.googleapis.com/maps/api/streetview?size=600x300&location=Rumah+Sakit+Santo+Borromeus&key=AIzaSyALPfhhnemi3xC4-FUtHkWidaugsZTwJq4>, dengan parameter *heading* yang berbeda-beda jelaskan di bab 2 tentang keluaran *directions*.

### 3.2 Menampilkan *StreetView API* pada Dunia VR

Dengan menggunakan HTTP/HTTPS URL, gambar yang dihasilkan merupakan gambar persegi panjang saat menghadap satu arah sehingga tidak dapat secara langsung ditampilkan pada dunia VR [2]. Untuk melakukannya, gambar dari empat arah 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 *StreetView* dengan berbagai macam *StreetView* dengan syarat tersebut.

Setelah mendapatkan gambar *StreetView* dari semua arah, gambar-gagmbar 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 pemandangan *StreetView* dapat ditampilkan pada dunia VR.

### 3.3 Integrasi dengan *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.



Gambar 3.3: Contoh hasil penggabungan empat gambar *StreetView*

### 3.3.1 Menentukan Atribut yang Bermanfaat

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.

### 3.3.2 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.

## 3.4 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 3.3.2, tetapi perubahan gambar *StreetView* yang sesuai *Directions API* harus berubah dengan tahap yang benar agar animasi pemandangan terlihat mulus.



## **DAFTAR REFERENSI**

- [1] (2018) *Quickstart for Google VR SDK for Android* / Google Developers.
- [2] (2020) *Developer Guide / Street View Static API* / Google Developers.
- [3] (2020) *Developer Guide / Directions API* / Google Developers.
- [4] (2018) *Choose Your Development Environment* / Google VR / Google Developers.
- [5] (2020) *com.google.vr.sdk.base* / Google VR / Google Developers.
- [6] (2020) *Get an API Key and Signature* / Street View Static API.
- [7] (2020) *Motion sensors* / Android Developers.