

SKRIPSI

DETEKSI GERAKAN KEPALA DENGAN GOOGLE CARDBOARD



EGA PRIANTO

NPM: 2013730047

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2016

UNDERGRADUATE THESIS

HEAD MOTION DETECTOR USING GOOGLE CARDBOARD



EGA PRIANTO

NPM: 2013730047

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

2016

LEMBAR PENGESAHAN

DETEKSI GERAKAN KEPALA DENGAN GOOGLE CARDBOARD

EGA PRIANTO

NPM: 2013730047

Bandung, 20 Oktober 2016

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

**Pascal Alfadian, M.Comp.
Ketua Tim Penguji**

**«pembimbing pendamping/2»
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:

DETEKSI GERAKAN KEPALA DENGAN 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 dijatuuhkan 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 20 Oktober 2016

Meterai

Ega Prianto
NPM: 2013730047

ABSTRAK

Virtual Reality adalah salah satu interaksi antara manusia dengan komputer yang membuat penggunanya merasakan berada pada suatu lingkungan buatan yang berada pada komputer. Banyak cara untuk dapat merasakan pengalaman *Virtual Reality* salah satunya adalah dengan menggunakan Google Cardboard. Google Cardboard merupakan gawai murah yang terbuat dari kardus untuk mengalami pengalaman *Virtual Reality* pada perangkat *smartphone* Android atau IOS. Google Cardboard dapat memberikan pengalaman *Virtual Reality* karena menggunakan sensor-sensor yang terdapat pada perangkat *smartphone*.

Masukkan pada Google Cardboard sangatlah terbatas. Masukkan tersebut adalah dengan menarik atau menekan tombol yang ada pada kacamata Google Cardboard dan orientasi dari perangkat *smartphone* saja. Tombol pada Google Cardboard ini pun terkadang tidak berfungsi dengan baik. Oleh karena itu dibuatlah aplikasi untuk mendeteksi gerakan kepala, khususnya mengangguk dengan menggeleng.

Pengujian dilakukan dengan mencoba menjalankan aplikasi yang telah dibuat pada beberapa perangkat dengan spesifikasi yang berbeda-beda. Selain itu pengujian juga dilakukan dengan mengajak beberapa orang untuk menggunakan aplikasi ini. Berdasarkan hasil pengujian, aplikasi ini dapat berjalan dengan baik pada perangkat *smartphone* dengan spesifikasi tertentu. Aplikasi ini juga sudah dapat membaca gerakan kepala yang sesuai yang dilakukan oleh pengguna, tetapi terkadang simpangan yang ditetapkan pada aplikasi ini tidak sesuai dengan kriteria masing-masing pengguna. Jadi pengguna harus menyesuaikan gerakan kepalamya agar sesuai dengan batasan simpangan yang telah ditetapkan pada aplikasi ini.

Kata-kata kunci: Virtual Reality, Google Cardboard, Sensor Gyroscope, Google VR, Android, Unity

ABSTRACT

Virtual Reality is one of the interactions between humans and computers that make its users feel in the artificial environment on a computer. There is a lot of ways to experience the Virtual Reality, one of them is using Google Cardboard. Google Cardboard is a low-cost system to encourage interest and development in Virtual Reality applications on a Android or IOS smartphone. Google Cardboard using sensors on smartphone devices to perform Virtual Reality.

Input method on Google Cardboard is very limited. The only input on the Google Cardboard is to pull or press the button on the Google Cardboard. Sometimes the button doesn't work very well. Therefore this application was made to detect the head, especially nodding by shaking his head.

Testing is done by trying to run this application that have been made on several devices with different specifications. In addition, testing is also done by inviting some people to use this application. Based on the results of testing, this application runs well on smartphone devices with certain specifications. This application is also able to read the appropriate head movement performed by the user, But sometimes the deviation set in this app does not match the criteria of each user. So the user must adjust his head movement to fit the deviation limit set in this application.

Keywords: Virtual Reality, Google Cardboard, Sensor Gyroscope, Google VR, Android, Unity

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

KATA PENGANTAR

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Bandung, Oktober 2016

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	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metode Penelitian	2
1.6 Sistematika Penulisan	3
2 DASAR TEORI	5
2.1 Android SDK	5
2.1.1 Struktur File Android Studio Project	5
2.1.2 Membuat User Interface	5
2.1.3 Activity	8
2.1.4 Android Sensor Framework	10
2.2 Google VR SDK	19
2.2.1 API Audio	19
2.2.2 API Base	20
2.3 Teori Kuaternion	22
2.3.1 Struktur Ajabar	22
2.3.2 <i>Ajabar Kuaternion</i> dan Operasi-operasi pada Kuaternion	24
2.4 Unity Game Engine	27
2.4.1 Struktur Hierarki GameObject	27
2.4.2 Prefabs	27
2.4.3 Scene	28
2.4.4 GameObject	28
2.4.5 Transformasi Rotasi dan Orientasi	30
2.5 Google VR SDK for Unity	31
3 ANALISIS	33
3.1 Analisis Aplikasi Sejenis	33
3.2 Perekaman Data Sensor	36
3.2.1 Perekaman Grafik Sensor <i>Accelerometer</i>	37
3.2.2 Perekaman Grafik Sensor <i>Gyroscope</i>	38
3.2.3 Perekaman Grafik Sensor <i>Rotation Vector</i>	40
3.3 Analisis Data Sensor untuk Mendeteksi Gerakan Kepala	42

3.4	Analisis Metode Pendekripsi Gerakan Kepala	43
3.4.1	Algoritma Mendekripsi Gerakan Mengangguk	44
3.4.2	Algoritma Mendekripsi Gerakan Menggeleng	45
4	PERANCANGAN	49
4.1	Perancangan Kelas Algoritma Pendekripsi Gerakan Kepala	49
4.2	Perancangan Hierarki GameObject pada Unity	53
4.3	Perancangan <i>Gameplay</i> dan Perancangan Antarmuka	55
5	IMPLEMENTASI DAN PENGUJIAN	57
5.1	Implementasi	57
5.1.1	Lingkungan Implementasi dan Pengujian	57
5.1.2	Hasil Implementasi	57
5.2	Pengujian	58
5.2.1	Pengujian Fungsional	62
5.2.2	Pengujian Eksperimental	63
5.3	Masalah yang Dihadapi pada Saat Implementasi	63
6	KESIMPULAN	67
6.1	Kesimpulan	67
6.2	Saran	67
DAFTAR REFERENSI		69
A	KODE PROGRAM APLIKASI UNTUK ANALISIS	71
B	KODE PROGRAM APLIKASI <i>Game</i>	77

DAFTAR GAMBAR

2.1	Tampilan struktur folder pada <i>project</i> Android Studio	6
2.2	Illustrasi bagaimana percabangan objek ViewGroup pada <i>layout</i> dan mengandung objek View lainnya.	7
2.3	<i>State diagram</i> siklus Activity	9
2.4	Sistem koordinat (relatif dengan perangkatnya) yang digunakan oleh Sensor API	12
2.5	Sistem koordinat sensor rotasi vektor terhadap Bumi	18
2.6	Contoh perputaran dengan bilangan kompleks	24
2.7	Contoh perputaran tiga puluh derajat dengan bilangan kompleks	25
2.8	Right-hand rule dalam <i>cross product</i> vektor	26
2.9	Contoh Hierarchy Parenting.	28
2.10	Contoh <i>scene</i> kosong.	29
2.11	Contoh penggunaan komponen pada GameObject.	29
2.12	Contoh komponen pada GameObject kubus.	30
3.1	<i>Screenshot</i> task yang diberikan aplikasi InMind VR.	34
3.2	<i>Screenshot</i> aplikasi InMind VR ketika permainan berlangsung.	34
3.3	<i>Screenshot</i> aplikasi InMind VR ketika meminta pengguna untuk mengangguk jika telah siap.	35
3.4	Gambar untuk mendeskripsikan posisi dari muka sebelum melakukan gerakan menggeleng atau mengangguk. Gambar (a) adalah kondisi muka ketika sedang menghadap ke depan. Gambar (b) adalah kondisi muka ketika sedang menghadap ke atas. Gambar(c) adalah kondisi muka ketika sedang menghadap ke serong kiri atas.	36
3.5	Grafik nilai kuaternion dari sensor <i>accelerometer</i> ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.	37
3.6	Grafik nilai kuaternion dari sensor <i>accelerometer</i> ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.	38
3.7	Gambar grafik nilai sensor <i>gyroscope</i> ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.	39
3.8	Gambar grafik nilai sensor <i>gyroscope</i> ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.	39
3.9	Grafik nilai kuaternion dari sensor <i>rotation vector</i> ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.	40
3.10	Grafik nilai kuaternion dari sensor <i>rotation vector</i> ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.	41
3.11	Dua buah rotasi yang identik dengan nilai kuaternion yang berbeda.	42

3.12 Deskripsi grafik pada saat pengguna menggeleng. Yang ditunjuk oleh panah berwarna hitam adalah selang waktu yang terjadi saat pengguna melawan arah gerakan kepala.	43
4.1 Diagram Kelas Algoritma Pendekripsi Gerakan Kepala.	50
4.2 Perancangan hierarki Game Object	54
4.3 Meteran pada permainan.	55
4.4 Desain ruangan untuk permainan yang akan dibuat.	56
4.5 Desain User Interface	56
5.1 Tampilan ketika aplikasi baru dimulai.	59
5.2 Tampilan ketika perangkat tidak memiliki sensor Gyroscope.	59
5.3 Tampilan ketika notifikasi muncul.	60
5.4 Tampilan ketika <i>pop-up</i> pertanyaan muncul.	60
5.5 Tampilan setelah pengguna mengangguk.	61
5.6 Tampilan setelah pengguna menggeleng.	61
5.7 Tampilan setelah permainan usai.	62
5.8 Pengujian oleh Ricky Setiawan.	63
5.9 Pengujian oleh Harseto Pandityo.	64
5.10 Pengujian oleh Herfan Heryandi.	64

DAFTAR TABEL

2.1 Tipe-tipe Sensor pada Android	11
5.1 Tabel Perangkat yang digunakan	58

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Virtual Reality(VR) adalah teknologi yang mampu membuat penggunanya dapat berinteraksi dengan lingkungan buatan oleh komputer, suatu lingkungan yang sebenarnya ditiru atau hanya ada di dalam imajinasi.^[1] VR membuat pengalaman sensorik, di antaranya penglihatan, pendengaran, perabaan, dan penciuman secara buatan.^[2] Gawai VR terbaru menggunakan metode *head-mounted display*, Google Cardboard salah satunya. *Head-mounted display* adalah menempatkan layar di kepala, sehingga pengguna hanya dapat melihat tampilan yang ditampilkan oleh layar.^[3]

Google Cardboard^[4] adalah gawai murah yang terbuat dari kardus untuk dapat merasakan pengalaman VR dengan *smartphone* Android atau iOS. Kita dapat membuat Google Cardboard kita sendiri secara gratis dengan mengunduh templatnya di situs web Google Cardboard. Template tersebut membantu dalam merakit kardus dengan dibentuk, dilipat dan dipotong sedemikian rupa sehingga berbentuk kacamata. Bahan-bahan untuk merakit Google Cardboard hanyalah kardus, lem, dan lensa dengan spesifikasi tertentu. Ada pula beberapa perusahaan yang membuat gawai ini dengan kualitas yang lebih baik dari pada kardus seperti plastik.

Pada gawai Google Cardboard cara pengguna memberikan *input* kepada program sangatlah terbatas. Cara tersebut hanyalah dengan gerakan kepala dan tombol pada Google Cardboard. Tombol ini pun terkadang tidak berfungsi dengan baik. Cara lainnya agar dapat memberikan masukkan kepada program adalah dengan menghubungkan *smartphone* yang digunakan dengan *bluetooth controller*.

Skripsi ini akan membuat aplikasi untuk menambahkan cara baru memberikan *input* pada Google Cardboard. Pada skripsi ini, akan dibuat dua buah perangkat lunak. Perangkat lunak pertama akan digunakan untuk menganalisis data yang didapat dari sensor-sensor pada Android. Perangkat lunak kedua akan dapat mendeteksi gerakan kepala penggunanya ketika sedang menggeleng atau mengangguk. Pada perangkat lunak kedua ini akan memberikan *input* baru kepada program VR. Jenis *input* yang diberikan kepada komputer hanya ya(mengangguk) atau tidak(menggeleng).

Agar VR yang menggunakan Google Cardboard dapat berjalan dengan sempurna, dibutuhkan *smartphone* yang memiliki 3 jenis sensor. Ketiga sensor itu adalah *Magnetometer*, *Accelerometer*, dan *Gyroscope*.^[5] Jika salah satu sensor itu tidak ada, tampilan gambar pada VR akan tidak akurat atau lambat. *Magnetometer* digunakan untuk mengetahui arah pandang pengguna. *Accelerometer* digunakan untuk mengetahui arah gaya gravitasi.^[6] *Gyroscope* digunakan untuk mengetahui percepatan perputaran sudut kepala pengguna. Ketiga sensor ini juga harus menggunakan sensor 3

sumbu. Ketiga sensor tersebut tidak hanya berfungsi agar dapat menjalankan VR dengan Google Cardboard dan *smartphone*, tetapi juga dapat berfungsi sebagai pendekripsi gerakan kepala.

1.2 Rumusan Masalah

- Bagaimana cara menampilkan grafik data yang diambil dari sensor-sensor pada *smartphone*?
- Bagaimana cara mendekripsi gerakan kepala dari data yang didapat dari sensor-sensor pada *smartphone*?

1.3 Tujuan

- Mengetahui cara untuk menampilkan grafik data dari sensor-sensor pada *smartphone*.
- Mengetahui cara mendekripsi gerakan kepala dari data yang didapat dari sensor-sensor pada *smartphone*.

1.4 Batasan Masalah

Penelitian ini dibuat berdasarkan batasan-batasan sebagai berikut:

1. Program pertama yang akan dibuat dalam skripsi ini hanya akan digunakan untuk membantu dalam menganalisis sensor.
2. Program kedua yang akan dibuat hanya dapat melakukan pendekripsi gerakan kepala khusus untuk mengangguk dan menggeleng saja.

1.5 Metode Penelitian

Berikut adalah metode penelitian yang digunakan dalam penelitian ini:

1. Melakukan studi literatur tentang Android SDK, Google VR SDK, Quaternion, *Sensor Fusion*, dan algoritma *Head Motion Detection*.
2. Merancang dan membuat aplikasi untuk menampilkan grafik sensor-sensor pada *smartphone* Android.
3. Menganalisis aplikasi-aplikasi sejenis.
4. Merekam dan menganalisis grafik dari sensor-sensor pada *smartphone* ketika mengangguk dan menggeleng.
5. Menganalisis metode pendekripsi gerakan kepala.
6. Merancang aplikasi untuk mendekripsi gerakan kepala
7. Mengimplementasi algoritma pendekripsi gerakan kepala ke aplikasi *virtual reality*.

1.6 Sistematika Penulisan

Setiap bab dalam penelitian ini memiliki sistematika penulisan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan, yaitu membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.
2. Bab 2: Dasar Teori, yaitu membahas teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang Android SDK, Google VR SDK, Quaternion.
3. Bab 3: Analisis, yaitu membahas mengenai analisa masalah. Berisi tentang analisis aplikasi-aplikasi sejenis, perekaman data sensor, analisis grafik dari sensor-sensor pada *smartphone* ketika mengangguk dan menggeleng, analisis metode pendekripsi gerakan kepala.
4. Bab 4: Perancangan, yaitu membahas mengenai perancangan aplikasi VR yang dapat mendekripsi gerakan menggeleng dan mengangguk.
5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dan pengujian aplikasi yang telah dilakukan. Berisi tentang implementasi dan hasil pengujian aplikasi.

BAB 2

DASAR TEORI

Pada bab ini akan dijelaskan dasar-dasar teori mengenai Android SDK, Google VR SDK, Quaternion, *Sensor Fusion*, dan algoritma *Head Motion Detection*.

2.1 Android SDK

Android SDK(*software development kit*) adalah kumpulan *source code*, *development tools*, *emulator*,^[7] dan semua *libraries* untuk membuat suatu aplikasi untuk *platform* Android. IDE (*integrated development environment*) yang resmi untuk Android SDK adalah Android Studio. Android Studio dapat di download di halaman situs web Google Developer^[7], sekaligus dengan Android SDKnya.

2.1.1 Struktur File Android Studio Project

[8] Dalam membuat aplikasi pada perangkat Android, dibutuhkan Android SDK. Android SDK(*software development kit*) adalah kumpulan *source code*, *development tools*, *emulator*,^[7] dan semua *libraries* untuk membuat suatu aplikasi untuk *platform* Android. IDE (*integrated development environment*) yang resmi untuk Android SDK adalah Android Studio. Android Studio dapat di download di halaman situs web Google Developer^[7], sekaligus dengan Android SDKnya. Struktur File Android Studio Project

Pada saat **project** baru telah dibuat, Android Studio akan membuatkan folder-folder standar seperti pada Gambar 2.1.

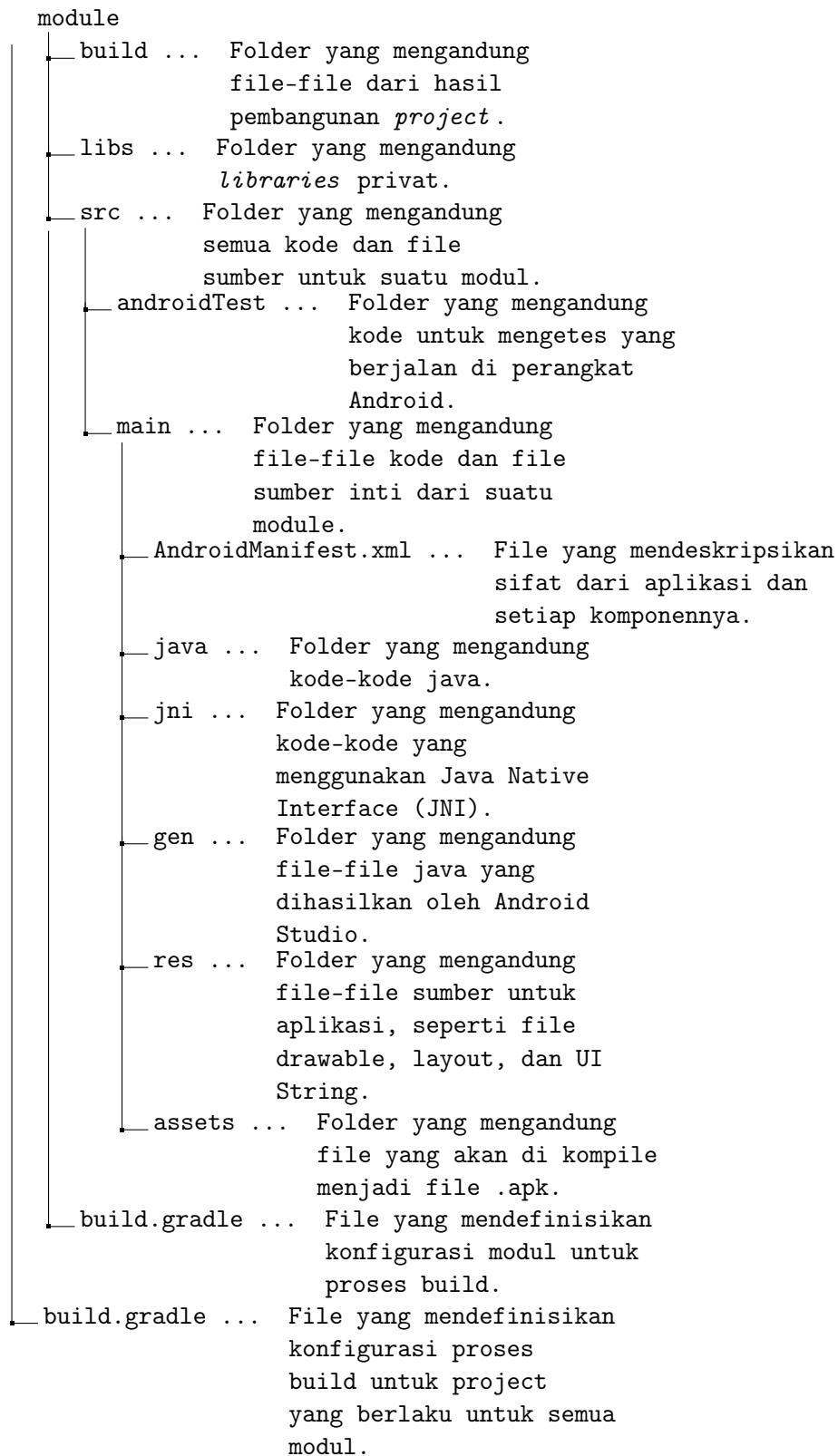
2.1.2 Membuat User Interface

[8] Pada subbab ini akan dijelaskan bagaimana membuat layout di XML termasuk *text field* dan *button*

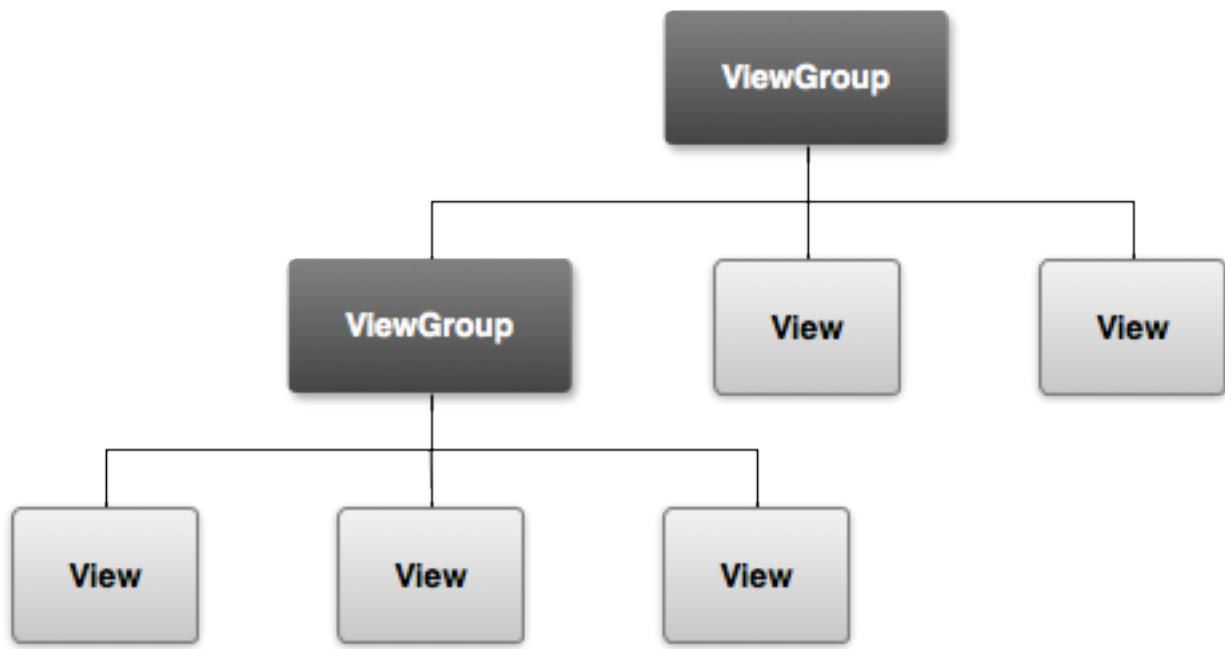
Hierarki *Graphical User Interface (GUI)* untuk Aplikasi Android

GUI untuk aplikasi Android dibuat dengan hierarki dari objek View dan ViewGroup (Gambar 2.2). Objek-objek dari View biasanya adalah *UI(User Interface) Widgets* seperti *button* atau *text field*. Objek-objek dari ViewGroup tidak terlihat oleh *view containers* yang mendefinisikan bagaimana *child views* ditata seperti *grid* atau *vertical list*.

Android menggunakan file XML yang berkorespondensi kepada *subclasses* dari View dan ViewGroup, sehingga UI dapat didefinisikan dalam XML menggunakan hierarki dari elemen UI.



Gambar 2.1: Tampilan struktur folder pada *project* Android Studio



Gambar 2.2: Illustrasi bagaimana percabangan objek ViewGroup pada *layout* dan mengandung objek View lainnya.

Attribut-attribut Objek View

Pada subbab ini akan dijelaskan attribut-attribut object View yang digunakan dalam membuat GUI pada file activity_main.xml

- **android:id** Attribut ini merupakan pengidentifikasi dari suatu view. Attribut ini dapat digunakan untuk menjadi referensi object dari kode aplikasi seperti membaca dan memanipulasi objek tersebut (Akan dijelaskan lebih lanjut pada subbab 2.1.3). Tanda '@' dibutuhkan ketika mereferensi object dari suatu XML. Tanda '@' tersebut diikuti dengan tipe (id pada kasus ini), *slash*, dan nama (edit_message pada Listing 2.2). Tanda tambah (+) sebelum tipe hanya dibutuhkan jika ingin mendefinisikan *resource ID* untuk pertama kalinya.
- **android:layout_width** dan **android:layout_height** Attribut ini digunakan untuk mendefinisikan panjang dan lebar dari suatu objek View. Daripada menggunakan besar spesifik untuk panjang dan lebarnya, lebih baik menggunakan "wrap_content" yang menspesifikasi viewnya hanya akan sebesar yang dibutuhkan untuk memuat konten-konten dari View. Jika menggunakan "match_parent" pada kasus Listing 2.2 View akan memenuhi layar, karena besarnya akan mengikuti besar dari parentnya LinearLayout.
- **android:hint** Attribut ini merupakan *default string* untuk ditampilkan ketika objek View kosong. Daripada menggunakan *hard-coded string* sebagai *nilai* untuk ditampilkan, value "@string/edit_message" mereferensi ke sumber string pada file yang berbeda. Karena mereferensi ke sumber konkret, maka tidak dibutuhkan tanda tambah (+). Nilai string ini akan disimpan pada file Strings.xml yang ditunjukkan pada Listing 2.1.

Listing 2.1: Contoh kode pada string.xml

```

2 |     <string name="app_name">MyFirstAndroidApp</string>
3 |     <string name="edit_message">Ini adalah hint</string>
4 |     <string name="button_send">Send</string>
5 |   </resources>

```

- **android:onClick** Attribut ini akan memberitahu *system* untuk memanggil method yang sesuai namanya (contoh pada Listing 2.2 adalah `sendMessage()`) di Activity ketika pengguna melakukan klik pada *button* tersebut. Agar *system* dapat memanggil method yang tepat, method tersebut harus memenuhi kriteria berikut.
 - Access Modifier haruslah *public*.
 - Harus *void return value*nya.
 - Mempunyai View sebagai parameter satu-satunya. View ini akan diisi dengan View yang di klik.

Listing 2.2: Contoh kode file XML pada folder layout

```

1 | <LinearLayout
2 |   xmlns:android="http://schemas.android.com/apk/res/android"
3 |   xmlns:tools="http://schemas.android.com/tools"
4 |   android:layout_width="match_parent"
5 |   android:layout_height="match_parent"
6 |   android:orientation="horizontal">
7 |     <EditText android:id="@+id/edit_message"
8 |       android:layout_weight="1"
9 |       android:layout_width="0dp"
10 |       android:layout_height="wrap_content"
11 |       android:hint="@string/edit_message" />
12 |     <Button
13 |       android:layout_width="wrap_content"
14 |       android:layout_height="wrap_content"
15 |       android:text="@string/button_send"
16 |       android:onClick="sendMessage" />
17 |   </LinearLayout>

```

2.1.3 Activity

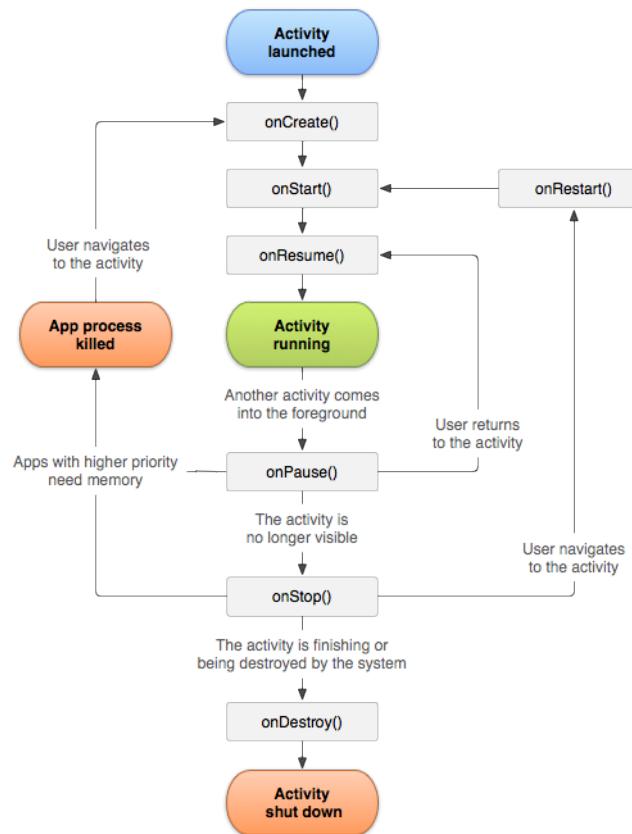
[8] Activity adalah suatu hal yang terfokuskan dengan apa yang bisa pengguna lakukan. Hampir semua *Activity* berinteraksi dengan pengguna, jadi kelas *Activity* akan membuat suatu halaman baru yang bisa ditambahkan dengan konten-konten View. Selain dapat direpresentasikan kepada pengguna dengan halaman *full-screen*, *Activity* juga dapat direpresentasikan dengan cara lain: seperti halaman *floating* atau tertanam di *Activity* lain.

Activity Lifecycle

Aktivitas dalam sistem android di atur sebagai *activity stack*. Ketika ada *Activity* baru yang dimulai, *Activity* tersebut ditempatkan di paling atas pada *stack* dan menjadi *Activity* aktif. *Activity* sebelumnya akan tetap berada di bawah *stack*, dan tidak akan muncul lagi sampai *Activity* yang baru berakhir.

Activity didasari dari empat kondisi:

- Jika *Activity* berada di latar depan pada layar, *Activity* tersebut sedang aktif.
- Jika *Activity* sudah tidak terfokuskan tetapi masih dapat terlihat, *Activity* tersebut sedang berhenti sementara. Pada kondisi ini *Activity* tersebut masih berjalan, tapi bisa diberhentikan ketika *system* berada dalam situasi kekurangan memori.



Gambar 2.3: *State diagram siklus Activity*

- Jika suatu Activity benar-benar dihalangi oleh Activity lain, Activity tersebut telah berhenti. Activity tersebut akan tetap mengingat seluruh keadaan dan infomasi anggota tetapi, Activity tersebut tidak lagi terlihat oleh pengguna jadi tampilan jendelanya akan tersembunyi dan seringkali akan diberhentikan Activitynya ketika system membutuhkan memori.
- Jika suatu Activity sedang berhenti sementara atau berhenti total, sistem dapat membuang Activity dari memory dengan cara menanyakan kepada pengguna untuk memberhentikannya atau langsung diberhentikan oleh sistem. Jika Activity tersebut ditampilkan lagi kepada pengguna, Activity tersebut harus memulai dari awal dan kembali ke keadaan sebelumnya.

Gambar 2.3 menunjukkan pentingnya alur keadaan dari suatu Activity. Gambar segi empat merepresentasikan *callback methods* yang dapat diimplementasikan untuk melakukan operasi ketika Activity berubah kondisi. Oval berwarna merupakan kondisi-kondisi utama dari suatu Activity. Ada 3 *key loops* untuk memantau suatu Activity:

- *Entire lifetime* terjadi diantara pemanggilan pertama pada `onCreate(Bundle)` sampai ke satu pemanggilan akhir `onDestroy()`. Suatu Activity akan melakukan semua persiapan pada kondisi umum pada method `onCreate()`, dan melepaskan seluruh sisa pemrosesan pada method `onDestroy()`.
- *Visible lifetime* terjadi antara pemanggilan `onStart()` sampai pemanggilan yang sesuai pada `onStop()`. Pada tahap ini pengguna dapat melihat Activity pada layar meskipun tidak berada pada *foreground* dan berinteraksi dengan pengguna.

- *Foreground lifetime* terjadi antara pemanggilan method `onResume()` sampai ke satu pemanggilan akhir `onDestroy()`. Pada tahap ini Activity berada di depan semua Activity lainnya dan sedang berinteraksi dengan user.

2.1.4 Android Sensor Framework

[8] Sebagian besar dari perangkat android sudah memiliki sensor yang mengukur gerakan, orientasi, dan berbagai keadaan lingkungan. Sensor-sensor ini dapat memberikan data mentah dengan tingkat akurasi yang tinggi. Sensor ini juga berguna untuk memantau pergerakan tiga dimensi atau posisi perangkat. Sensor ini juga dapat memantau perubahan keadaan lingkungan yang dekat dengan perangkat. Android mendukung tiga kategori sensor:

- **Sensor gerak** Sensor-sensor ini mengukur akselerasi dan rotasi pada tiga sumbu. Kategori sensor ini meliputi *accelerometers*, sensor gravitasi, *gyroscope*, dan *rotation vector*.
- **Sensor keadaan lingkungan** Sensor-sensor ini mengukur berbagai keadaan lingkungan seperti suhu udara, tekanan, pencahayaan, dan kelembaban. Kategori sensor ini termasuk barometer, fotometer dan termometer.
- **Sensor posisi** Sensor-sensor ini mengukur posisi perangkat. Kategori sensor ini meliputi sensor orientasi dan magnetometer.

Android Sensor Framework membantu developers untuk mengakses berbagai jenis sensor. Beberapa sensor berbasis perangkat keras dan beberapa sensor berbasis perangkat lunak. Sensor berbasis perangkat keras mendapatkan data dengan langsung mengukur sifat lingkungan tertentu, seperti percepatan, kekuatan medan geomagnetik, atau perubahan sudut. Sensor berbasis perangkat lunak mendapatkan data dari satu atau lebih sensor berbasis perangkat keras. Sensor berbasis perangkat lunak ini juga terkadang disebut sensor virtual atau sensor sintetis. Pada tabel berikut akan dirincikan tipe-tipe setiap sensor posisi dan gerak, deskripsi, dan penggunaan umumnya.

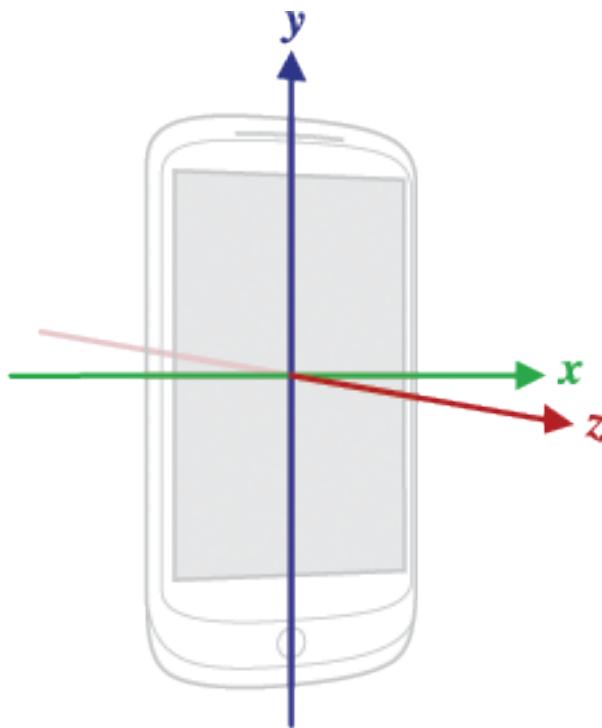
Sistem Koordinat Sensor

Pada umumnya, sensor framework menggunakan sistem tiga sumbu koordinat standar untuk mengexpresikan nilai data. Sebagian besar sensor sistem koordinat didefinisikan relatif terhadap layar perangkat bila perangkat dibuat dalam orientasi standar (lihat Gambar 2.4) Sensor-sensor yang menggunakan sistem tiga sumbu seperti Gambar 2.4 adalah sebagai berikut :

- Accelerometer
- Sensor Gravitasi
- Gyroscope
- Sensor Percepatan Linear
- Sensor Medan Geomagnetik

Tabel 2.1: Tipe-tipe Sensor pada Android

Sensor	Tipe	Deskripsi	Penggunaan umum
TYPE_ACCELEROMETER	Perangkat Keras	Mengukur percepatan dalam m/s^2 yang terjadi pada perangkat di semua tiga sumbu fisik (x,y,z), termasuk percepatan gravitasi.	Deteksi gerak(guncangan, keseimbangan, dan lain-lain)
TYPE_GRAVITY	Perangkat Lunak atau Perangkat Keras	Mengukur percepatan gravitasi dalam m/s^2 yang terjadi pada perangkat di tiga sumbu fisik (x,y, dan z)	Deteksi gerak(guncangan, keseimbangan, dan lain-lain)
TYPE_GYROSCOPE	Perangkat Keras	Mengukur rata-rata rotasi sudut dalam rad/s di tiga sumbu fisik (x,y, dan z).	Deteksi rotasi (putaran, belokan, dan lain-lain).
TYPE_LINEAR_ACCELERATION	Perangkat Lunak atau Perangkat Keras	Mengukur percepatan dalam m/s^2 yang terjadi pada perangkat di semua tiga sumbu fisik (x,y,z), tidak termasuk percepatan gravitasi.	Memantau percepatan pada suatu sumbu.
TYPE_MAGNETIC_FIELD	Perangkat Keras	Mengukur medan magnet sekitar untuk semua tiga sumbu fisik (x,y, dan z) di satuan μT .	Membuat Kompas.
TYPE_ORIENTATION	Perangkat Lunak	Mengukur derajat rotasi yang terjadi pada perangkat pada semua tiga sumbu fisik (x,y, dan z).	Menentukan posisi perangkat
TYPE_ROTATION_VECTOR	Perangkat Lunak dan Perangkat Keras	Mengukur orisentasi dari suatu perangkat dengan menyediakan tiga element dari vektor rotasi perangkat.	Deteksi gerak dan deteksi rotasi.



Gambar 2.4: Sistem koordinat (relatif dengan perangkatnya) yang digunakan oleh Sensor API

Koordinat sistem yang sumbunya tidak tertukar ketika orientasi perangkat berubah. Sistem koordinat sensor tidak pernah berubah seiring perangkatnya bergerak. Dalam aplikasi android tidak dapat diassumsikan bahwa standar orientasi perangkat android adalah *portrait*. Kebanyakan perangkat *Tablet* standar orientasinya adalah *landscape*. Sistem koordinat sensor selalu di dasarkan pada orientasi dasar dari suatu perangkat android.

Mengidentifikasi Sensor dan Kapabilitas Sensor

Android Sensor Framework menyediakan beberapa method yang dapat membuat developer mudah untuk menentukan sensor mana yang akan digunakan. APInya juga dapat menyediakan method yang memungkinkan penggunanya menentukan kapabilitas masing-masing sensor, seperti jangkauan maksimum, resolusi, dan kebutuhan dayanya. Untuk mengidentifikasi sensor-sensor yang ada pada perangkat, hal pertama yang perlu dilakukan adalah mendapatkan referensi sensor tersebut. Untuk mendapatkan referensi tersebut, dapat dilakukan dengan membuat instansiasi dari kelas SensorManager dan memanggil method getSystemService() dan memasukkan isi Parameternya dengan SENSOR_SERVICE. Contohnya pada Listing 2.3.

Listing 2.3: Contoh inisialisasi kelas SensorManager

```

1| private SensorManager mSensorManager;
2| ...
3| mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

```

Kemudian untuk mendapatkan daftar dari setiap sensor pada suatu perangkat dapat dilakukan dengan cara memanggil method getSensorList() dan menggunakan konstanta TYPE_ALL pada kelas Sensor. Contohnya pada Listing 2.4

Listing 2.4: Contoh untuk mendapatkan daftar dari setiap sensor yang ada

```
1| List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

Namun jika ingin mendapatkan sensor-sensor yang sesuai dengan tipe sensor yang diberikan, dapat menggunakan TYPE_GYROSCOPE, TYPE_LINEAR_ACCELERATION, TYPE_GRAVITY, atau TYPE_GRAVITY.

Untuk menentukan jenis tertentu dari sensor yang ada pada perangkat dapat didapatkan dengan method getDefaultSensor() dengan dimasukkan dengan konstanta yang berada pada kelas Sensor. Jika perangkatnya memiliki lebih dari satu sensor dari tipe sensor yang diberikan, salah satu dari sensor tersebut akan dianggap sebagai sensor dasar. Jika sensor dasanya tidak ada untuk sensor tersebut, perangkat tersebut berarti tidak memiliki sensor dengan jenis yang diberikan. Listing 2.5 adalah contoh untuk mengecek apakah perangkat yang digunakan memiliki sensor dengan jenis yang diberikan.

Listing 2.5: Contoh untuk mengecek apakah perangkat yang digunakan memiliki sensor dengan jenis yang diberikan

```

1 private SensorManager mSensorManager;
2 ...
3 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
4 if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
5     // Sukses! Perangkat ini memiliki sensor magnetometer.
6 }
7 else {
8     // Gagal! Perangkat ini tidak memiliki sensor magnetometer.
9 }
```

Jika ingin membatasi versi atau vendor dari sensor yang akan digunakan, dapat menggunakan method getVendor() dan getVersion(). Seperti pada Listing 2.6 yang mengharuskan sensor gravitasi bervendor "Google Inc." dan memiliki versi 3. Jika sensor tersebut tidak tersedia pada perangkat, sensor accelerometerlah yang digunakan.

Listing 2.6: Contoh untuk mengecek apakah perangkat yang digunakan memiliki sensor dengan jenis yang diberikan

```

1 private SensorManager mSensorManager;
2 private Sensor mSensor;
3 ...
4 ...
5
6 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
7 mSensor = null;
8
9 if (mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY) != null){
10     List<Sensor> gravSensors = mSensorManager.getSensorList(Sensor.TYPE_GRAVITY);
11     for(int i=0; i<gravSensors.size(); i++) {
12         if ((gravSensors.get(i).getVendor().contains("Google Inc.")) &&
13             (gravSensors.get(i).getVersion() == 3)){
14             // menggunakan sensor gravitasi versi 3.
15             mSensor = gravSensors.get(i);
16         }
17     }
18 }
19 if (mSensor == null){
20     // Use the accelerometer.
21     if (mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null){
22         mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
23     }
24     else{
25         // Tidak ada sensor gravitasi dan sensor accelerometer!
26     }
27 }
```

Salah satu method yang sangat berguna lagi adalah, getMinDelay(). Method ini digunakan untuk mengetahui minimum interval waktu suatu sensor dapat menerima data dalam mikrodetik. Jika method getMinDelay() mengembalikan nilai nol, hal ini berarti sensor ini akan mengembalikan data setiap kali ada perubahan nilai pada sensor tersebut.

Memonitor Nilai Sensor

Untuk memonitor data mentah dari sensor, dibutuhkan untuk mengimplement dua buah method callback yang berada pada interface SensorEventListener. Kedua method tersebut adalah onAccuracyChanged(Sensor sensor, int accuracy) dan onSensorChanged(SensorEvent event). Sistem android akan memanggil kedua method ini ketika terjadi salah satu kondisi ini:

- **Perubahan akurasi sensor.**

Dalam kasus ini sistem memanggil method onAccuracyChanged(Sensor sensor, int accuracy). Parameter sensor akan diberikan objek Sensor yang telah berubah akurasinya, dan parameter accuracy adalah nilai akurasi sensor yang baru.

- **Sensor memberitahu adanya nilai baru.**

Dalam kasus ini sistem memanggil method onSensorChanged(SensorEvent event), dengan parameter event akan diisi dengan objek SensorEvent untuk mendapatkan nilai barunya. Objek SensorEvent Mengandung semua infomasi tentang data sensor yang baru, termasuk: akurasi dari data, sensor yang mendapatkan data, dan catatan waktu data tersebut didapatkan, dan data yang baru yang telah didapatkan.

Pada Listing 2.7 akan ditunjukkan bagaimana menggunakan method onSensorChanged(SensorEvent event) untuk memonitor data dari sensor cahaya. Pada Listing 2.7 akan menampilkan data mentah dari sensor ke TextView yang telah didefinisikan pada file main.xml sebagai sensor_data.

Listing 2.7: Contoh memonitor data mentah pada sensor cahaya

```

1 public class SensorActivity extends Activity implements SensorEventListener {
2     private SensorManager mSensorManager;
3     private Sensor mLight;
4
5     @Override
6     public final void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(R.layout.main);
9
10        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
11        mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
12    }
13
14    @Override
15    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
16        // Hal yang perlu dilakukan aplikasi ketika akurasinya berubah.
17    }
18
19    @Override
20    public final void onSensorChanged(SensorEvent event) {
21        // Sensor cahaya akan mengembalikan 1 nilai saja.
22        // Banyak sensor lain yang akan mengembalikan lebih dari 1 nilai.
23        float lux = event.values[0];
24        // Hal yang perlu dilakukan ketika ada perubahan data.
25    }
26
27    @Override
28    protected void onResume() {
29        super.onResume();
30        mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
31    }
32
33    @Override
34    protected void onPause() {
35        super.onPause();
36        mSensorManager.unregisterListener(this);
37    }
38 }
```

Pada method `onSensorChanged(SensorEvent event)`, struktur nilai-nilai yang dikembalikan akan dijelaskan pada subbab 2.1.4. Pada method `onResume()`, ada pemanggilan method `registerListener()`. Method `registerListener()` ini berguna untuk menspesifikasi waktu delay pada pemanggilan `onSensorChanged()`. Untuk menspesifikasi delaynya dapat menggunakan konstanta yang ada pada kelas `SensorManager`. Konstanta-konstanta tersebut adalah `SENSOR_DELAY_NORMAL` (200.000 mikrodetik), `SENSOR_DELAY_GAME` (20.000 mikrodetik), `SENSOR_DELAY_UI` (60.000 mikrodetik), atau `SENSOR_DELAY_FASTEST` (0 mikrodetik). Pengaturan dasar untuk waktu delay ini menggunakan konstanta `SENSOR_DELAY_NORMAL`. Perlu diperhatikan juga pemanfaatan sensor ketika activity sedang di berhentikan sementara maupun dilanjutkan kembali. Sistem tidak boleh tetap merekam sensor ketika activity tidak aktif. Hal ini diperlukan karena pada saat perangkat android menggunakan sensor, perangkat akan menggunakan tenaga yang banyak. Akan lebih baik jika penggunaan sensor diberhentikan ketika activitynya sudah tidak lagi digunakan.

Struktur Nilai yang Dikembalikan oleh Sensor

Setiap sensor android akan mengembalikan nilai-nilainya dengan struktur-struktur tertentu. Pada bab ini akan dijelaskan secara detil struktur nilai sistem android dalam mengembalikan nilai-nilai yang diperoleh dari sensor. Nilai ini akan didapatkan dengan tipe data array of float. Besar dan isi dari array tergantung pada sensor yang sedang di pantau. Berikut ini adalah struktur-struktur nilai dari setiap sensor pada sistem android :

- **TYPE_ACCELEROMETER**

Semua nilai didefinisikan sebagai satuan m/s^2

- `values[0]`: Percepatan yang terjadi pada sumbu x dikali -1
- `values[1]`: Percepatan yang terjadi pada sumbu y dikali -1
- `values[2]`: Percepatan yang terjadi pada sumbu z dikali -1

Sensor ini mengukur percepatan(Ad) yang diterapkan pada perangkat. Sensor tersebut dapat mengukur percepatan dengan mengukur gaya(F_s) yang terjadi pada sensor menggunakan relasi berikut:

$$Ad = -\Sigma F_s/mass$$

Secara khusus, gravitasi selalu mempengaruhi percepatan yang diukur :

$$Ad = -g - \Sigma F/mass$$

Karena inilah ketika perangkat android sedang diam, accelerometer membaca percepatan gravitasi sebesar $g = 9.81m/s^2$. Demikian pula ketika perangkat android sedang dalam keadaan jatuh bebas. Perangkat akan mempercepat menuju ke tanah pada percepatan $9.81m/s^2$, sehingga accelerometer membaca percepatan total sebesar $0m/s^2$. Suatu saat akan dibutuhkan untuk mengukur percepatan asli yang terjadi pada perangkat, sehingga kontribusi gravitasi harus dieliminasi. Hal ini bisa dilakukan dengan menerapkan *high-pass* filter. Sebaliknya, *low-pass* filter dapat digunakan untuk mendapatkan nilai gravitasi saja.

Listing 2.8: Implementasi *low-pass* filter

```

1  public void onSensorChanged(SensorEvent event)
2  {
3      // alpha dikalkulasikan sebagai t / (t + dT)
4      // dengan t adalah low-pass filter's time-constant
5      // dan dT, rata-rata event tersampaikan
6
7      final float alpha = 0.8;
8
9      gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
10     gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
11     gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];
12
13     linear_acceleration[0] = event.values[0] - gravity[0];
14     linear_acceleration[1] = event.values[1] - gravity[1];
15     linear_acceleration[2] = event.values[2] - gravity[2];
16 }

```

Low-pass filter dapat diimplementasikan pada Listing 2.8

- **TYPE_MAGNETIC_FIELD**

Sensor ini mengukur medan magnet sekitar perangkat pada sumbu X,Y, dan Z dalam satuan micro-Tesla.

- **TYPE_GYROSCOPE**

Sensor ini mengukur rata-rata perputaran pada perangkat yang berputar di sumbu X,Y, dan Z dalam satuan radians/second. Sistem koordinat yang digunakan sama dengan sistem koordinat pada sensor percepatan(Accelerometer). Jika perangkat berputar berlawanan arah jarum jam pada sumbu tertentu, maka rotasi yang terjadi akan bernilai positif. Perhatikan bahwa standar perputaran ini adalah definisi matematika standar pada rotasi positif.

- values[0]: Percepatan angular pada sumbu X.
- values[1]: Percepatan angular pada sumbu Y.
- values[2]: Percepatan angular pada sumbu Z.

Biasanya keluaran dari gyroscope terintegrasi dari waktu ke waktu untuk menghitung rotasi yang menggambarkan perubahan sudut atas langkah waktu, misalnya pada Listing 2.9

Listing 2.9: contoh implementasi gyroscope

```

1  private static final float NS2S = 1.0f / 1000000000.0f;
2  private final float[] deltaRotationVector = new float[4]();
3  private float timestamp;
4
5  public void onSensorChanged(SensorEvent event) {
6      // Pada tahapan ini delta rotasi akan dikalikan dengan rotasi saat ini
7      // setelah mengcomputasinya dari data gyro.
8      if (timestamp != 0) {
9          final float dT = (event.timestamp - timestamp) * NS2S;
10         // Sumbu dari rotasi, masih belum di normalisasi.
11         float axisX = event.values[0];
12         float axisY = event.values[1];
13         float axisZ = event.values[2];
14
15         // Menghitung percepatan angular
16         float omegaMagnitude = sqrt(axisX*axisX + axisY*axisY + axisZ*axisZ);
17
18         // Normalisasi rotasi vektor jika cukup besar untuk mendapatkan sumbunya.
19         if (omegaMagnitude > EPSILON) {
20             axisX /= omegaMagnitude;
21             axisY /= omegaMagnitude;
22             axisZ /= omegaMagnitude;

```

```

23         }
24
25         // Integrate around this axis with the angular speed by the time step
26         // in order to get a delta rotation from this sample over the time step
27         // We will convert this axis-angle representation of the delta rotation
28         // into a quaternion before turning it into the rotation matrix.
29         float thetaOverTwo = omegaMagnitude * dT / 2.0f;
30         float sinThetaOverTwo = sin(thetaOverTwo);
31         float cosThetaOverTwo = cos(thetaOverTwo);
32         deltaRotationVector[0] = sinThetaOverTwo * axisX;
33         deltaRotationVector[1] = sinThetaOverTwo * axisY;
34         deltaRotationVector[2] = sinThetaOverTwo * axisZ;
35         deltaRotationVector[3] = cosThetaOverTwo;
36     }
37     timestamp = event.timestamp;
38     float[] deltaRotationMatrix = new float[9];
39     SensorManager.getRotationMatrixFromVector(deltaRotationMatrix, deltaRotationVector);
40     // User code should concatenate the delta rotation we computed with the current rotation
41     // in order to get the updated rotation.
42     // rotationCurrent = rotationCurrent * deltaRotationMatrix;
43 }
44 }
```

Dalam prakteknya, gyroscope *noise* dan *offset* akan menyebabkan beberapa kesalahan yang harus dikompensasi. Cara untuk mengkompensasinya biasanya dilakukan dengan menggunakan informasi dari sensor lain.

• TYPE_GRAVITY

Sensor ini menunjukkan arah dan besarnya vektor gaya gravitasi. Sensor ini mengembalikan nilai dengan satuan m/s^2 . Sistem koordinat sama seperti sistem koordinat yang umum digunakan sensor percepatan.

Catatan: Bila perangkat sedang diam, maka keluaran dari sensor gravitasi harus identik dengan accelerometer.

• TYPE_LINEAR_ACCELERATION

Sensor yang menunjukkan percepatan pada setiap sumbu perangkat, tidak termasuk percepatan yang terjadi karena gravitasi. Nilai diberikan dalam satuan m/s^2 . Sistem koordinat yang digunakan sama seperti sistem koordinat yang digunakan sensor percepatan. Keluaran dari sensor accelerometer, gravitasi dan percepatan linear harus mengikuti aturan berikut:

$$\text{percepatan} = \text{gravitasi} + \text{percepatan linear}$$

• TYPE_ORIENTATION

Semua nilai adalah sudut dalam derajat.

- values[0]: Azimuth, sudut diantara arah magnetik utara dengan sumbu y, sekitar sumbu z (0 sampai 359). 0 = Utara, 90 = Timur, 180 = Selatan, 270 = Barat
- values[1]: Pitch, rotasi sekitar sumbu x (-180 sampai 180), dengan nilai positif ketika sumbu x bergerak menuju sumbu y.

- `values[2]`: Roll, perputaran sekitar sumbu y (-90 sampai 90) pada kondisi portrait, sensor akan bernilai 0. Pada kondisi landscape ke kanan sensor akan bernilai 90 dan sebaliknya yaitu kondisi landscape ke kiri sensor akan bernilai -90.

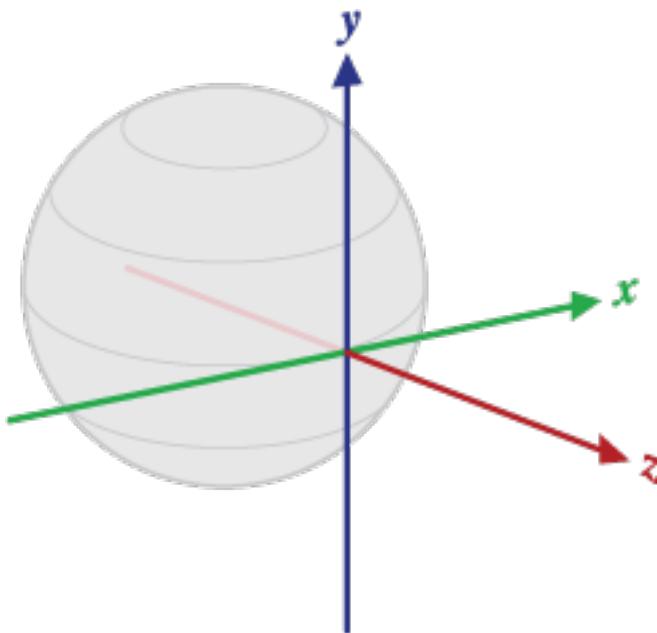
Catatan: Definisi ini berbeda dengan definisi yaw, pitch, dan roll yang digunakan pada aviasi yang sumbu X adalah sepanjang sisi bidang.

Catatan: Sensor ini sudah tidak digunakan lagi(deprecated), yang digunakan sekarang adalah sensor rotasi vector.

• TYPE_ROTATION_VECTOR

Sensor ini merepresentasikan orientasi perangkat dengan kombinasi dari sumbu dan sudut. Perangkat akan di putar sebesar sudut θ mengelilingi sumbu x, y, z . Tiga elemen dari vektor rotasi adalah $(x \sin(\frac{\theta}{2}), y \sin(\frac{\theta}{2}), z \sin(\frac{\theta}{2}))$, sehingga besarnya vektor rotasi sama dengan $\sin(\frac{\theta}{2})$, dan arah vektor rotasi sama dengan sumbu rotasi. Tiga elemen dari vektor rotasi sama dengan tiga komponen terakhir pada unit kuaternion $(\cos(\frac{\theta}{2}), x \sin(\frac{\theta}{2}), y \sin(\frac{\theta}{2}), z \sin(\frac{\theta}{2}))$ yang dijelaskan pada subbab 2.3. Elemen dari vektor rotasi tak memiliki satuan. Sistem koordinat yang digunakan sama dengan sistem koordinat yang digunakan pada sensor percepatan. Referensi koordinat didefinisikan sebagai basis orthonormal, yaitu:

- X didefinisikan sebagai perkalian dot product **Y.Z**
- Y merupakan tangensial ke tanan pada lokasi perangkat saat ini dan menunjuk ke arah utara.
- Z menghadap ke langit dan tegak lurus dengan tanah. Untuk lebih jelasnya dapat dilihat pada Gambar 2.5



Gambar 2.5: Sistem koordinat sensor rotasi vektor terhadap Bumi

- `values[0]`: $x \sin(\frac{\theta}{2})$

- values[1]: $y \sin(\frac{\theta}{2})$
- values[2]: $z \sin(\frac{\theta}{2})$
- values[3]: $\cos(\frac{\theta}{2})$
- values[4]: Perkiraan akurasi (dalam radians) (-1 jika tidak tersedia)

2.2 Google VR SDK

Google VR SDK[9] digunakan untuk membantu dalam pembuatan aplikasi Virtual Reality pada *smartphone*. Google VR SDK memberikan beberapa fitur sebagai berikut :

- **Binocular rendering:** Fitur untuk tampilan layar terpisah untuk masing-masing dalam pandangan VR.
- **Spatial audio:** Fitur untuk mengeluarkan suara yang datang dari daerah-daerah tertentu dari dunia VR.
- **Head movement tracking:** Fitur untuk mendapatkan memperbaharui pengelihatan dunia VR yang sesuai dengan gerakan kepala pengguna.
- **Trigger input:** Fitur untuk memberikan input pada dunia VR dengan menekan tombol.

Ada beberapa persyaratan untuk menggunakan Google VR SDK, persyaratan tersebut adalah:

- Android Studio versi 1.0 atau lebih.
- Android SDK versi 23
- Gradle versi 23.0.1 atau lebih. Android Studio akan membantu meningkatkan versinya jika versinya terlalu rendah.
- Perangkat Android fisik yang menjalankan Android versi 4.4 (KitKat) atau lebih.

Dalam membuat aplikasi Google Cardboard VR membutuhkan beberapa API(Application Program Interface) dari Google VR SDK. API-API umum yang akan digunakan adalah sebagai berikut.

- API audio: API untuk mengimplementasikan ***Spatial Audio*** (Metode untuk menspasialisasikan sumber suara dalam ruang tiga dimensi).
- API base: API untuk fondasi dari suatu aplikasi Google VR.

2.2.1 API Audio

[9] API ini membantu developer untuk menspasialisasikan sumber suara dalam tiga dimensi, termasuk jarak dengan tinggi isyarat sumber suara. Pada API ini hanya terdapat satu class utama yaitu **GvrAudioEngine**. **GvrAudioEngine** mampu memutarkan suara secara spasial dalam dua cara yang berbeda :

- Metode pertama dikenal sebagai *Sound Object rendering*. Metode ini memungkinkan pengguna membuat sumber suara virtual dalam ruang tiga dimensi.

- Metode kedua memungkinkan pengguna untuk memutar kembali rekaman *Ambisonic soundfield*. Rekaman *Ambisonic soundfield* adalah file audio *multi-channel* yang telah terspasialisasi.

API ini juga dapat memutarkan suara secara *stereo*. Kelas **GvrAudioEngine** memiliki tiga buah *nested class* yaitu:

- **GvrAudioEngine.DistanceRolloffModel**: Kelas ini mendefinisikan konstanta-konstanta yang merepresentasikan perbedaan jarak dari efek model-model rolloff.
- **GvrAudioEngine.MaterialName**: Kelas ini mendefinisikan konstanta-konstanta yang merepresentasikan bahan permukaan ruangan untuk disesuaikan dengan efek suara pada suatu ruangan.
- **GvrAudioEngine.RenderingMode**: Kelas ini mendefinisikan konstanta-konstanta untuk menyesuaikan dengan mode rendering. Semakin baik kualitas renderin akan semakin besar penggunaan CPU(Central Processing Unit).

2.2.2 API Base

[9] API ini digunakan sebagai fondasi dari suatu aplikasi Google VR. Fitur-fitur Binocular rendering, Head movement tracking, dan Trigger input diimplementasikan pada API ini. Kelas-kelas penting yang ada di API ini adalah AndroidCompat, Eye, GvrActivity, GvrView, HeadTransform, Viewport.

- **AndroidCompat**

Kelas ini merupakan kelas utilitas untuk menggunakan fitur VR. Fitur-fitur ini mungkin tidak tersedia pada semua versi android. Kelas ini memiliki method-method sebagai berikut:

- `setSustainedPerformanceMode(Activity activity, boolean enabled)`:
Method ini digunakan untuk mengubah window android ke mode performa secara berkelanjutan.
- `public static void setVrModeEnabled (Activity activity, boolean enabled)`:
Mengatur pengaturan yang tepat untuk "mode VR" pada suatu Activity. Method ini tidak digunakan karena hanya dapat digunakan pada Android N+.
- `public static boolean trySetVrModeEnabled (Activity activity, boolean enabled)`: Method ini kegunaanya sama dengan method **setVrModeEnabled (Activity activity, boolean enabled)**, namun mengembalikan boolean true jika berhasil dan sebaliknya.

- **Eye**

Kelas ini mendefinisikan detil perenderan stereoskopik mata. Method penting yang dimiliki kelas ini adalah **public float[] getEyeView ()**. Method ini mengembalikan matriks yang mentransformasikan kamera virtual ke mata. Transformasi yang diberikan termasuk melacak rotasi kepala, perubahan posisi dan perubahan IPD(interpupillary distance).

- **GvrActivity**

Kelas ini merupakan Activity dasar yang menyediakan integrasi yang mudah dengan headset Google VR. Kelas ini mengekspos kejadian untuk berinteraksi dengan headset Google VR dan menangani detil-detil yang biasa diperlukan saat membuat suatu Activity untuk perenderan

VR. Activity ini membuat layar tetap menyala selama perangkat android bergerak. Jika tidak ada pergerakan dari perangkat android maka layar reguler (*wakeclock*) akan ditampilkan. Pada kelas ini terdapat method **onCardboardTrigger ()** untuk mendeteksi ketika Cardboard Trigger sedang ditarik dan dilepaskan (Magnet yang berada pada sisi Google Cardboard).

- **GvrView**

Kelas ini merupakan kelas View yang menyediakan perenderan VR. Kelas ini didesain untuk berkerja pada mode layar penuh dengan orientasi *landscape* atau *reverse landscape*. Kelas View ini dapat digunakan dengan mengimplementasikan salah satu Interface perenderan. Interface-interface tersebut adalah:

- **GvrView.StereoRenderer**: Interface untuk perenderan detil stereoskopik secara abstrak oleh perender.
- **GvrView.Renderer**: Interface untuk mesin yang kompleks yang membutuhkan untuk menangai semua detil perenderan.

Kelas GvrView.Renderer jarang digunakan dan sebaiknya tidak digunakan jika tidak sangat dibutuhkan. Ketika suatu kelas mengimplementasikan Kelas GvrView.StereoRenderer, kelas tersebut harus mengimplementasikan method-method berikut ini:

- **public void onNewFrame(HeadTransform headTransform)**
method ini terpanggil ketika Frame baru akan digambar. Method ini memungkinkan untuk membedakan antara perenderan pandangan mata dan frame-frame yang berbeda. Setiap operasi per-frame harus tidak spesifik pada satu tampilan saja.
- **public abstract void onDrawEye (Eye eye)**
Method ini meminta untuk menggambar suatu konten dari sudut pandang mata.
- **public abstract void onFinishFrame (Viewport viewport)**
Method ini dipanggil ketika suatu frame telah selesai. Dengan pemanggilan ini, konten frame telah di gambar dan jika koreksi distorsi diaktifkan, koreksi distorsi akan diterapkan. Setiap perrenderan pada tahap ini relatif terhadap seluruh permukaan, tidak terhadap satu pandangan mata tunggal.
- **public abstract void onRendererShutdown ()**
Method ini dipanggil ketika thread perender sedang menutup. Melepaskan sumber GL(Graphics Library) dan sedang melakukan penutupan operasi pada thread perender. Dipanggil hanya jika sebelumnya ada pemanggilan method onSurfaceCreated.
- **public abstract void onSurfaceChanged (int width, int height)**
Dipanggil ketika ada perubahan dimensi permukaan. Semua nilai adalah relatif ke ukuran yang dibutuhkan untuk merender sebuah mata.
- **public abstract void onSurfaceCreated (EGLConfig config)**
Method ini dipanggil ketika suatu permukaan dibangun atau dibangun ulang.

- **HeadTransform**

Method ini mendeskripsikan transformasi kepala secara independen dari setiap parameter

mata. Kelas ini digunakan di kelas GvrView.StereoRenderer sebagai parameter pada method **onNewFrame**. Method-method yang perlu diperhatikan pada kelas ini adalah:

- **public void getHeadView (float[] headView, int offset)**

Method ini digunakan untuk mendapatkan matriks transformasi dari camera virtual ke kepala. Kepala disini didefinisikan sebagai titik tengah diantara kedua mata. Matriks yang didapatkan akan disimpan pada parameter **headView**.

- **public void getQuaternion (float[] quaternion, int offset)**

Method ini digunakan untuk mendapatkan kuaternion yang merepresentasikan rotasi kepala.

- **Viewport**

Kelas ini didefinisikan sebagai *viewport*(area pandang) berbentuk persegi.

2.3 Teori Kuaternion

Pada Android SDK **SensorEvent.values** [8] tipe sensor **Sensor.TYPE_ROTATION_VECTOR**, yaitu tipe sensor yang mendeteksi vektor perputaran pada *smartphone*. Tipe sensor ini dijelaskan akan mengembalikan nilai-nilai dari komponen kuaternion. Kuaternion[10] adalah objek penggabungan dari suatu skalar dengan suatu vektor, sesuatu yang tidak dapat didefinisikan dalam aljabar linear biasa. Kuaternion ditemukan oleh William Rowan Hamilton dengan memperpanjang notasi dari bilangan kompleks menjadi Kuaternion.

2.3.1 Struktur Ajabar

Karena Kuaternion merupakan bilangan kompleks yang diperpanjang notasinya, struktur aljabar Kuaternion hampir mirip dengan bilangan kompleks. Untuk mengerti struktur-struktur aljabar kuaternion, diperlukan untuk mengerti bilangan kompleks terlebih dahulu. Berikut adalah penjelasan singkat tentang bilangan kompleks.

Bilangan Kompleks

[10] Bilangan kompleks adalah bilangan yang merupakan gabungan dari bilangan imajiner dengan bilangan riil. Notasi umum dari bilangan kompleks adalah :

$$a + bi \tag{2.1}$$

Pada notasi 2.1 bilangan a dengan b merupakan bilangan riil, dan i merupakan bilangan imajiner tertentu yang memiliki sifat $i^2 = -1$. Bilangan kompleks juga dapat beroperasi dengan bilangan kompleks lainnya seperti penjumlahan, perkalian, pengurangan, dan pembagian. Berikut adalah contoh-contoh operasi pada bilangan kompleks 2.1 dengan bilangan kompleks $c + di$:

- Penjumlahan

$$(a + bi) + (c + di) = (a + c) + i(b + d)$$

- Perkalian

$$(a + bi)(c + di) = (acbd) + (bc + ad)i$$

- Pengurangan

$$(a + bi) - (c + di) = (a - c) + i(b - d)$$

- Pembagian

$$\frac{(a + bi)}{(c + di)} = \frac{(ac + bd)}{c^2 + d^2} + i \frac{bc - ad}{c^2 + d^2}$$

Pada operasi penjumlahan dan perkalian untuk kedua bilangan kompleks tersebut memiliki hukum assosiatif dan komutatif. Notasi 2.2 menunjukkan bagaimana bagaimana kedua hukum tersebut berlaku pada penjumlahan.

$$\begin{aligned} (a + ib) + (c + id) &= (a + c) + i(b + d) = \\ (c + id) + (a + ib) &= (c + a) + i(d + b) \end{aligned} \tag{2.2}$$

Suatu bilangan dapat dikatakan konjugasi kompleks dari suatu bilangan kompleks jika nilai bilangan riilnya sama, tetapi nilai bilangan imajinernya berlawanan dengan nilai pada bilangan kompleks tersebut. Maka konjugasi kompleks dari bilangan kompleks 2.1 adalah $a - bi$.

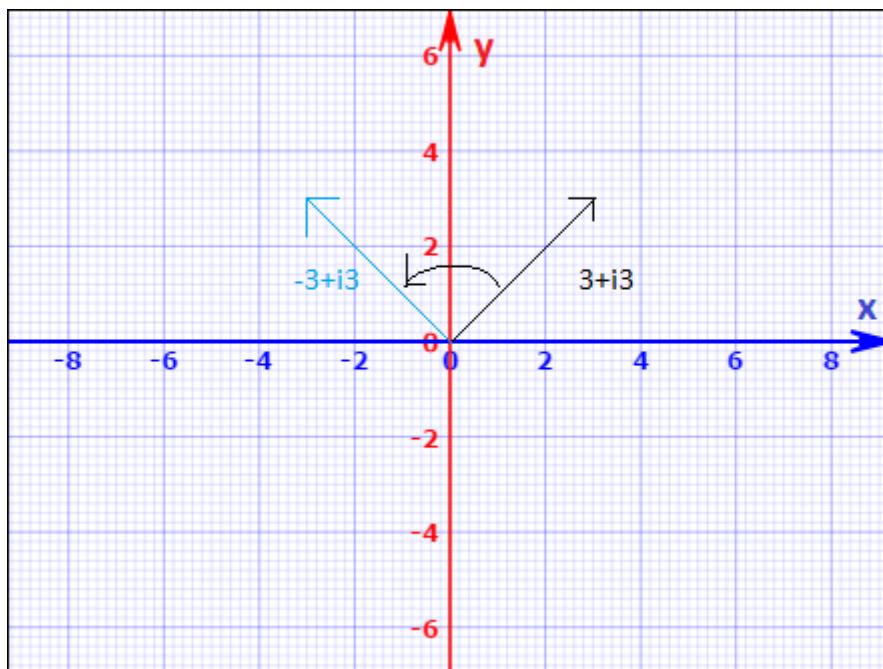
Bilangan kompleks ini dapat digunakan untuk rotasi vektor pada bidang dua dimensi. Rotasi ini dapat dilakukan dengan mengalikan suatu vektor dengan bilangan imajiner i . Mengalikan suatu vektor dengan bilangan imajiner i akan memutar vektor sebesar 90° berlawanan arah jarum jam. Mengalikan suatu vektor dengan bilangan imajiner i^2 akan memutar vektor sebesar 180° berlawanan arah jarum jam. Untuk memperjelas perputaran dengan bilangan kompleks diberikan contoh berikut: Sebuah vektor $v = 3 + i3$ akan diputar 90° berlawanan arah jarum jam dengan mengalikan vektor tersebut dengan bilangan imajiner i . Maka vektor hasil perputarannya(v') adalah :

$$\begin{aligned} v' &= i(3 + i3) \\ &= i3 + i^23 \\ &= i3 + (-1)3 \\ &= -3 + i3 \end{aligned} \tag{2.3}$$

Dengan bilangan pada bilangan riil diasumsikan sebagai nilai pada sumbu x dan bilangan yang dikalikan dengan bilangan i diasumsikan pada sumbu y($x + iy$) Seperti yang ditunjukkan pada Gambar 2.6. Oleh karena itu rumus perputaran menggunakan bilangan kompleks dapat di rumuskan sebagai berikut:

$$v' = v \times (\cos \theta + i \sin \theta)$$

dengan θ adalah besar sudut perputaran. Jika vektor $v = 3 + i3$ diputar 30° berlawanan arah jarum



Gambar 2.6: Contoh perputaran dengan bilangan kompleks

jam menggunakan konsep diatas, akan menghasilkan vektor berikut:

$$\begin{aligned}
 v' &= (3 + i3)(\cos 30^\circ + i \sin 30^\circ) \\
 &= (3 + i3)\left(\frac{\sqrt{3}}{2} + i\frac{1}{2}\right) \\
 &= \frac{3\sqrt{3} - 3}{2} + i\frac{3\sqrt{3} + 3}{2} \\
 &= 1.098 + i4.098
 \end{aligned} \tag{2.4}$$

Dari persamaan tersebut dapat divisualisasikan pada Gambar 2.7.

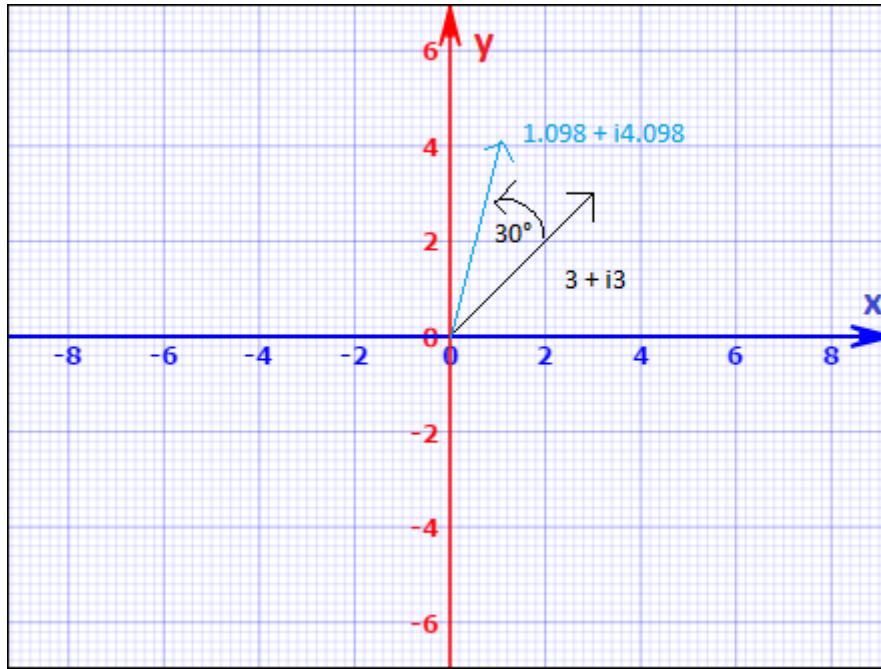
2.3.2 Aljabar Kuaternion dan Operasi-operasi pada Kuaternion

[10] Kuaternion ditemukan oleh ahli matematika dan astronomi Inggris, William Rowan Hamilton, dengan memperpanjang aritmatika dari bilangan kompleks. Dari penemuan tersebut William Rowan Hamilton menemukan bahwa dia tidak hanya membutuhkan bilangan imajiner i saja untuk melakukan rotasi pada ruang tiga dimensi. Dia menemukan bahwa dia juga membutuhkan tiga komponen imajiner lainnya yaitu i, j dan k . Persamaan umum Kuaternion memiliki empat bilangan riil atau skalar. Persamaan tersebut adalah

$$q = q_0 + iq_1 + jq_2 + kq_3 \tag{2.5}$$

Ketiga komponen tersebut memiliki relasi sebagai berikut:

$$i^2 = j^2 = k^2 = ijk = -1$$



Gambar 2.7: Contoh perputaran tiga puluh derajat dengan bilangan kompleks

Hasil dari perkalian dua *kuaternion* memiliki aturan yang lebih rumit, sehingga memiliki aturan-aturan khusus. Berikut aturan-aturan khususnya :

$$\begin{aligned} ij &= k = -ji \\ jk &= i = -kj \\ ki &= j = -ik \end{aligned} \tag{2.6}$$

Ketiga persamaan diatas mirip dengan aturan tangan kanan (*right-hand rule*) pada perkalian cross product dari suatu vector. Pada Gambar 2.8, *A* berperan sebagai *i*, *B* berperan sebagai *j*, dan *C* berperan sebagai *k*.

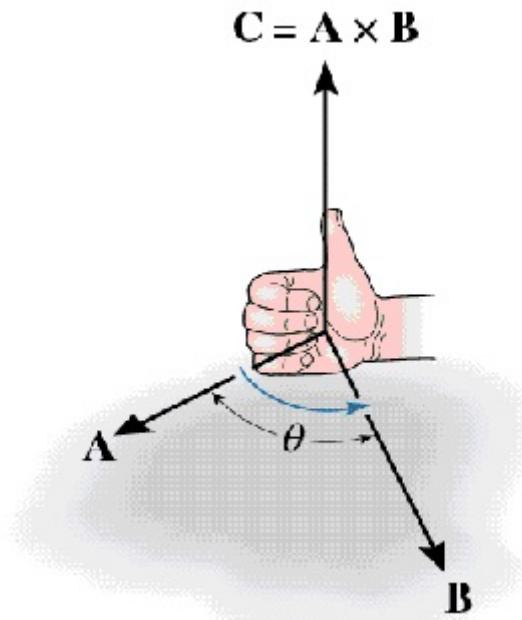
Seperti pada bilangan kompleks, kuaternion juga memiliki konjugasinya. Konjugasi kuaternion ini akan digunakan untuk melakukan operasi rotasi tiga dimensi(Akan dijelaskan pada subsubbab berikut). Sama dengan konjugasi pada bilangan kompleks, kuaternion yang bilangan riilnya sama, dan bilangan imajinernya berlawanan dengan kuaternionnya disebut dengan konjugasi kuaternion. Oleh karena itu konjugasi kuaternion dari notasi kuaternion 2.5 adalah:

$$q = q_0 - iq_1 - jq_2 - kp_3$$

Operasi pada Kuaternion

Dua buah kuaternion dapat dikatakan identik jika dan hanya jika kedua kuaternion memiliki komponen yang identik.

$$p = p_0 + ip_1 + jp_2 + kp_3$$



Gambar 2.8: Right-hand rule dalam *cross product* vektor

dan,

$$q = q_0 + iq_1 + jq_2 + kq_3$$

maka $p = q$ jika dan hanya jika

$$\begin{aligned} p_0 &= q_0 \\ p_1 &= q_1 \\ p_2 &= q_2 \\ p_3 &= q_3 \end{aligned} \tag{2.7}$$

Penjumlahan dari kedua kuaternion diatas dapat didefinisikan sebagai komponen penjumlahan yaitu:

$$(p + q) = (p_0 + q_0) + i(p_1 + q_1) + j(p_2 + q_2) + k(p_3 + q_3)$$

Perkalian dari kedua kuaternion diatas dapat didefinisikan sebagai komponen perkalian yaitu:

$$pq = (p_0 + ip_1 + jp_2 + kp_3)(q_0 + iq_1 + jq_2 + kq_3)$$

Begitu pula untuk komponen pengurangan dengan pembagian. Dari keempat operasi kuaternion tersebut, operasi kuaternion yang digunakan untuk rotasi bidang tiga dimensi adalah operasi perkalian. Fungsi rotasi vektor dapat menggunakan operasi kuaternion seperti pada bilangan kompleks, dengan rumus:

$$v' = qvq^*$$

dengan,

- $v = 1 + x_v i + y_v j + z_v k$

- $q = q_0 + iq_1 + jq_2 + kq_3$

- $q^* = q_0 - iq_1 - jq_2 - kq_3$
- $v' = 1 + x_v'i + y_v'j + z_v'k$

Seperti pada bilangan kompleks, bilangan q_0, iq_1, jq_2, kq_3 akan memiliki nilai sebagai berikut jika suatu kuaternion ingin digunakan untuk rotasi tiga dimensi:

$$\begin{aligned} q_0 &= \cos\left(\frac{\theta}{2}\right) \\ q_1 &= \sin\left(\frac{\theta}{2}\right)x_f \\ q_2 &= \sin\left(\frac{\theta}{2}\right)y_f \\ q_3 &= \sin\left(\frac{\theta}{2}\right)z_f \end{aligned} \tag{2.8}$$

dengan vektor f (x_f, y_f, z_f) merupakan sumbu perputaran dan θ merupakan besar sudut putar berlawanan arah dengan jarum jam.

2.4 Unity Game Engine

[11] Unity merupakan salah satu *Game Engine* dalam pembuatan permainan dua dimensi atau tiga dimensi. Unity merupakan *Game Engine multi-platform* sehingga permainan yang dibuat menggunakan Unity akan dapat berjalan diberbagai macam perangkat seperti *smartphone*, komputer, *console*, dan lain-lain. *Game Engine* ini menggunakan bahasa C# atau Javascript.

Game Engine ini dapat diperoleh secara gratis, yaitu dengan menggunakan Unity Personal Edition. Game Engine edisi Personal ini tidak akan berlaku jika penggunaan unity mendapatkan pendapatan kotor tahunan sebesar USD 100.000. Jika pendapatan kotor tahunan sudah melebihi USD 100.000, pengguna unity personal harus segera mengganti edisi Unity menjadi Unity Plus, Unity Pro, atau Unity Enterprise. Jika menggunakan Unity Personal, *splashscreen* dan logo *icon* pada permainan yang dibuat tidak diganti maupun dihapus.

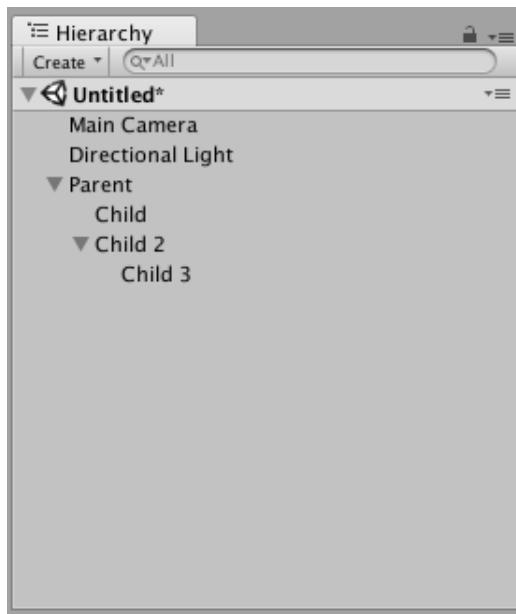
2.4.1 Struktur Hierarki GameObject

[12] Hierarki pada unity merupakan kumpulan dari GameObject. GameObject akan dijelaskan lebih lanjut pada subbab 2.4.4. Beberapa dari GameObject pada hierarki ini merupakan instansi dari file *assets*. Setiap *scene* akan memiliki hierarki masing-masing. Setiap objek yang ditambahkan pada suatu *scene*(dijelaskan pada subbab 2.4.3) akan muncul pada hierarki juga.

Pada hierarki ini juga ada konsep *parent-child* objects. Konsep ini digunakan untuk setiap kumpulan object. Setiap objek yang berada paling luar dinamakan "parent object", dan objek-objek yang berada didalamnya dinamakan "child object". Suatu parent object juga dapat memiliki parent objek(biasa disebut jika "nested parent object"). Contoh dari konsep *parent-child* ada pada Gambar 2.9.

2.4.2 Prefabs

Unity memiliki suatu jenis assets yang dinamakan Prefab. Prefab ini adalah suatu GameObject yang disimpan menjadi assets. Prefab berperan sebagai contoh/blueprint dari suatu GameObject.



Gambar 2.9: Contoh Hierarchy Parenting.

Prefab ini juga dapat dibentuk dari GameObject yang telah dibuat dan menjadi assets, sehingga dapat digunakan pada proyek lain. Prefab juga data di ubah dan dimodif sehingga menjadi sesuai dengan yang diinginkan.

Dalam membuat Prefab dapat dengan mudah *me-drag and drop* suatu GameObject pada tab Scene ke tab Assets. Begitupula dengan sebaliknya, untuk menggunakan suatu prefab, dapat dengan mudah *me-drag and drop* dari tab Assets ke tab Scene.

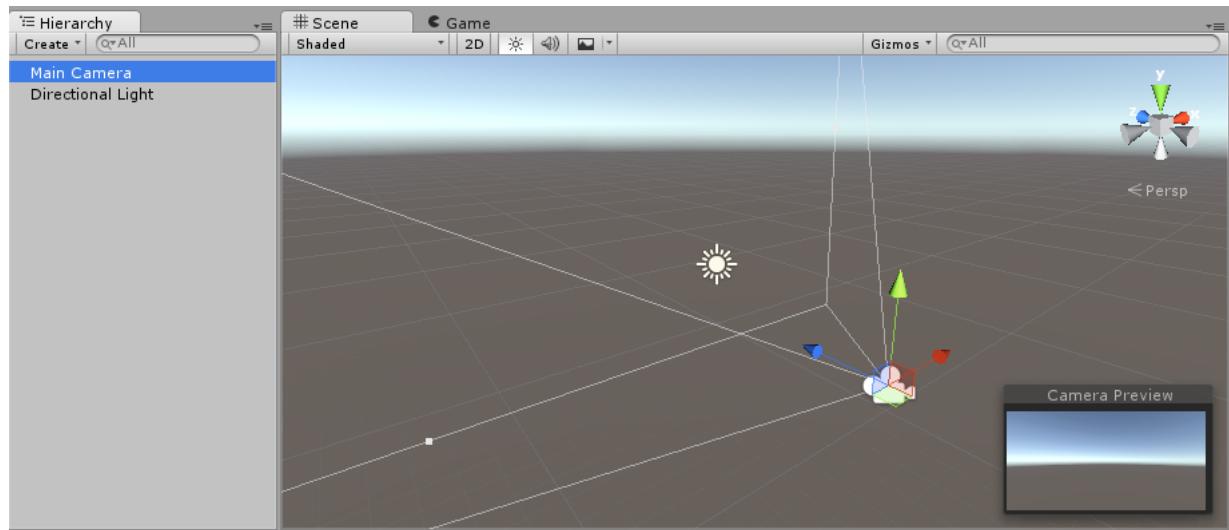
2.4.3 Scene

Scene menyimpan seluruh objek pada game yang akan dibuat. *Scene* dapat digunakan untuk membuat *main menu*, *game level*, dan lain sebagainya. Untuk setiap *scene*, akan ditempatkan barang, bangunan, dekorasi, dan lain sebagainya untuk merancang dan membangun suatu permainan bagian per bagian.

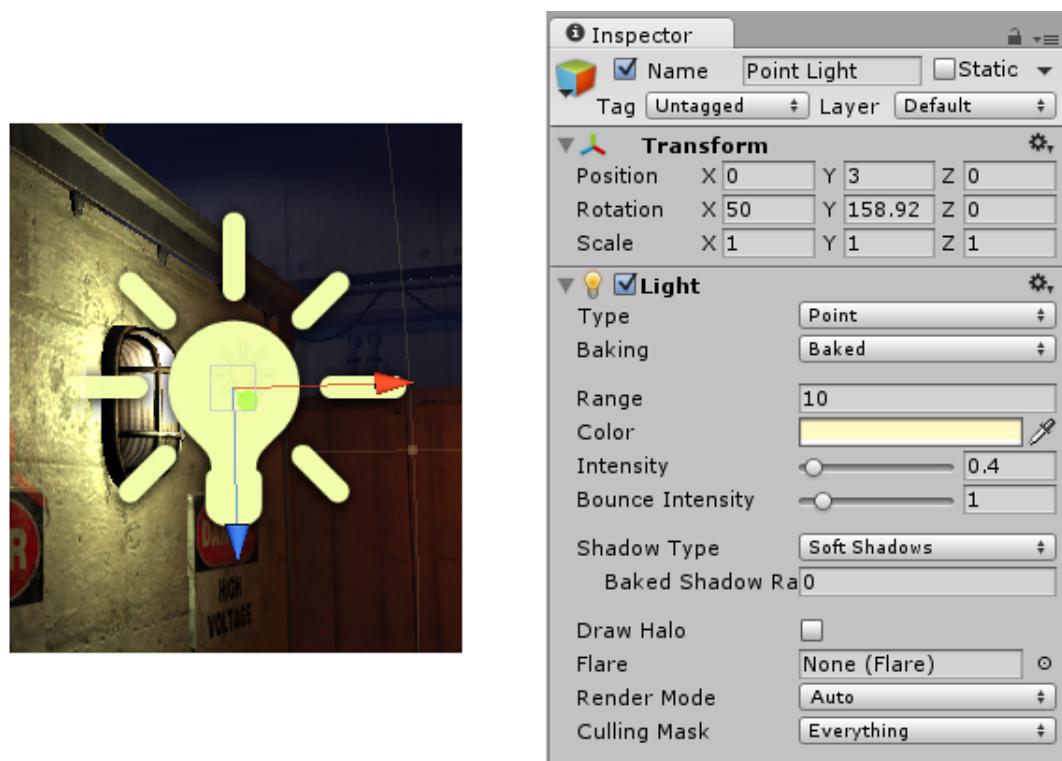
Pada saat pembuatan proyek pertama, unity akan membuatkan suatu *scene* baru. *Scene* tersebut merupakan *scene default*. *Scene* tersebut hanya akan diberikan objek-objek *default* saja seperti kamera dan cahaya. Kamera yang akan diberikan antara dua buah jenis yaitu *orthographic camera* atau *perspective camera*, proyek 2D akan diberikan *orthographic camera* dan proyek 3D akan diberikan *perspective camera*. Contoh dari *scene default* pada proyek 3D ditunjukkan pada Gambar 2.10.

2.4.4 GameObject

GameObject pada unity yang merupakan representasi karakter, barang, dan pemandangan. GameObject tidak dapat melakukan apa pun dengan sendirinya, tetapi GameObject melakukan sesuatu sesuai dengan komponen yang ada pada GameObject. komponen mengimplementasikan fungsi-fungsi nyata pada GameObject. Contohnya objek cahaya dapat terbuat dengan menambahkan komponen Light kepada suatu GameObject (Gambar 2.11).

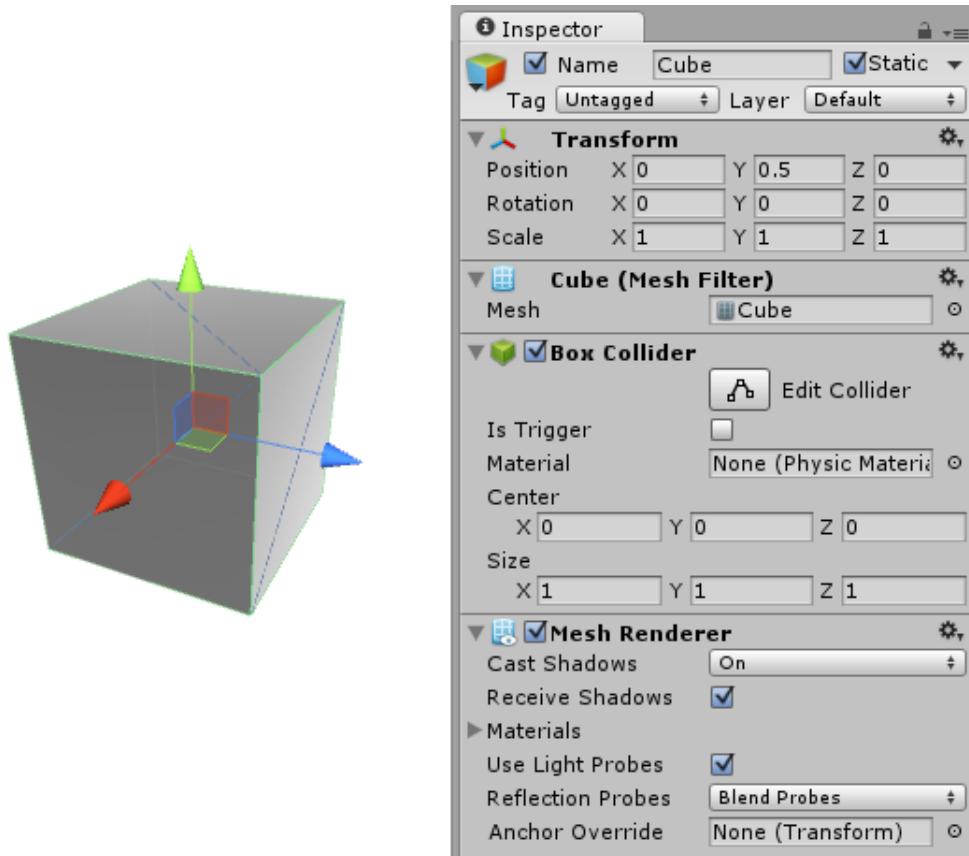


Gambar 2.10: Contoh *scene* kosong.



Gambar 2.11: Contoh penggunaan komponen pada GameObject.

Selain itu ada pula contoh GameObject Kubus. GameObject ini memiliki komponen Cube(Mesh Filter) dan Mesh Renderer. Kedua komponen tersebut berguna untuk menggambar permukaan dari kubus tersebut. Selain itu ada juga komponen Box Collider yang digunakan untuk merepresentasikan *GameObject* tersebut dalam hal-hal fisika seperti gravitasi, gaya, dan lain sebagainya(Gambar 2.12).



Gambar 2.12: Contoh komponen pada GameObject kubus.

2.4.5 Transformasi Rotasi dan Orientasi

Setiap GameObject selalu mempunyai komponen Transform untuk merepresentasikan posisi, orientasi, dan skala GameObject tersebut. Komponen ini tidak dapat dihapus. Komponen Transform ini nilainya relatif terhadap nilai komponen Transform pada *parent*-nya. Jika Transform ini tidak memiliki *parent*, nilainya akan relatif pada dunianya (*world space*).

Komponen Transform memiliki beberapa atribut. Atribut-atribut yang penting pada komponen ini adalah position, lossyScale, dan rotation. Atribut position menyimpan posisi dari suatu GameObject. Atribut ini disimpan dengan menggunakan kelas Vector3. Atribut lossyScale menyimpan besar penskalaan suatu GameObject. Atribut lossyScale ini pun disimpan dengan menggunakan kelas Vector3. Atribut rotation digunakan untuk menyimpan orientasi dari suatu GameObject. Atribut rotation berbeda dengan atribut position dan lossyScale. Atribut rotation disimpan dengan menggunakan kelas Quaternion.

Rotasi pada aplikasi tiga dimensi biasanya menggunakan antara metode Quaternion atau Euler Angles. Masing-masing memiliki kelebihan dan kekurangan. Seperti yang tadi sudah dijelaskan,

Unity menggunakan Quaternion dalam merepresentasikan rotasi dan orientasi. Pada kelas Quaternion terdapat atribut yang menyimpan nilai-nilai orientasi pada Euler Angles, sehingga juga unity dapat merepresentasikan rotasi dan orientasi dengan menggunakan Euler Angles.

Kelebihan menggunakan representasi Euler Angles.

- Euler Angles lebih mudah untuk di pahami dalam format tiga sudut pada sumbu-sumbu tiga dimensi.
- Euler Angles dapat merepresentasikan rotasi dari satu orientasi ke orientasi dengan sumbu yang lebih besar dari 180 derajat.

Kekurangan menggunakan representasi Euler Angles.

- Euler Angles dapat mengalami Gimbal Lock. Gimbal Lock adalah kejadian ketika dua sumbu dari sudut perputaran pada Euler Angles berputar pada poros yang sama.

Kelebihan menggunakan representasi Quaternion.

- Kuaternion tidak mengalami Gimbal Lock.

Kekurangan menggunakan representasi Quaternion.

- suatu kuaternion tidak dapat merepresentasikan rotasi yang melebihi 180 derajat pada arah manapun.
- Representasi angka-angka pada Quaternion lebih susah untuk dimengerti.

Pada Unity, meski pun semua orientasi disimpan dengan menggunakan Quaternion, tetapi pada *Transform Inspector* (Gambar ??) Unity menampilkan nilai-nilai orientasi pada Euler Angles. Hal ini bertujuan untuk mempermudah mengubah orientasi bagi pengguna, karena nilai-nilainya lebih mudah untuk dimengerti. Representasi pada *Transform Inspector* tersebut menyebabkan masukkan yang diberikan oleh pengguna terkadang berbeda dengan data yang disimpan. Contohnya ketika berada pada sudut 365 derajat, maka data yang akan disimpan akan menunjukkan perputaran sebesar 5 derajat saja.

2.5 Google VR SDK for Unity

Google juga menyediakan SDK untuk *Game Engine* Unity. Berbeda dengan Google Cardboard API, Google VR SDK for Unity tidak hanya berfungsi untuk membuat aplikasi untuk Google Cardboard. Google VR SDK juga berfungsi untuk membuat aplikasi untuk Google Daydream, yaitu Aplikasi VR dari Google sebagai pembaruan Google Cardboard. Google daydream memiliki kelebihan memiliki *remote controller* yang dapat merekam gerakan tangan pengguna. Jadi jika menggunakan Google VR SDK for Unity, pembuat permainan juga dapat membuat pengaturan untuk *remote controller* pada Google Daydream.

Untuk mendapatkan Google VR SDK for Unity, SDK tersebut dapat di download pada halaman situs web Google VR Developers [9]. Pada situs tersebut akan diberikan assets package dengan ekstensi file ".unitypackage". Assets package tersebut akan memasukkan seluruh kebutuhan untuk membuat Google VR pada proyek permainan yang sedang dikerjakan.

Untuk dapat me-*render* permainan dengan tampilan *stereo rendering* (tampilan layar yang dibagi menjadi dua bagian sesuai dengan mata manusia), hierarki GameObject pada scene yang sedang dikerjakan harus memiliki GvrViewerMain dengan komponen script GvrViewer. GameObject ini dapat diperoleh dengan menggunakan *prefabs* yang telah di-*import*. Selain untuk me-*render*, permainan object tersebut juga mengimplementasikan orientasi pada *smartphone* secara langsung.

Seperi yang sudah dijelaskan Google Cardboard memberikan masukkan dengan menarik/menelek pelatuk pada Google Cardboard tersebut. Untuk mendapatkan masukkan tersebut, dapat dilakukan dengan cara menggunakan *prefab* GvrEventSystem. Kemudian menambahkan komponen Event Trigger pada GameObject yang akan merespons masukkan tersebut. Respons dapat dilakukan dengan menambahkan Script baru pada GameObject tersebut.

BAB 3

ANALISIS

Pada bab ini akan dijelaskan mengenai analisis grafik dari data sensor-sensor pada *smartphone* ketika sedang mengangguk dan menggeleng, analisis aplikasi-aplikasi sejenis, dan analisis metode pendekripsi gerakan kepala.

3.1 Analisis Aplikasi Sejenis

Aplikasi sejenis pada perangkat *Virtual Reality* Google Cardboard hanyalah ada satu aplikasi saja. Aplikasi tersebut adalah aplikasi InMind VR. Aplikasi sejenis pada perangkat *Virtual Reality* Oculust Rift adalah Trial of the Rift Drifter dengan Asunder yang dikembangkan oleh AldinDynamics [13]. Aplikasi sejenis pada *Virtual Reality* Oculust Rift tidak dapat dianalisis karena penulis tidak memiliki perangkat Oculust Rift.

Analisis InMind VR

InMind VR merupakan permainan *Virtual Reality* yang seakan-akan pengguna berada dalam sel-sel otak seorang manusia. Permainan ini dimainkan dengan mengarahkan arah pandang kepala ke arah sel-sel neuron yang dapat menyebabkan gangguan mental (Gambar 3.1). Akan ada 50 sel neuron yang dapat menyebabkan gangguan mental. Sel-sel yang dapat menyebabkan gangguan mental adalah sel-sel yang berwarna merah seperti pada Gambar 3.2.

Pada permulaan permainan pengguna diberi pertanyaan apakah pengguna sudah siap untuk melakukan permainan tersebut seperti yang ditunjukkan pada Gambar 3.3.

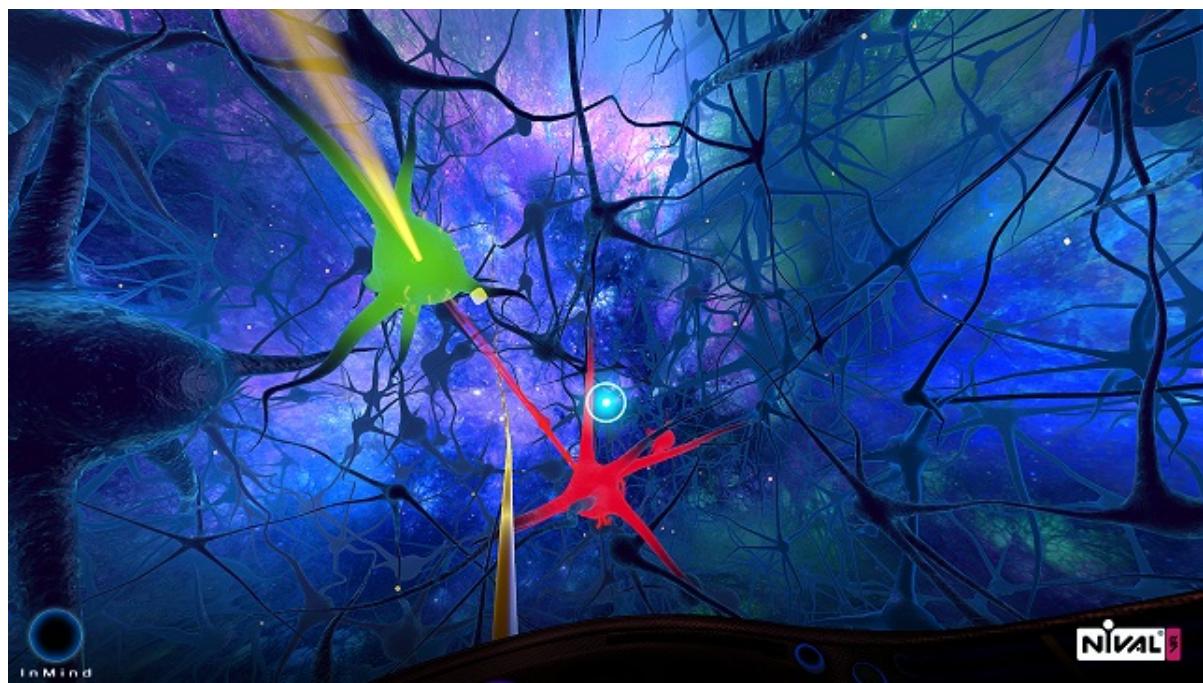
Analisis dilakukan dengan cara mengetes aplikasi ini dengan mengangguk dan menggeleng yang spesifik. Mengangguk dilakukan dengan cara-cara berikut:

1. Mengangguk secara pelan.
2. Mengangguk yang diawali dengan gerakan ke atas.
3. Melihat kebawah saja tanpa membalikkan kepala ke posisi semula.
4. Tidak menggerakkan kepala sama sekali.

Dari keempat percobaan mengangguk yang dilakukan, hanya percobaan pertama dengan percobaan ketiga yang terdeteksi mengangguk. Aplikasi ini dapat disimpulkan dengan hanya menggerakkan kepala ke bawah saja sudah dapat dianggap mengangguk oleh aplikasi ini. Pada percobaan



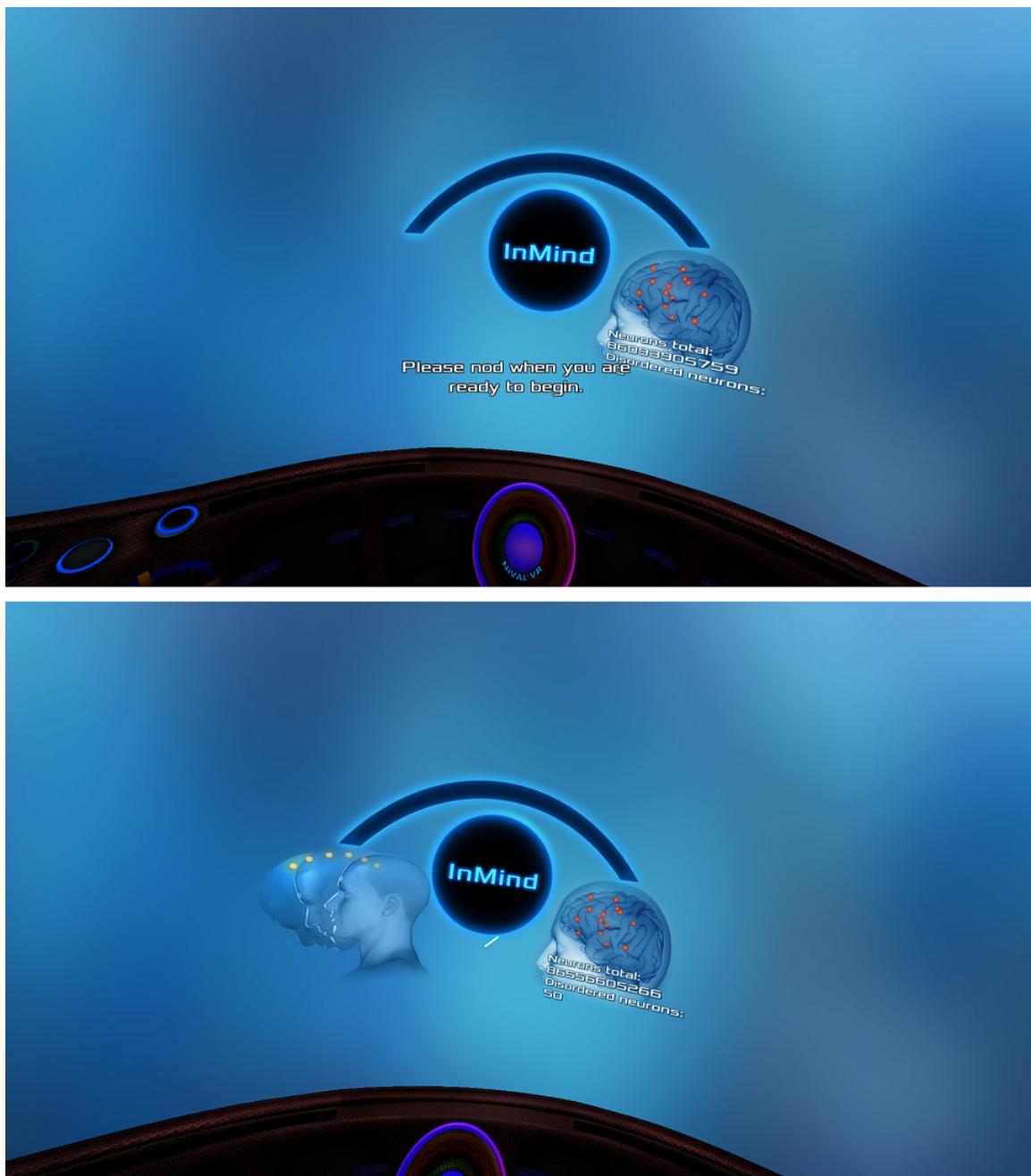
Gambar 3.1: *Screenshot* task yang diberikan aplikasi InMind VR.



Gambar 3.2: *Screenshot* aplikasi InMind VR ketika permainan berlangsung.

kedua dan keempat terjadi keganjilan dari pendekripsi anggukan. Pada percobaan kedua dan keempat aplikasi tetap akan melanjutkan permainan setelah beberapa detik berlalu. Sel-sel yang dapat menyebabkan gangguan mental adalah sel-sel yang berwarna merah seperti pada Gambar 3.2.

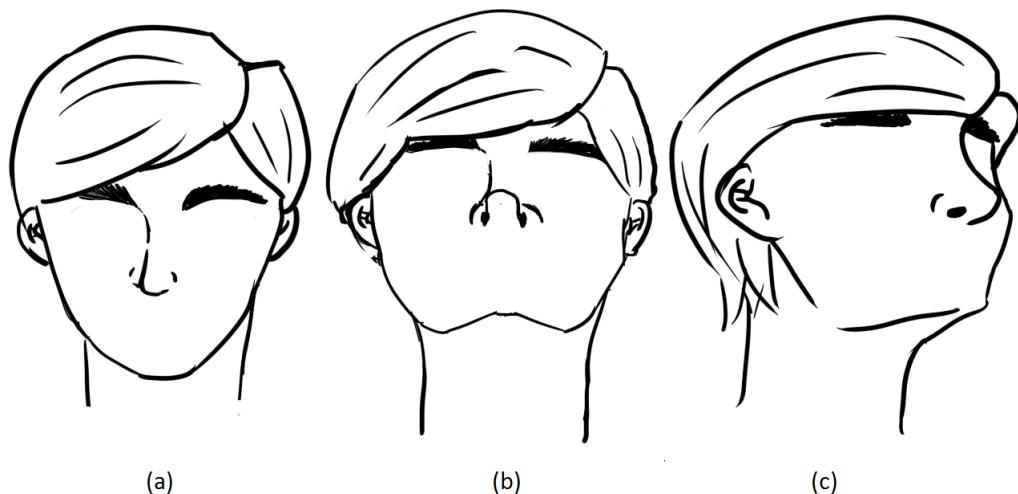
Ketika dilakukan percobaan menggeleng, tidak ada respon apapun dari aplikasi ini. Aplikasi ini akan tetap melanjutkan ke permainan setelah beberapa detik telah berlalu. Hal tersebut menyimpulkan bahwa aplikasi ini tidak dapat mendekripsi gerakan menggeleng. Oleh karena itu hal yang dilakukan oleh aplikasi ini hanyalah melihat apakah pengguna sudah melihat ke bawah atau belum saja.



Gambar 3.3: *Screenshot* aplikasi InMind VR ketika meminta pengguna untuk mengangguk jika telah siap.

3.2 Perekaman Data Sensor

Pada analisis ini akan dilakukan perekaman mengangguk dan menggeleng dengan sensor-sensor pada Android. Perekaman-perekaman ini akan dilakukan pada tiga kondisi muka pengguna. Kondisi muka yang pertama adalah kondisi muka pengguna ketika menghadap ke depan, digambarkan dengan Gambar 3.4 bagian (a). Kondisi muka yang kedua adalah kondisi muka pengguna ketika menghadap ke atas sekitar 45° dari pandangan muka menghadap ke atas digambarkan dengan Gambar 3.4 bagian (b). Kondisi muka yang ketiga adalah kondisi muka ketika menghadap serong ke kiri atas digambarkan dengan Gambar 3.4 bagian (c). Anggukan yang dilakukan oleh pengguna hanya sebanyak satu kali mengangguk ke bawah saja. Sedangkan dalam menggeleng akan bergerak ke kiri terlebih dahulu dan ke kanan setelahnya dan diakhiri pada posisi muka kembali ke posisi awal.



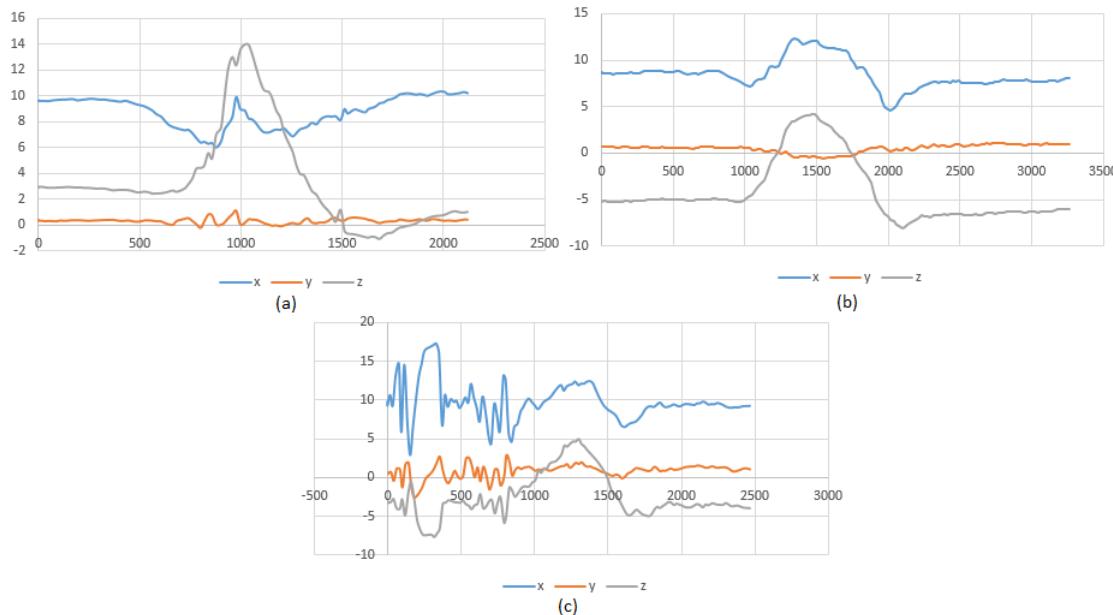
Gambar 3.4: Gambar untuk mendeskripsikan posisi dari muka sebelum melakukan gerakan menggeleng atau mengangguk. Gambar (a) adalah kondisi muka ketika sedang menghadap ke depan. Gambar (b) adalah kondisi muka ketika sedang menghadap ke atas. Gambar(c) adalah kondisi muka ketika sedang menghadap ke serong kiri atas.

Grafik-grafik yang akan ditunjukkan pada bab ini akan memiliki beberapa karakteristik. Sumbu y pada grafik yang akan ditunjukkan akan merepresentasikan besar nilainya, Sedangkan sumbu x akan merepresentasikan waktunya. Grafik yang ditunjukkan akan memiliki beberapa nilai, tergantung dari jumlah nilai yang dikembalikan untuk setiap sensornya. Contohnya pada sensor *accelerometer* yang memiliki tiga jenis nilai, sehingga pada grafik akan terbentuk tiga buah garis nilai. Aplikasi akan merekam beberapa sensor secara langsung ketika pengguna mengangguk atau menggeleng, sehingga nilai waktunya akan sama untuk kondisi muka yang sama walaupun sensornya berbeda.

Analisis grafik data sensor-sensor pada Android dilakukan dengan membuat suatu aplikasi perrekam sensor-sensor yang ada pada *smartphone* Android terlebih dahulu. Aplikasi ini akan merekam nilai-nilai yang dihasilkan dari sebagian sensor-sensor pada android setiap ada perubahan. Pada skripsi ini, nilai sensor-sensor yang dibutuhkan adalah sensor *accelerometer*, *gyroscope*, *rotation vector*, dan *geomagnetic rotation*. Aplikasi menyimpan nilai sensor-sensor menggunakan format CSV (*Comma Separated Values*). Dari data yang diperoleh oleh aplikasi tersebut dibuatkan grafiknya menggunakan aplikasi Microsoft Excel. Penjelasan dari setiap grafik akan dijelaskan pada subbab-subbab berikut.

3.2.1 Perekaman Grafik Sensor *Accelerometer*

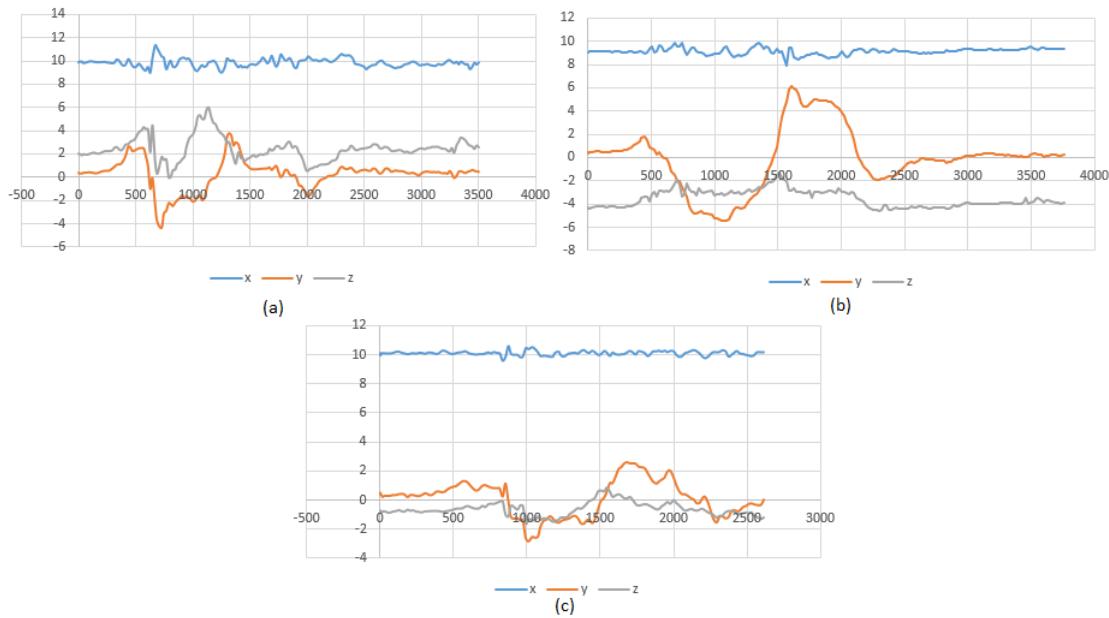
Seperti yang sudah dijelaskan pada bab sebelumnya, sensor *accelerometer* akan mendeteksi seluruh percepatan yang terjadi pada perangkat Android. Perekaman ini perangkat android akan diletakkan di depan muka pengguna, sehingga percepatan yang memengaruhi perangkat Android hanya percepatan gravitasi dengan percepatan yang dilakukan oleh gerakan kepala pengguna.



Gambar 3.5: Grafik nilai kuaternion dari sensor *accelerometer* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

Gambar 3.5 merupakan grafik-grafik yang terbentuk dari nilai yang diterima oleh sensor *accelerometer* ketika pengguna sedang mengangguk. Pada Gambar 3.5 grafik (a) terlihat nilai z menaik dan nilai x menurun ketika sedang mengangguk. Tetapi nilai x kembali menaik ketika nilai z sudah hampir mencapai nilai tertinggi. Sedangkan nilai y terlihat cukup konstan di sekitar angka 0. Pada grafik (b) terlihat nilai x dengan y memiliki pola yang serupa ketika mengangguk. Nilai x berada pada nilai sekitar sebesar 9 sedangkan nilai z bernilai sekitar sebesar -5. Sama seperti pada grafik (a) nilai y terlihat konstan di sekitar angka 0. Selanjutnya grafik (c) pada Gambar 3.5 merupakan grafik yang terbentuk dari nilai yang diterima oleh sensor *accelerometer* ketika pengguna mengangguk dan sedang menghadap ke kiri atas. Grafik pada bagian ini sangat tidak beraturan. Pada Grafik ini sulit untuk membedakan kondisi kapan pengguna sedang mengangguk. Goncangan yang terjadi terhadap *smartphone*, seperti munculnya notifikasi mungkin dapat menyebabkan hal ini. Namun pada waktu mencapai 1000 milidetik terlihat cukup stabil. Pada grafik (c) di Gambar 3.5 juga menunjukkan bahwa nilai x dengan z memiliki pola yang sama hingga akhir, dan nilai y konstan di sekitar angka 0. Hasil nilai tersebut serupa dengan kasus ketika menghadap ke atas.

Gambar 3.6 merupakan grafik yang terbentuk dari nilai yang diterima oleh sensor *accelerometer* ketika pengguna menggeleng. Pada grafik (a), nilai x konstan di sekitar angka 10. Nilai y dengan z pada grafik (a) menaik dan menurun ketika pengguna menggelengkan kepala. Pada grafik (b) nilai



Gambar 3.6: Grafik nilai kuaternion dari sensor *accelerometer* ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

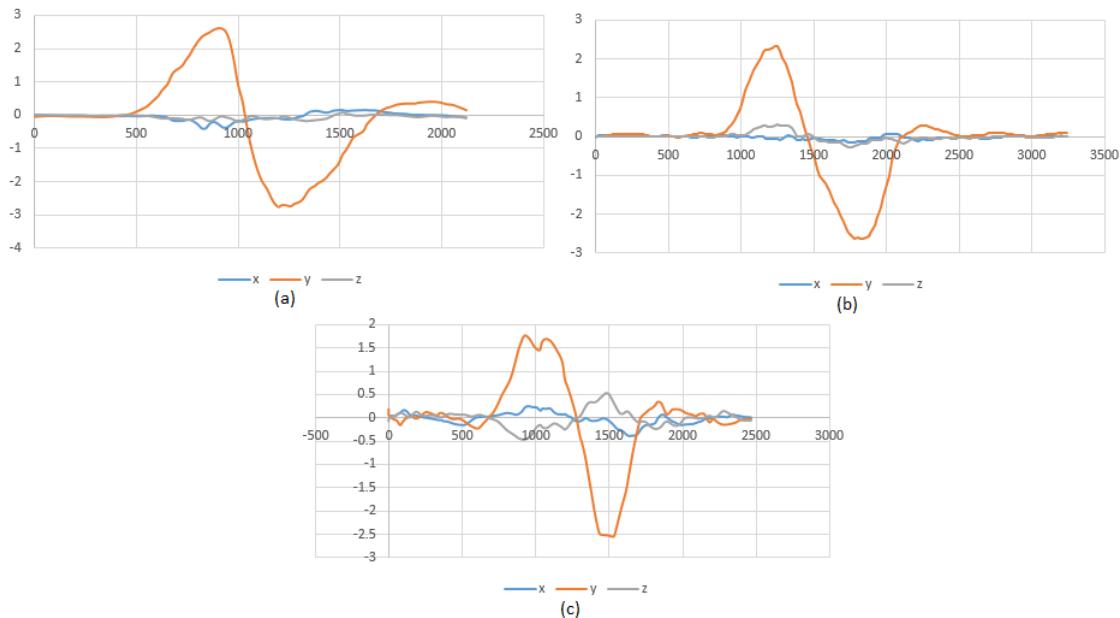
x terlihat konstan di sekitar nilai 10 dan nilai z sedikit tidak beraturan di sekitar nilai -4 hingga -2. Nilai y mengalami kenaikan dan penurunan saat menggeleng. Pada grafik (c) di Gambar 3.6 nilai x konstan di sekitar nilai 10. Nilai y menaik dan menurun, tetapi tidak beraturan. Nilai pada z juga mengalami sedikit kenaikan dan penurunan. Pada grafik (b) dengan (c) nilai z mengalami perubahan yang tidak beraturan dan puncak tertinggi dengan puncak terendahnya tidak terlalu berbeda jauh dengan nilai awalnya.

Dari keenam grafik tersebut terlihat bahwa nilai yang terpengaruh ketika pengguna sedang mengangguk adalah nilai x dengan z. Nilai y tidak berpengaruh karena nilai y cenderung konstan. Nilai yang terpengaruh ketika pengguna sedang menggeleng adalah nilai y. Nilai x terlihat konstan pada setiap grafik, tetapi nilai z mengalami sedikit pergerakan ketika pengguna sedang mengangguk sehingga tidak dapat dipastikan bahwa nilai z terpengaruh gerakan menggeleng.

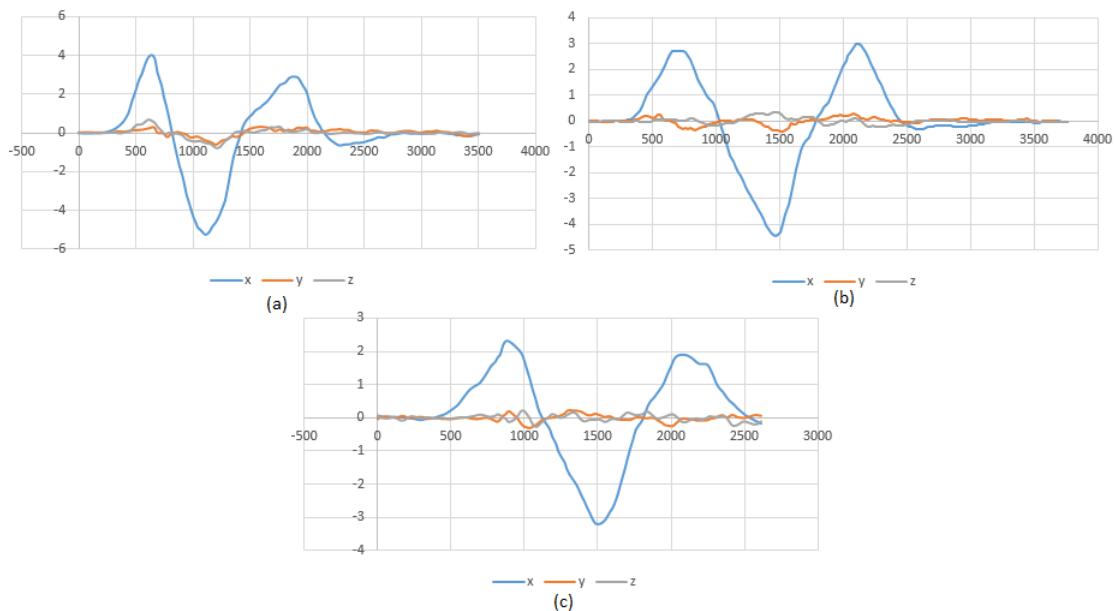
3.2.2 Perekaman Grafik Sensor *Gyroscope*

Perekaman menggunakan sensor *gyroscope* akan mendapatkan percepatan angular yang terjadi setiap waktunya. Berbeda dengan sensor *accelerometer* yang dapat terpengaruh oleh lingkungan sekitar seperti gravitasi dan percepatan lainnya, sensor ini hanya akan merekam perputaran yang terjadi pada perangkat saja. Hal ini sangat menguntungkan dalam mendeteksi suatu gerakan karena tidak harus memedulikan kasus dari pengaruh luar.

Gambar 3.7 menunjukkan nilai-nilai sensor *gyroscope* ketika pengguna sedang mengangguk. Grafik (a) membentuk sebuah bukit dan lembah pada nilai y. Bukit yang terjadi menunjukkan ketika pengguna menggerakkan kepala ke bawah, dan ketika kepala pengguna kembali ke posisi semula percepatan angularnya berbalik arah sehingga menimbulkan lembah. Nilai x dengan z cenderung bernilai 0. Grafik (b) mirip seperti grafik pada Gambar 3.7. Begitu pula pada grafik (c) yang memiliki pola yang serupa dengan yang grafik-grafik sebelumnya.



Gambar 3.7: Gambar grafik nilai sensor *gyroscope* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.



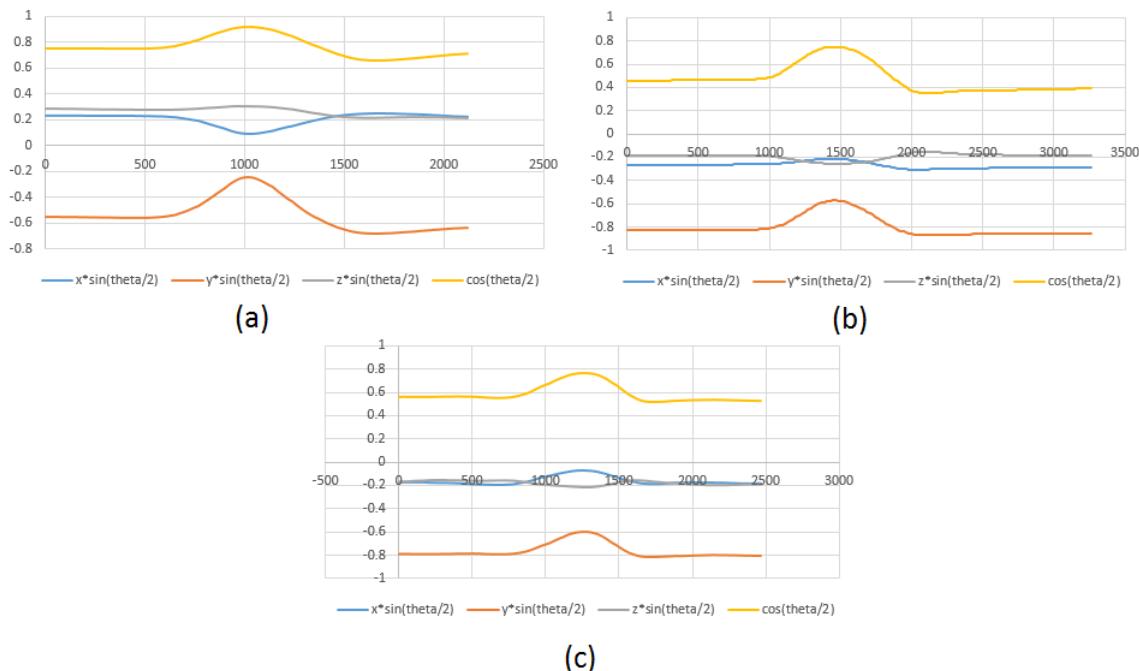
Gambar 3.8: Gambar grafik nilai sensor *gyroscope* ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

Gambar 3.8 menunjukkan nilai-nilai sensor *gyroscope* ketika pengguna sedang menggeleng. Pada grafik (a) nilai yang mengalami kenaikan dan penurunan adalah nilai x dan nilai-nilai lainnya cenderung berada pada nilai 0. Nilai x membentuk 2 buah bukit dan 1 buah lembah. Bukit pertama terjadi ketika pengguna menggerakkan kepalamanya ke kiri. Lembah pertama terjadi ketika pengguna menggerakkan kepalamanya ke kanan. Bukit kedua terjadi ketika pengguna menggerakkan kepalamanya kembali ke posisi semula. Pada grafik (b) dengan grafik (c) menunjukkan pola grafik yang serupa dengan grafik pada Gambar (a).

Dari hasil-hasil tersebut dapat disimpulkan bahwa arah pandang pengguna tidak memengaruhi sensor *gyroscope* dalam mendeteksi gerakan kepala. Nilai-nilai yang dikembalikan oleh sensor *gyroscope* memiliki pola grafik yang jauh lebih rapi dibandingkan grafik-grafik yang dihasilkan oleh sensor *accelerometer*. Selain itu sensor *gyroscope* hanya menggunakan 1 jenis nilai yang dipengaruhi oleh pergerakan kepala, sedangkan *accelerometer* ada 2 jenis nilai yang dipengaruhi gerakan kepala pada saat mengangguk. Oleh karena itu sensor *gyroscope* lebih baik dalam mendeteksi gerakan yang terjadi pada perangkat android.

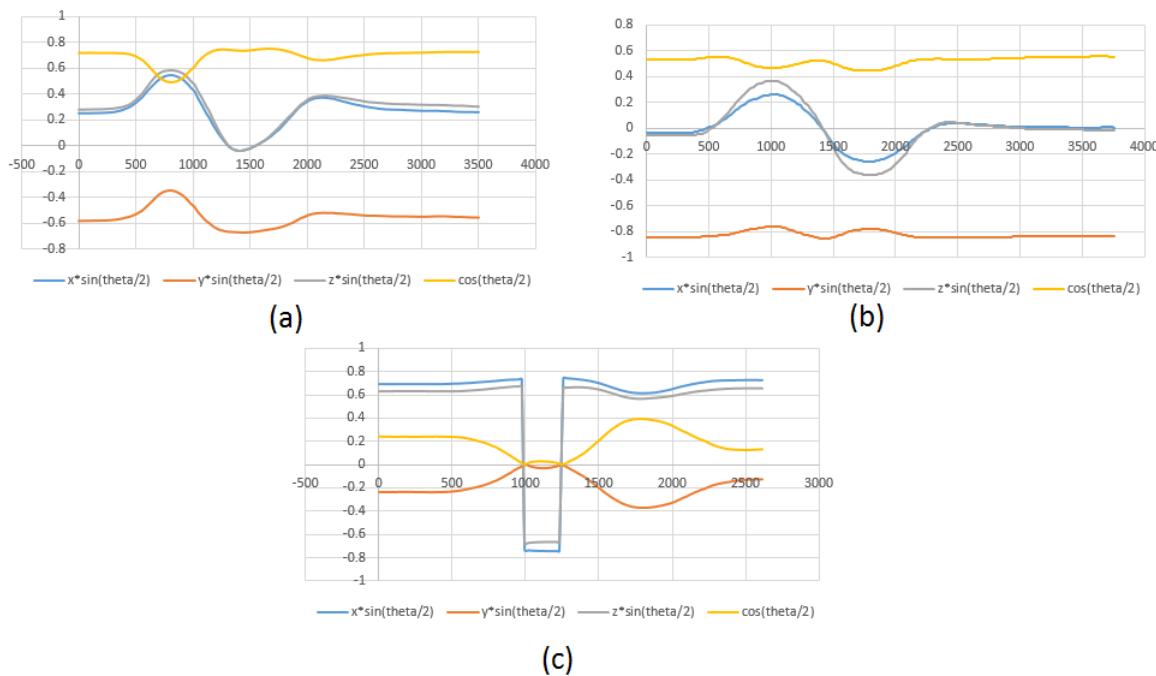
3.2.3 Perekaman Grafik Sensor *Rotation Vector*

Perekaman menggunakan sensor *rotation vector* akan mendapatkan sebuah kuaternion yang merepresentasikan perputaran yang terjadi pada perangkat Android. Perputaran ini akan dideskripsikan dengan suatu vektor sebagai sumbu putarnya dan sudut perputarannya. Berbeda dengan sensor *gyroscope* yang merekam kecepatan perputaran yang terjadi pada suatu waktu, sensor *rotation vector* akan mengembalikan nilai kuaternion untuk mendefinisikan suatu kondisi putaran pada saat itu.



Gambar 3.9: Grafik nilai kuaternion dari sensor *rotation vector* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

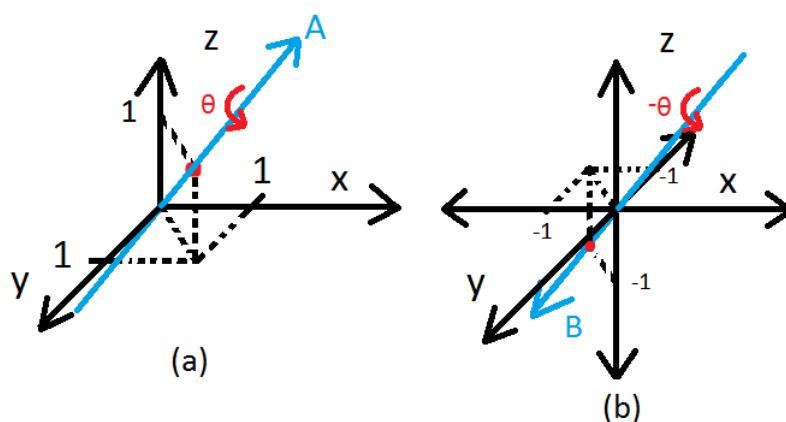
Gambar 3.9 menunjukkan nilai-nilai sensor *rotation vector* ketika pengguna sedang mengangguk. Pada grafik (a) terbentuk sebuah bukit pada garis berwarna jingga dengan kuning, dan lembah pada garis berwarna biru. Garis yang berwarna abu cenderung stabil di angka 0.3. Grafik (b) menunjukkan pola yang mirip pada bagian (a) namun berbeda nilainya saja. Pada grafik (a) garis berwarna kuning dimulai pada angka sekitar 0.7, sedangkan pada grafik (b) garis berwarna jingga dimulai pada angka sekitar 0.4. Garis biru pada grafik ini tidak membentuk sebuah lembah seperti pada grafik pada grafik (a). Garis berwarna abu cenderung konstan pada nilai -0.2, sedangkan pada grafik (a) cenderung konstan di sekitar 0.3. Garis berwarna jingga memiliki pola yang sama dengan garis berwarna kuning, hanya berbeda pada nilainya saja. Seperti pada grafik-grafik sebelumnya, grafik (c) ini memiliki pola yang sama dengan grafik lainnya. Grafik ini juga hanya nilainya saja yang berbeda dengan grafik lainnya. Kemiripan pola ini memungkinkan mempermudah pendekripsi gerakan kepala.



Gambar 3.10: Grafik nilai kuaternion dari sensor *rotation vector* ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

Gambar 3.10 menunjukkan nilai-nilai sensor *rotation vector* ketika pengguna sedang menggeleng. Pada grafik (a) terbentuk sebuah bukit yang diikuti dengan lembah pada garis berwarna biru dengan abu-abu. Garis berwarna kuning membentuk suatu lembah yang setelahnya cenderung konstan. Berbeda pada garis kuning yang membentuk bukit kemudian setelahnya cenderung konstan. Grafik (b) memiliki kemiripan dengan grafik (a), garis biru dengan garis abu-abu memiliki pola yang sama yaitu membentuk bukit dengan lembah ketika pengguna menggelengkan kepala. Garis kuning dengan jingga cenderung konstan, berbeda dengan grafik (a) yang membentuk bukit atau lembah. Pada grafik (c) garis-garis membentuk suatu pola yang tidak normal. Garis kuning membentuk sebuah lembah, tetapi membentuk suatu bukit kecil ketika nilainya mencapai nilai 0 pada milidetik ke 1000. Garis jingga memiliki pola yang berlawanan dengan garis kuning. Pada saat garis kuning dengan garis jingga mencapai angka 0 perubahan drastis pun terjadi pada garis biru dengan abu-abu.

Pada milidetik ke 1000 garis biru dengan abu mengalami perubahan nilai yang sangat drastis. Kedua nilai tersebut berubah dari nilai yang berkisar diantara 0.6 sampai 0.7 menjadi berkisar diantara -0.7 sampai -0.8. Kasus ini tidak berarti sensor sedang mengalami kegagalan akurasi (*accuracy fail*), tetapi memang seperti itulah karakteristik dari perputaran menggunakan kuaternion pada android. Perputaran yang terjadi pada batas mendekati sebelum terjadinya dengan setelah perubahan nilai yang drastis memiliki hasil perputaran yang sama. Hal ini disebabkan karena melakukan perputaran dengan vektor sebagai sumbu dapat memiliki 2 buah nilai yang sejenis. Dua buah nilai tersebut akan menghasilkan perputaran yang sama ketika arah vektor dengan arah putarnya di balikkan seperti yang ditunjukkan pada Gambar 3.11. Sepertinya pada sistem android nilai $\cos(\theta/2)$ didesain agar tidak bernilai negatif, sehingga nilai-nilai yang lainnya akan mengalami perubahan nilai yang drastis ketika nilai $\cos(\theta/2)$ mencapai angka 0.



Gambar 3.11: Dua buah rotasi yang identik dengan nilai kuaternion yang berbeda.

3.3 Analisis Data Sensor untuk Mendeteksi Gerakan Kepala

Dari ketiga hasil pecobaan pada sensor-sensor pada bab 3.2 dapat disimpulkan bahwa sensor *gyroscope* adalah sensor yang terbaik untuk mendeteksi gerakan kepala. Data dari sensor *accelerometer* akan susah untuk digunakan dalam mendeteksi gerakan kepala karena terganggu dengan aktivitas-aktivitas diluar gerakan pengguna yang juga ikut terekam oleh sensor *accelerometer*. Data dari sensor *rotation vector* juga akan lebih rumit dibandingkan sensor *gyroscope*. Hal ini karena perputaran yang di rekam oleh sensor *rotation vector* merekam kondisi putar pada suatu saat. Hasil rekaman ini akan mempersulit pada saat pendekripsi gerakan kepala karena harus melakukan proses untuk menghitung kecepatan kepala bergerak, agar dapat membedakan gerakan mengangguk atau menggeleng atau sekedar menoleh biasa. Dalam mendeteksi gerakan mengangguk dengan menggeleng banyak batas-batas yang perlu diperhatikan. Batas-batas tersebut untuk mengetahui apakah pengguna benar-benar mengangguk atau menggeleng atau sekedar menoleh biasa. Batas-batas yang perlu diperhatikan adalah:

- Kecepatan pengguna menoleh.
- Jarak waktu pada saat pengguna melakukan melawan arah gerakan.

- Simpangan terbesar kepala saat mengangguk atau menggeleng.

Kecepatan pengguna menjadi batas karena gerakan kepala yang kecepatannya cenderung pelan biasanya bukan merupakan gerakan mengangguk ataupun menggeleng. Kecepatan ini dapat langsung diperoleh menggunakan sensor *gyroscope*. Kecepatan yang dibutuhkan adalah kecepatan perputaran maksimum yang dilakukan oleh pengguna. Kecepatan maksimum dapat diperoleh dengan mengambil nilai puncak tertinggi dengan terendah. Jika nilai puncaknya mencapai kecepatan tertentu, gerakan tersebut dapat diperkirakan merupakan gerakan mengangguk ataupun menggeleng.

Gerakan menggeleng atau mengangguk biasanya memiliki selang waktu yang sangat sempit, karena pengguna biasanya langsung melawan arah secara langsung ketika sedang mengangguk ataupun menggeleng. Nilai ini dapat diperoleh dengan menghitung jarak antara bukit dengan lembah yang terbentuk pada grafik seperti yang dijelaskan pada Gambar 3.12. Idealnya gerakan menggeleng tidak memiliki rentang waktu ini, namun mungkin sebagian orang masih menghasilkan rentang waktu yang cukup sedikit. Oleh karena itu mungkin batas waktu yang ditentukan di sini adalah sekitar 100 milidetik. Jika rentang waktunya melebihi batas waktu tersebut, maka gerakan tersebut mungkin bukan merupakan gerakan mengangguk ataupun gerakan menggeleng.



Gambar 3.12: Deskripsi grafik pada saat pengguna menggeleng. Yang ditunjuk oleh panah berwarna hitam adalah selang waktu yang terjadi saat pengguna melawan arah gerakan kepala.

Simpangan terbesar ini juga penting untuk dijadikan batasan-batasan dalam mendeteksi gerakan mengangguk ataupun menggeleng. Simpangan kepala yang sangat kecil dapat diragukan untuk dianggap sebagai gerakan mengangguk atau menggeleng. Simpangan ini dapat diperoleh dengan menghitung luas yang dibentuk dari bukit atau lembah yang terbentuk pada grafik. Contoh pada Gambar 3.12 simpangan terbesar yang terjadi ketika pengguna menggerakkan kepalanya pertama kali ke kiri adalah luas pada bidang yang diarsir merah. Simpangan terbesar yang terjadi ketika pengguna menggerakkan kepalanya ke kanan adalah luas bidang yang diarsir berwarna hijau dikurangi dengan luas pada bidang yang diarsir merah. Pengurangan ini dilakukan karena luas bidang yang diarsir berwarna hijau merupakan simpangan yang terjadi setelah kepala sudah menghadap ke kiri. Begitu pula dengan luas bidang yang diarsir berwarna jingga yang akan dikurangi dengan hasil pengurangan luas sebelumnya.

3.4 Analisis Metode Pendeksi Gerakan Kepala

Seperti yang sudah dijelaskan pada bab 2.1.4, data akan didapatkan setiap ada perubahan nilai pada sensor dengan rentang waktu tertentu, bergantung dengan konfigurasinya. Pendeksi ini

membutuhkan data yang *real-time*, sehingga akan lebih baik jika menggunakan konfigurasi kecepatan pengambilan data setiap 20.000 mikrodetik. Penggunaan konfigurasi dengan kecepatan 0.000 mikrodetik tidak terlalu baik, karena akan menggunakan kemampuan processor yang sangat tinggi.

3.4.1 Algoritma Mendeteksi Gerakan Mengangguk

Algoritma akan dipanggil setiap kali ada perubahan nilai dari sensor. Algoritma ini akan menyimpan nilai-nilai batas-batas yang telah didefinisikan, luas yang terbentuk dari lembah dan bukit terakhir, waktu mulai dan berakhirnya suatu bukit atau lembah. Algoritma ini akan dipanggil secara berulang-ulang hingga menghasilkan hasil yang benar. Data akan disimpan pada attribut, agar setiap pemanggilan dapat mendapatkan data dari pemanggilan sebelumnya.

Berikut adalah keterangan dari data-data yang disimpan pada attribut:

- **passLimitUp**, attribut ini akan mencatat apakah pengguna sudah melewati batas kecepatan sudut ketika kepala pengguna bergerak ke atas.
- **passLimitDown**, attribut ini akan mencatat apakah pengguna sudah melewati batas kecepatan sudut ketika kepala pengguna bergerak ke bawah.
- **currentlyLimitUp**, attribut ini akan menunjukkan bahwa pada saat ini gerakan pengguna sedang bergerak ke atas dan melebihi batas kecepatan sudutnya.
- **currentlyLimitDown**, attribut ini akan menunjukkan bahwa pada saat ini gerakan pengguna sedang bergerak ke bawah dan melebihi batas kecepatan sudutnya.
- **angSpeedLimit**, attribut ini akan menyimpan batas kecepatan sudut.
- **lastYAngSpeed**, attribut ini akan memiliki kecepatan sudut Y pada pemanggilan method sebelumnya.
- **lastT**, attribut ini akan memiliki waktu dipanggilnya method sebelumnya.
- **calcArea**, attribut ini akan menyimpan besar luas bukit atau, lembah yang terjadi. Luas bukit akan bernilai positif, dan nilai lembah akan bernilai negatif.
- **startTValley**, attribut ini akan menyimpan waktu mulai lembah yang memenuhi syarat kecepatan sudut minimal.
- **endTValley**, attribut ini akan menyimpan waktu akhir lembah yang memenuhi syarat kecepatan sudut minimal.
- **startTCurrMotion**, attribut ini akan menyimpan waktu mulai suatu gerakan(bukit atau lembah) yang sedang berlangsung sekarang.
- **endTCurrMotion**, attribut ini akan menyimpan waktu akhir suatu gerakan(bukit atau lembah) yang sedang berlangsung sekarang.
- **deviationLimit** adalah batas simpangan untuk melakukan gerakan mengangguk.

- `deviationMotionDown`, attribut ini akan menyimpan simpangan yang terjadi ketika pengguna menggeraka kepalanya ke bawah.
- `deviationMotioUp`, attribut ini akan menyimpan simpangan yang terjadi ketika pengguna menggeraka kepalanya ke atas.

Algoritma 1 akan mengembalikan nilai true jika pengguna sedang mengangguk. Baris ke-2 hingga baris ke-8 adalah untuk mengecek kecepatan putar sekarang, apakah sudah melampaui batas yang ditentukan atau belum. Untuk mengetahui titik potong secara akurat dapat menggunakan rumus persamaan garis dari dua buah titik yang dinotasikan dengan Notasi 3.1.

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1} \quad (3.1)$$

dengan,

$$\begin{aligned} y_1 &= lastY \text{AngSpeed} \\ y_2 &= y \text{AngSpeed} \\ x_1 &= lastT \\ x_2 &= currT \end{aligned}$$

Notasi 3.1 dijabarkan sesuai dengan penjabaran 3.2 sehingga menjadi rumus yang berada pada algoritma 1 baris ke-10. Nilai titik potong x ini juga menjadi indikator waktu untuk rentang waktu dari suatu bukit atau lembah, yang akan digunakan untuk mendapatkan jarak waktu pada saat pengguna melawan arah gerakan. Baris ke-14 hingga ke-25 pada algoritma 1 adalah untuk mengecek apakah simpangan yang terjadinya sudah melewati batas yang ditetapkan atau belum dan mencatatnya ke dalam attribut `passLimitDownDeviation` atau `passLimitUpDeviation`. Pada baris ke-26 hingga ke-28 untuk memulai menghitung luas yang baru. Baris ke-32 dan baris ke-33 untuk menghitung luas ketika tidak berpotongan dengan sumbu x. Baris ke-36 untuk pengecekan jarak antara bukit dengan lembah dapat diselesaikan dengan mengurangi waktu mulai bukit dengan waktu akhir lembah. Sehingga hasil untuk mengecek apakah semua batas sudah terpenuhi dapat di selesaikan dengan operasi (AND) biasa.

$$\begin{aligned} y &= 0 \\ \frac{0 - y_1}{y_2 - y_1} &= \frac{x - x_1}{x_2 - x_1} \\ \frac{-y_1}{y_2 - y_1} x_2 - x_1 &= x - x_1 \\ x &= \left(\frac{-y_1}{y_2 - y_1} x_2 - x_1 \right) + x_1 \end{aligned} \quad (3.2)$$

3.4.2 Algoritma Mendeteksi Gerakan Menggeleng

Sama seperti pada pendeksan gerakan mengangguk, algoritma ini akan dipanggil setiap kali ada perubahan nilai sensor, menyimpan batas-batas, waktu mulai dan berakhirnya suatu bukit atau lembah, di panggil secara berulang-ulang hingga menghasilkan hasil yang benar, dan data disimpan

Algorithm 1 Nod Detection Algoritm

```

1: function DETECTNOD( $yAngSpeed$ )
2:    $currT \leftarrow$  current Time
3:   if  $yAngSpeed > angSpeedLimit$  then
4:      $passLimitUp \leftarrow true$ 
5:      $currPassLimitUp \leftarrow true$ 
6:   else if  $yAngSpeed < angSpeedLimit * -1$  then
7:      $passLimitDown \leftarrow true$ 
8:      $currPassLimitDown \leftarrow true$ 
9:   if X values intersect with x(time) axis then
10:     $xIntersect \leftarrow ((-lastYAngSpeed/(yAngSpeed - lastYAngSpeed)) * (currT - lastT)) +$ 
      $lastT$ 
11:     $endTCurrMotion \leftarrow xIntersect$ 
12:     $areaBeforeIntersect \leftarrow ((xIntersect - lastT)/1000) * lastYAngSpeed/2$ 
13:     $calcArea \leftarrow calcArea + areaBeforeIntersect$ 
14:    if  $currPassLimitDown$  then
15:       $startTVValley \leftarrow startTCurrMotion$ 
16:       $endTVValley \leftarrow endTCurrMotion$ 
17:       $deviationMotionDown \leftarrow calcArea$ 
18:      if  $deviationMotionDown < deviationLimit * -1$  then
19:         $passLimitDownDeviation \leftarrow true$ 
20:    else if  $currPassLimitUp$  then
21:       $waktuMulaiBukit \leftarrow startTCurrMotion$ 
22:       $waktuAkhirBukit \leftarrow endTCurrMotion$ 
23:       $deviationMotionUp = calcArea$ 
24:      if  $deviationMotionUp > deviationLimit$  then
25:         $passLimitUpDeviation \leftarrow true$ 
26:     $calcArea \leftarrow 0$ 
27:     $areaAfterIntersect \leftarrow ((currT - xIntersect)/1000) * yAngSpeed/2$ 
28:     $calcArea \leftarrow calcArea + areaAfterIntersect$ 
29:     $startTCurrMotion \leftarrow xIntersect$ 
30:     $currPassLimitDown \leftarrow false$ 
31:     $currPassLimitUp \leftarrow false$ 
32:  else
33:     $calcArea \leftarrow calcArea + ((lastYAngSpeed + yAngSpeed)/1000) * (currT - lastT)/2$ 
34:   $lastYAngSpeed \leftarrow yAngSpeed$ 
35:   $lastT \leftarrow currT$ 
36:  if hill and valley time values has been set AND all condition have been met then return
      $true$ 
37: else return  $false$ 

```

pada attribut.

Sebagian attribut memiliki kegunaan yang sama dengan algoritma pendeksi gerakan mengangguk. Berikut adalah keterangan dari data-data yang disimpan pada attribut yang belum dijelaskan pada algoritma pendeksi gerakan mengangguk:

- **currentlyLimitLeft**, attribut ini akan menunjukkan bahwa pada saat ini gerakan pengguna sedang bergerak ke kiri dan melebihi batas kecepatan sudutnya.
- **currentlyLimitRight**, attribut ini akan menunjukkan bahwa pada saat ini gerakan pengguna sedang bergerak ke kanan dan melebihi batas kecepatan sudutnya.
- **lastXAngSpeed**, attribut ini akan memiliki kecepatan sudut X pada pemanggilan method sebelumnya.
- **deviationMotionLeft**, attribut ini akan menyimpan simpangan yang terjadi ketika pengguna menggeraka kepalanya ke kiri.
- **deviationMotioRight**, attribut ini akan menyimpan simpangan yang terjadi ketika pengguna menggeraka kepalanya ke kanan.

Sebagian besar algoritma untuk mendeksi gerakan menggeleng memiliki kemiripan dengan algoritma untuk mendeksi gerakan mengangguk. Pada algoritma menggeleng data waktu mulai dan berakhirnya suatu bukit atau lembah disimpan pada attribut dengan tipe data *List*. Attribut-attribut yang menggunakan tipe data *List* ini adalah **valleyTimes** dan **hillTimes**. Data-data waktu terjadinya bukti atau lembah yang dimasukkan ke dalam *List* ini diartikan sudah memenuhi syarat dari batas-batas simpangan dan kecepatan sudutnya. *List* ini akan dikosongkan kembali jika salah satu *List*-nya sudah lebih dari dua atau kedua *List* tersebut sudah memiliki isi sebanyak dua, atau lebih.

Untuk pengecekan jarak waktu antara lembah dengan bukit akan dilakukan sebanyak dua kali. Pengecekan pertama yaitu untuk pengecekan jarak waktu antara lembah atau bukit pertama dengan kedua. Pengecekan kedua yaitu untuk pengecekan jarak waktu antara lembah atau bukit kedua dengan ketiga. Penghitungan jarak waktu antara lembah atau bukit pertama dengan kedua dapat dilakukan dengan mengurangi waktu dimulainya lembah atau bukit kedua dengan waktu berakhirnya lembah atau bukit pertama. Begitu pula dengan penghitungan jarak waktu antara lembah atau bukit kedua dengan ketiga.

Algorithm 2 Shook Detection Algorit

```

1: function DETECTSHOOK( $xAngSpeed$ )
2:    $currT \leftarrow$  current Time
3:   if  $xAngSpeed > angSpeedLimit$  then
4:      $currPassLimitLeft \leftarrow$  true
5:   else if  $xAngSpeed < angSpeedLimit * -1$  then
6:      $currPassLimitRight \leftarrow$  true
7:   if X values intersect with x(time) axis then
8:      $xIntersect \leftarrow ((-lastXAngSpeed/(xAngSpeed - lastXAngSpeed)) * (currT - lastT)) +$ 
       $lastT$ 
9:      $endTCurrMotion \leftarrow xIntersect$ 
10:     $areaBeforeIntersect \leftarrow ((xIntersect - lastT)/1000) * lastXAngSpeed/2$ 
11:     $calcArea \leftarrow calcArea + areaBeforeIntersect$ 
12:    if  $currPassLimitRight$  then
13:       $deviationMotionRight \leftarrow calcArea$ 
14:      if  $deviationMotionRight < deviationLimit$  then
15:         $valleyTimes.add(\text{start and end time valley})$ 
16:    else if  $currPassLimitLeft$  then
17:       $deviationMotionUp = calcArea$ 
18:      if  $deviationMotionUp > deviationLimit * -1$  then
19:         $hillTimes.add(\text{start and end time hill})$ 
20:       $calcArea \leftarrow 0$ 
21:       $areaAfterIntersect \leftarrow ((currT - xIntersect)/1000) * xAngSpeed/2$ 
22:       $calcArea \leftarrow calcArea + areaAfterIntersect$ 
23:       $startTCurrMotion \leftarrow xIntersect$ 
24:       $currPassLimitRight \leftarrow false$ 
25:       $currPassLimitLeft \leftarrow false$ 
26:    else
27:       $calcArea \leftarrow calcArea + ((lastXAngSpeed + xAngSpeed)/1000) * (currT - lastT)/2$ 
28:       $lastXAngSpeed \leftarrow xAngSpeed$ 
29:       $lastT \leftarrow currT$ 
30:      if head moves right first AND time range between hill and valley no exceed the limit. then
31:        return true
32:      else if head moves left first AND time range between hill and valley no exceed the limit.
        then return true
33:      else return false

```

BAB 4

PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan aplikasi yang akan dibangun meliputi perancangan kelas algoritma pendekripsi gerakan kepala, *GameObject*, *Gameplay*, dan perancangan antarmuka.

4.1 Perancangan Kelas Algoritma Pendekripsi Gerakan Kepala

Pada subbab 3.4 telah dijelaskan algoritma pendekripsi gerakan kepala secara prosedural. Untuk merapihkan algoritmanya, dibuatlah diagram kelas rinci untuk memenuhi kebutuhan dalam pembangunan aplikasi. Desain diagram kelas ini pada dasarnya membagi masalah-masalah pada algoritma di subbab 3.4 menjadi masalah-masalah yang lebih sederhana. Kemudian setiap masalah disatukan kembali untuk menyelesaikan masalah pendekripsi kepala. Deskripsi kelas beserta fungsi dari diagram kelas akan dijelaskan sebagai berikut:

- Package Controller
 - Kelas GameOverController

- Kelas GameOverController

Kelas ini bertujuan untuk mendekripsi kondisi *game over* dan menampilkan pesan *game over*. Atribut-atribut pada kelas ini adalah sebagai berikut:

- * public QuestionMessageController[] itemMessages

Atribut ini digunakan untuk menyimpan seluruh QuestionMessageController yang ada pada permainan agar dapat diberhentikan dan diatur ulang.

Method-method pada kelas ini adalah sebagai berikut:

- * public void restartGame()

Method ini berfungsi untuk mengembalikan kondisi permainan ke kondisi awal, untuk mengulang permainan jika permainan telah selesai.

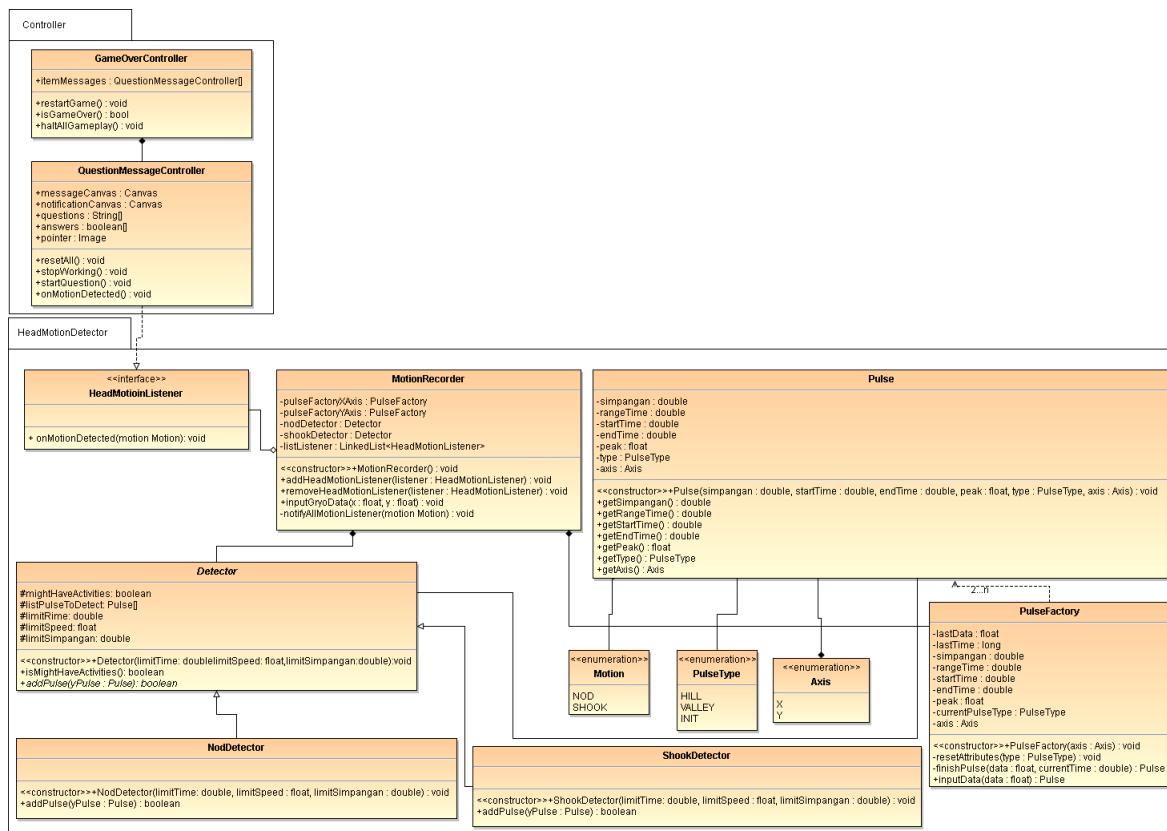
- * public boolean isGameOver()

Method ini berfungsi untuk mengecek apakah permainan sudah selesai atau belum.

Method ini akan dipanggil secara berulang-ulang setiap satu *frame* pada permainan.

- * public void haltAllGameplay()

Method ini berfungsi untuk memberhentikan seluruh fungsi permainan. Method ini akan dipanggil ketika permainan telah selesai.



Gambar 4.1: Diagram Kelas Algoritma Pendekripsi Gerakan Kepala.

– Kelas QuestionMessageController

Kelas ini digunakan untuk mengatur tampilan pesan pertanyaan pada permainan. Kelas ini juga mengatur kondisi permainan bergantung dari masukkan pengguna, yaitu anggukan dan gelangan kepala. Atribut-atribut pada kelas ini adalah sebagai berikut.

- * public Canvas messageCanvas

Atribut ini digunakan untuk mendapatkan Canvas yang menampilkan pesan pertanyaan kepada pengguna.

- * public Canvas notificationCanvas

Atribut ini digunakan untuk mendapatkan Canvas yang menampilkan pesan pemberitahuan kepada pengguna.

- * public String[] question

Atribut ini digunakan untuk menyimpan pertanyaan-pertanyaan yang akan ditampilkan pada messageCanvas.

- * public boolean[] answers

Atribut ini digunakan untuk menyimpan jawaban-jawaban penentu untuk setiap pertanyaan yang ada.

- * public Image pointer

Atribut ini digunakan untuk mendapatkan gambar yang digunakan sebagai pengukur kondisi permainan.

Method-method pada kelas ini adalah sebagai berikut:

- * public void resetAll()
Method ini berfungsi untuk mengembalikan kondisi permainan menjadi kondisi awal permainan.
- * public void stopWorking()
Method ini berfungsi untuk memberhentikan semua fungsi pada suatu objek QuestionMessageController.
- * public void startQuestion()
Method ini berfungsi untuk memulai atau memunculkan suatu pertanyaan pada tampilan.
- * public void onMotionDetected(Motion motion)
Method ini akan dipanggil jika terdeteksi suatu gerakan kepala. Parameter motion akan diisi dengan spesifikasi gerakan yang terdeteksi.

- Package HeadMotionDetector

- interface HeadMotionListener

Interface ini akan di implements oleh kelas-kelas yang akan menggunakan hasil dari pendekripsi gerakan kepala. Method-method abstrak pada interface ini adalah sebagai berikut:

- * public void onMotionDetected(Motion motion)
Method ini akan dipanggil oleh jika terdeteksi suatu gerakan kepala. Parameter motion akan diisi dengan spesifikasi gerakan yang terdeteksi.

- Kelas MotionRecorder

Kelas ini digunakan untuk merekam, menentukan, dan mendekripsi gerakan kepala. Atribut-atribut pada kelas ini adalah sebagai berikut:

- * private PulseFactory pulseFactoryXAxis
Atribut ini digunakan untuk menyimpan PulseFactory yang merekam data gyroscope pada sumbu X.
 - * private PulseFactory pulseFactoryYAxis
Atribut ini digunakan untuk menyimpan PulseFactory yang merekam data gyroscope pada sumbu Y.
 - * private Detector nodDetector
Atribut ini digunakan untuk menyimpan Pendekripsi gerakan mengangguk.
 - * private Detector shookDetector
Atribut ini digunakan untuk menyimpan Pendekripsi gerakan menggeleng.
 - * private LinkedList<HeadMotionListener> listListener
Atribut ini digunakan untuk menyimpan kelas-kelas yang menjadi *listener* pendekripsi gerakan kepala.

Method-method pada kelas ini adalah sebagai berikut:

- * public MotionRecorder()
Constructor ini digunakan untuk menginisialisasi atribut-atribut yang akan digunakan.

- * public void addHeadMotionListener(HeadMotionListener listener)
Method ini digunakan untuk menambahkan *listener* baru pada atribut listListener
- * public void removeHeadMotionListener(HeadMotionListener listener)
Method ini digunakan untuk menghapus suatu *listener* pada atribut listListener.
- * public void inputGyroData(float x, float y)
Method ini digunakan untuk memasukkan data gyroscope yang akan digunakan untuk mendeteksi gerakan mengangguk dengan menggeleng.
- * public void notifyAllMotionListener(Motion motion)
Method ini akan memberitahu seluruh *listener* ketika terdeteksi suatu gerakan. Jenis gerakan yang diberitahukan adalah jenis gerakan pada parameter tersebut.

– Kelas PulseFactory

Kelas ini berguna untuk mendeteksi *Pulse* yang terjadi pada grafik data gyroscope. Beberapa atribut yang dimiliki oleh kelas ini akan menjadi kriteria dari suatu *Pulse* yang akan terdeteksi. Atribut-atribut tersebut adalah:

- * private float lastData
Atribut ini menyimpan data sebelum data yang baru dimasukkan
- * private float lastTime
Atribut ini menyimpan waktu pada pemasukkan data sebelum daya yang baru dimasukkan.
- * private double simpangan
Atribut ini menyimpan simpangan terjauh pada *Pulse* terakhir yang terdeteksi.
- * private double rangeTime
Atribut ini menyimpan rentang waktu yang terjadi pada suatu *Pulse*
- * private double startTime
Atribut ini menyimpan waktu mulai dari suatu *Pulse* yang terjadi.
- * private double endTime
Atribut ini menyimpan waktu akhir dari suatu *Pulse* yang terjadi.
- * private float peak
Atribut ini menyimpan puncak nilai tertinggi atau terendah dari suatu *Pulse* bukit atau lembah.
- * private PulseType
Atribut ini menyimpan tipe *Pulse* antara bukit atau lembah.
- * private Axis axis
Atribut ini mendefinisikan sumbu yang di rekam datanya.

Method-method pada kelas ini adalah:

- * public PulseFactory(Axis axis)
Constructor ini berfungsi untuk mendefinisikan sumbu pada gyroscope yang akan dideteksi.
- * private void resetAttributes()
Method ini berfungsi untuk mengembalikan nilai-nilai atribut kembali ke nilai asal.

* private Pulse finishPulse(float data, double currentTime)

Method ini akan dipanggil ketika grafik melewati angka nol. Pada pemanggilan ini *Pulse* baru akan terdeteksi dan resetAttributes() akan dipanggil untuk pendekripsi *Pulse* selanjutnya.

* public Pulse inputData(float data)

Method ini digunakan untuk memasukkan data gyroscope baru.

– Kelas Detector

Kelas ini merupakan kelas yang mendefinisikan pendekripsi gerakan kepala. Sub-class kelas ini adalah NodDetector dengan ShookDetector Atribut-atribut pada kelas ini adalah:

* public Detector(float limitSpeed, double limitSimpangan)

Constructor ini digunakan untuk meninisialisasi batasan-batasan yang didefinisikan pada atribut-atribut kelas ini.

* protected boolean mightHaveActivities()

Atribut ini menunjukkan apakah suatu detector memiliki kemungkinan sedang mendekripsi gerakan atau tidak.

* protected Pulse[] listPulseToDetect()

Atribut ini menyimpan beberapa *Pulse* untuk di periksa karakteristiknya, apakah sesuai dengan algoritma pendekripsi gerakan kepala.

* protected float limitSpeed()

Atribut ini mendefinisikan batas kecepatan angular yang sesuai dengan pendekripsi gerakan kepala.

* protected double limitSimpangan()

Atribut ini mendefinisikan batas Simpangan yang sesuai dengan pendekripsi gerakan kepala.

– Kelas NodDetector dengan ShookDetector

Kelas ini algoritma untuk mendekripsi gerakan mengangguk atau menggeleng. Kelas ini merupakan turunan dari kelas Detector

– Kelas Pulse

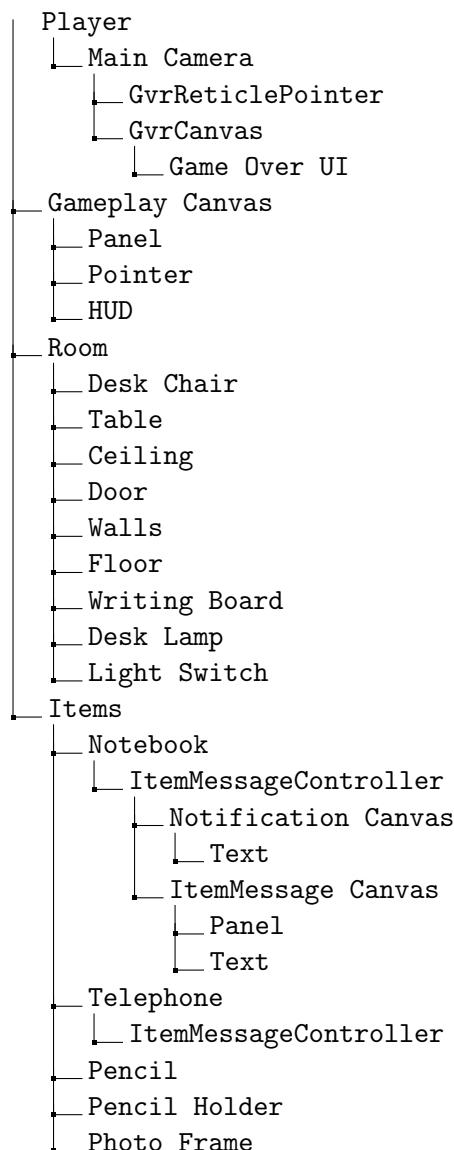
Kelas ini digunakan hanya untuk menyimpan karakteristik-karakteristik dari suatu *Pulse* yang terbentuk oleh kelas PulseDetector. Atribut-atribut pada kelas ini serupa dengan atribut-atribut pada kelas PulseDetector. Method-method pada kelas ini hanyalah Getter dengan Constructor saja.

4.2 Perancangan Hierarki GameObject pada Unity

Perancangan hierarki pada Unity merupakan salah satu perancangan yang cukup penting. Perancangan ini penting karena mempermudah pencarian, pemilihan, perubahan transformasi GameObject pada dunia permainan yang sedang dibuat. Selain itu perancangan ini dapat membantu pengguna Unity untuk membuat suatu GameObject yang memiliki sifat relatif terhadap suatu GameObject yang lain. Oleh Karena itu perancangan hierarki akan di bahas pada subbab ini.

permainan ini hanya akan dibangun dengan menggunakan satu buah scene saja. permainan ini tidak memerlukan Scene yang banyak karena permainan ini hanya memiliki satu ruangan saja.

Sebagian besar pada perancangan ini hanyalah mengelompokkan GameObject-GameObject pada Scene. Gambar 4.2 merupakan hasil dari perancangan hierarki permainan ini.



Gambar 4.2: Perancangan hierarki Game Object

Pada Perancangan di atas ada beberapa GameObject utama, diantaranya adalah Player, GameplayCanvas, Room, dan Items. GameObject Player akan digunakan sebagai camera pada dunia permainan tersebut. Camera pada Player ini akan bergerak mengikuti orientasi pada *smartphone*. Pada GameObject ini terdapat GameObject Main Camera yang berfungsi untuk menjadi kamera pada permainan dan menentukan titik pandang pengguna. Main Camera memiliki dua buah *Child* Game Object. Kedua Game Object tersebut adalah GvrReticlePointer dan GameOverCanvas. Gvr-ReticlePointer berguna untuk menampilkan titik pandang pengguna. GvrReticlePointer ini akan menjadi lingkaran ketika objek yang di pandangnya dapat memberikan respons dari masukkan tombol pada Google Cardboard. GameOverCanvas berguna untuk menampilkan pesan "Game Over!" ketika permainan berakhir.

GameObject GameplayCanvas digunakan untuk menunjukkan kondisi permainan sekarang. GameObject ini memiliki dua buah child yaitu Panel, Pointer, dan HUD. Panel berfungsi sebagai

bidang putih pada UI. Pointer berfungsi sebagai menunjuk kondisi permainan pada saat ini. HUD adalah bentuk kurang lebih 1/6 lingkaran yang membatasi Pointer.

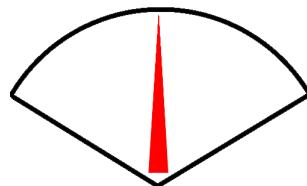
GameObject Room digunakan untuk menyimpan model-model dekorasi dalam ruangan. *Child* dari GameObject hanya merupakan model-model tiga dimensi yang membentuk ruangan pada dunia game.

GameObject Item digunakan untuk menyimpan model-model barang yang ada pada ruangan. Sebagian GameObject pada Item akan memiliki component yang akan diberikan script untuk mendeteksi gerakan kepala. GameObject yang akan diberikan script tersebut adalah GameObject yang memiliki *Child* GameObject ItemMessageController, yaitu Notebook dengan Telephone. Selain kedua GameObject tersebut GameObject lainnya merupakan model-model dekorasi pada meja.

4.3 Perancangan *Gameplay* dan Perancangan Antarmuka

permainan ini merupakan permainan yang pemainnya berperan sebagai pemimpin dari suatu perusahaan. Tugas dari permainan ini adalah memilih pilihan-pilihan yang sesuai dari kondisi yang sedang terjadi. Pemain akan diberi pertanyaan dalam mengatur perusahaan tersebut. Pertanyaan tersebut hanya dapat dijawab ya atau tidak. Jawaban ya dilakukan dengan mengangguk dan jawaban tidak dengan menggeleng. Jawaban yang mengacu untuk menguntungkan perusahaan akan membuat tingkat kesenangan karyawan. Jawaban yang mengacu untuk menyenangkan karyawan akan merugikan perusahaan.

Takaran kesenangan karyawan dengan keuntungan perusahaan di gambarkan dengan meteran seperti pada gambar 4.3. Jika jarum pada meteran tersebut mengarah ke kiri, berarti karyawan perusahaan sedang memiliki tingkat kesenangan yang tinggi, namun perusahaan tidak mendapatkan keuntungan yang baik. Begitu pula sebaliknya jika jarum mengarah ke kanan.



Gambar 4.3: Meteran pada permainan.

Tidak ada kondisi menang pada permainan ini, karena permainan ini akan terus berlangsung hingga pemain kalah. Pemain akan kalah jika meteran terlalu mengarah ke kiri, atau terlalu mengarah ke kanan. Ketika pemain sudah kalah, akan muncul pesan "Game Over!" (Gambar 4.5). Jika pemain menarik/menekan tombol pada Google Cardboard-nya, permainan akan dimulai dari awal.

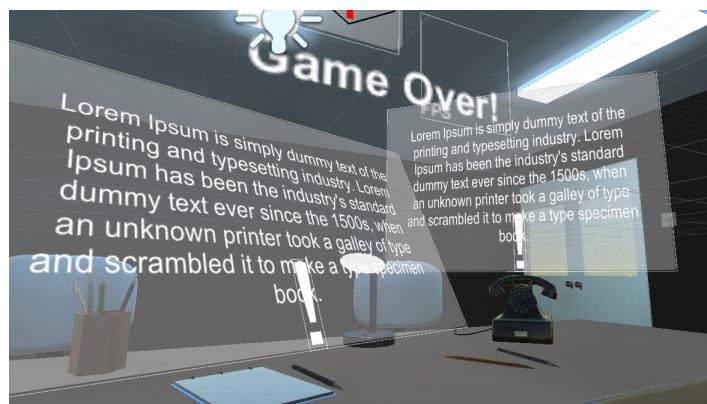
Antarmuka yang ditampilkan pada permainan ini merupakan pemandangan seseorang pemimpin perusahaan (Pemain) pada meja kerjanya. Ruangan tersebut akan sangat sederhana, yaitu ruangan dengan bentuk balok biasa dan diisi dengan beberapa model untuk dekorasi. Pada meja kerja akan terdapat beberapa alat seperti, pensi, buku catatan, telepon, lampu, dan lain sebagainya. Alat-alat

yang akan ditambahkan *script C#* untuk dapat mendeteksi anggukan dan gelengan adalah telepon dan buku catatan. Ruangan tersebut digambarkan pada gambar 4.4



Gambar 4.4: Desain ruangan untuk permainan yang akan dibuat.

Untuk menampilkan pertanyaan, pemain harus menunggu suatu alat menampilkan notifikasi yang ditunjukkan dengan memunculkan karakter tanda seru (!) di atas alat tersebut. Ketika notifikasi tersebut muncul, pemain dapat menghadapkan pandangannya ke alat tersebut dan menarik/menekan tombol pada Google Cardboard untuk menampilkan pertanyaan yang akan diberikan. Pertanyaan yang diberikan akan diberikan dengan memunculkan suatu *pop-up* di atas alat tersebut yang ditulis pertanyaan yang ditanyakan (Notifikasi dan *pop-up* ditunjukkan pada gambar 4.5). Jumlah pertanyaan yang dapat dimunculkan hanyalah satu pertanyaan saja, tidak dapat memunculkan dua buat pertanyaan secara bersamaan. Setelah pertanyaan dijawab dengan anggukan atau gelengan *pop-up* tersebut akan menghilang, dan meteran akan bergerak tergantung dari jawaban pemain.



Gambar 4.5: Desain User Interface

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini terdiri atas implementasi, pengujian, dan masalah yang dihadapi. Pada bagian implementasi dijelaskan mengenai lingkungan implementasi dan hasil dari implementasi. Pada bagian pengujian berisi hasil dari pengujian. Pada bagian masalah yang dihadapi akan dijelaskan masalah-masalah menarik yang dihadapi pada saat implementasi.

5.1 Implementasi

Implementasi ini dilakukan dengan menggunakan *Game Engine* Unity. Implementasi dilakukan pada satu buah laptop, dan aplikasi dijalankan pada tiga buah *smartphone* Android.

5.1.1 Lingkungan Implementasi dan Pengujian

Berikut adalah spesifikasi laptop yang digunakan untuk implementasi:

1. Processor : AMD A10-5750M Quad-core 2.5 - 3.5 Ghz
2. RAM : 8GB (7.21 Usable)
3. VGA : AMD Radeon HD 8650G + HD 8670M Dual Graphics
4. Sistem Operasi : Windows 10 64-bit
5. Versi Unity : 5.5.1f1 Personal Edition
6. Google VR SDK for Unity : 1.1

Tabel 5.1 adalah perangkat-perangkat *smartphone* Android yang digunakan:

5.1.2 Hasil Implementasi

Hasil Implementasi ini adalah aplikasi permainan VR berbasis Android yang menggunakan Game Engine Unity. Aplikasi ini dapat di *install* dengan menggunakan *Package Installer* yang berekstensi file ".apk" pada Android.

1. Aplikasi saat pertama kali terbuka.

Pada saat pertama kali dijalankan permainan akan langsung dimulai (Gambar 5.1). Jika perangkat tidak memiliki sensor Gyroscope, aplikasi akan menampilkan *pop-up* (Gambar 5.2) yang menunjukkan bahwa perangkat ini tidak dapat menjalankan aplikasi ini. Tombol 'X'

No	Processor	Sistem Operasi	Sensor-sensor
1	Exynos Dual-core 1.4 GHz Cortex-A9	Android OS v4.1.2 (Jelly Bean)	Accelerometer, gyro, proximity, compass, barometer
2	Qualcomm Snapdragon 800 Quad-core 2.3 GHz Krait 400	Android OS v6.0 (Marshmallow)	Accelerometer, gyro, proximity, compass, barometer
3	Qualcomm Snapdragon 625 Octa-core 2.0 GHz Cortex-A53	Android OS v6.0 (Marshmallow)	Fingerprint (rear-mounted), accelerometer, gyro, proximity, compass
4	Quad-core 1.2 GHz Cortex-A7	Android OS v4.4 (KitKat)	Accelerometer, proximity
5	1.3 GHz octa-core CPU, MediaTek Helio X10 MT6753 chipset	Android OS v6.0 (Marshmallow)	Accelerometer, Proximity, Compass, Light sensor, Fingerprint, Gyroscope(Software)
6	Qualcomm MSM8916 Snapdragon 410	Android 4.4.4 (KitKat)	Accelerometer, gyro, proximity, compass

Tabel 5.1: Tabel Perangkat yang digunakan

pada pojok kiri atas akan selalu ada pada permainan ini dan berguna sebagai tombol untuk keluar dari permainan.

2. Aplikasi saat muncul notifikasi.

Ketika ada notifikasi, akan muncul tanda seru (!) di atas benda yang memiliki notifikasi seperti pada Gambar 5.3.

3. Aplikasi saat memunculkan pertanyaan.

Ketika pengguna sudah memilih suatu notifikasi yang muncul, akan muncul jendela pertanyaan di atas benda yang dipilih (Gambar 5.4).

4. Aplikasi setelah memberikan respons mengangguk atau menggeleng.

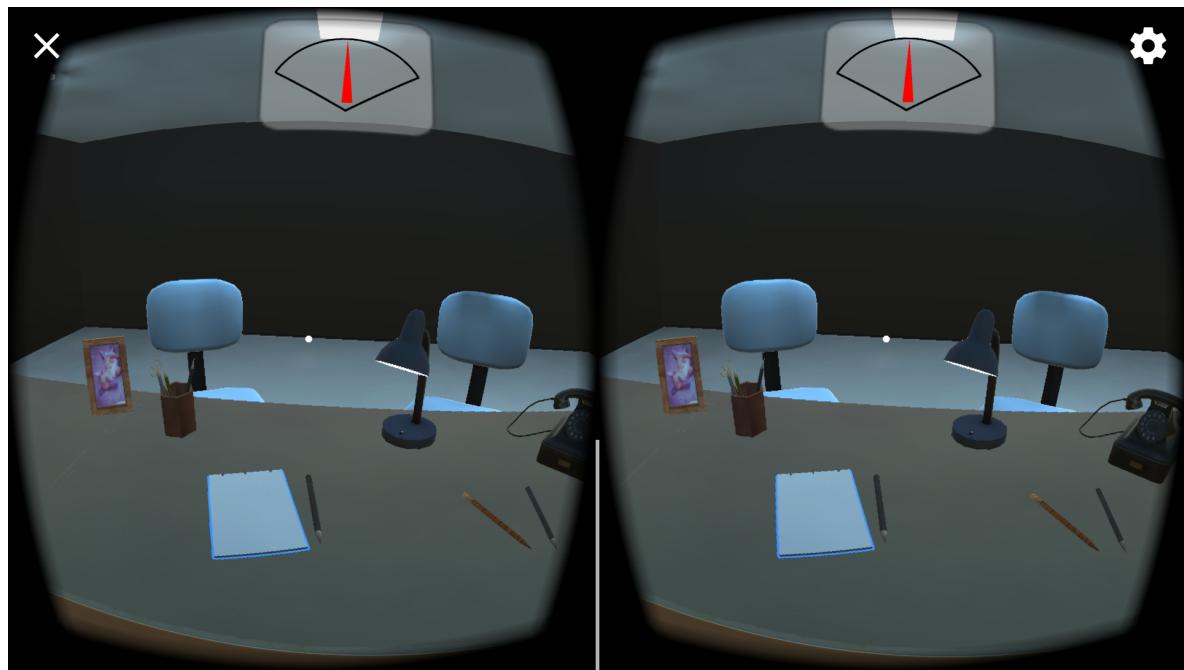
Ketika pengguna mengangguk pada saat pertanyaan diberikan, tulisan pada panel pertanyaan akan berubah menjadi kata "YES"(Gambar 5.5) dan "NO" ketika pengguna menggeleng(Gambar 5.6).

5. Aplikasi ketika pemain kalah dalam permainan tersebut.

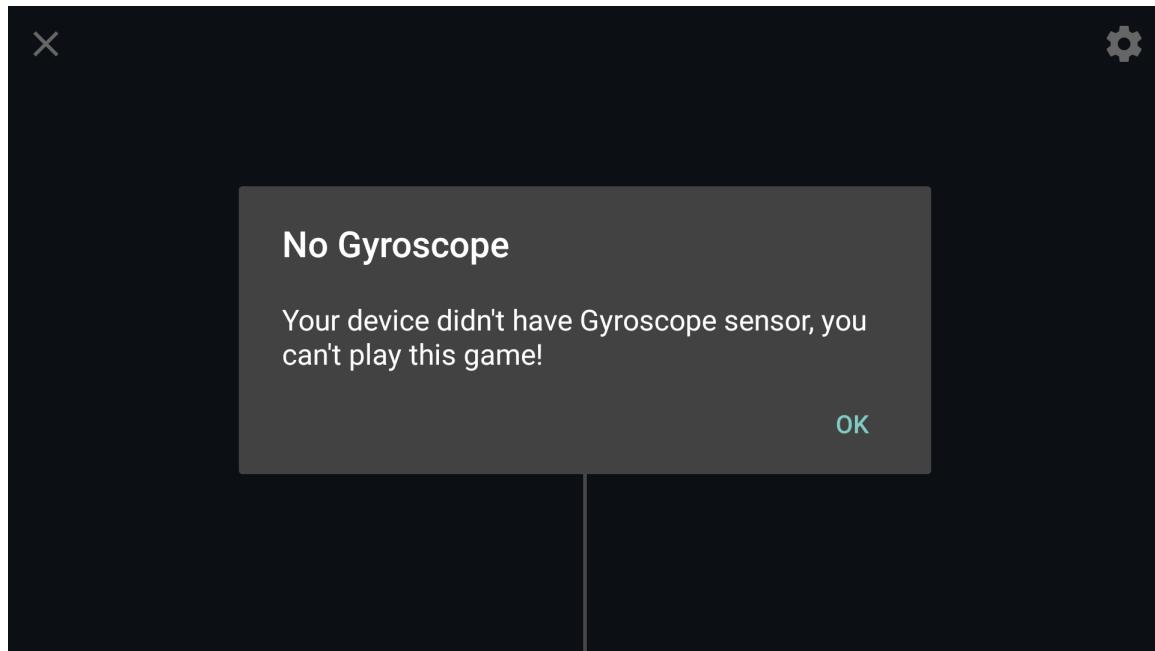
Ketika pengguna kalah, maka akan muncul tulisan "Game Over!" di depan pandangan pengguna (Gambar 5.7). Tulisan ini akan mengikuti pergerakan kepala pengguna. Jika pengguna menarik/menekan tombol pada Google Cardboard, permainan akan diulang menjadi kondisi awal.

5.2 Pengujian

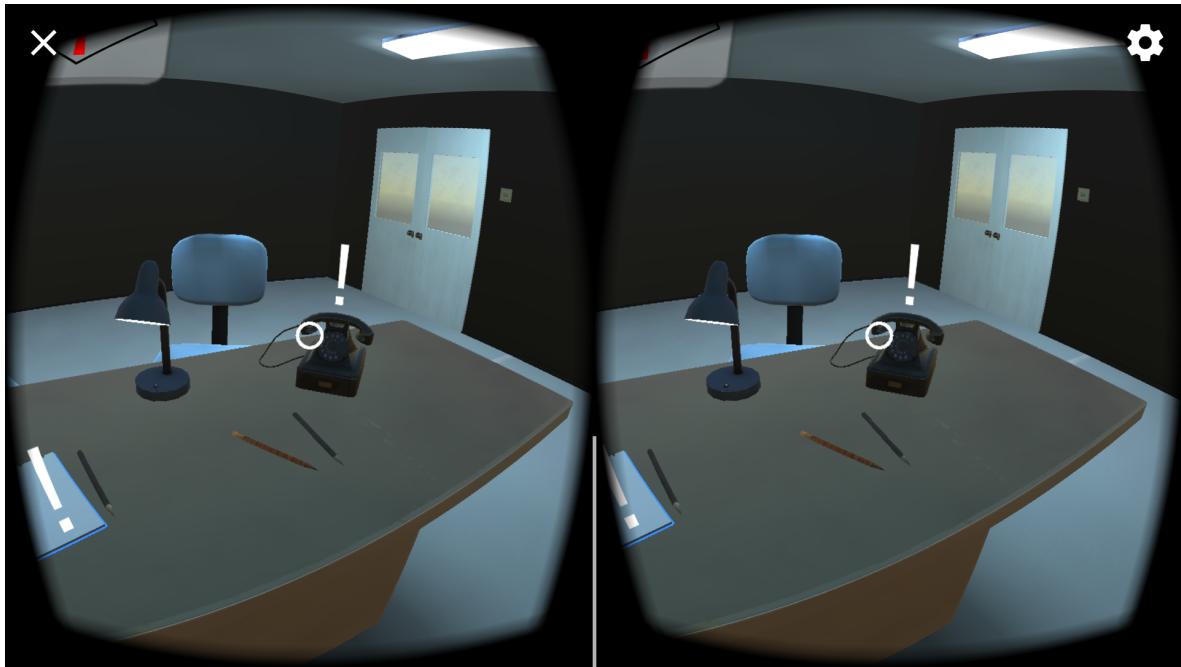
Pengujian dilakukan dengan menggunakan dua buah metode yaitu pengujian fungsional dan pengujian eksperimental. Pengujian fungsional bertujuan untuk menguji fungsi-fungsi yang disediakan



Gambar 5.1: Tampilan ketika aplikasi baru dimulai.



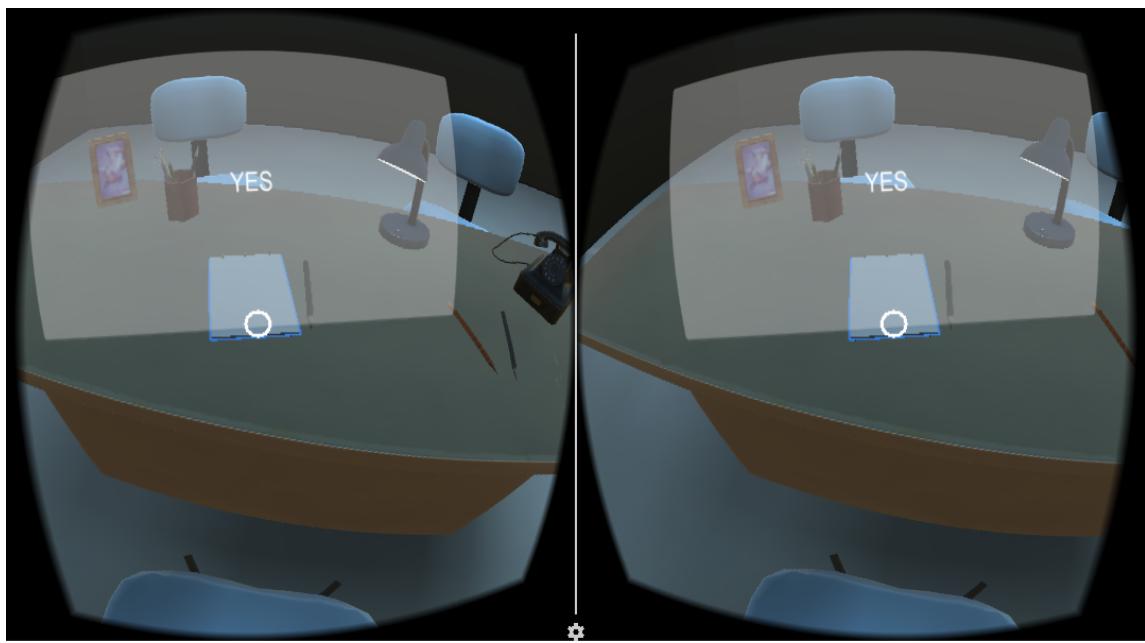
Gambar 5.2: Tampilan ketika perangkat tidak memiliki sensor Gyroscope.



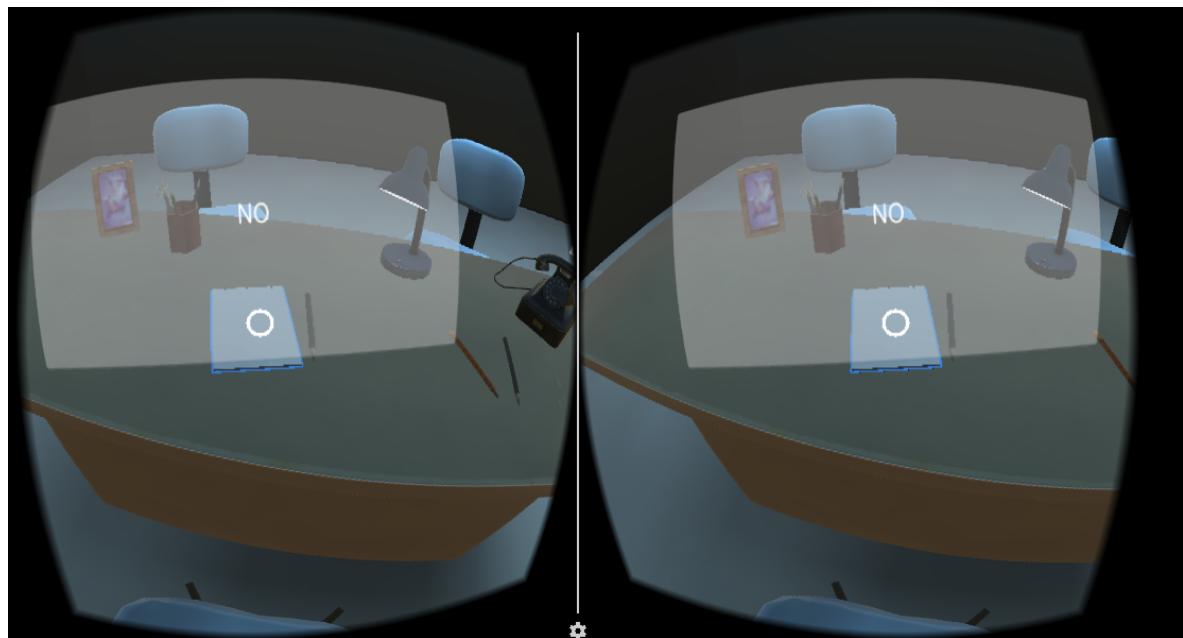
Gambar 5.3: Tampilan ketika notifikasi muncul.



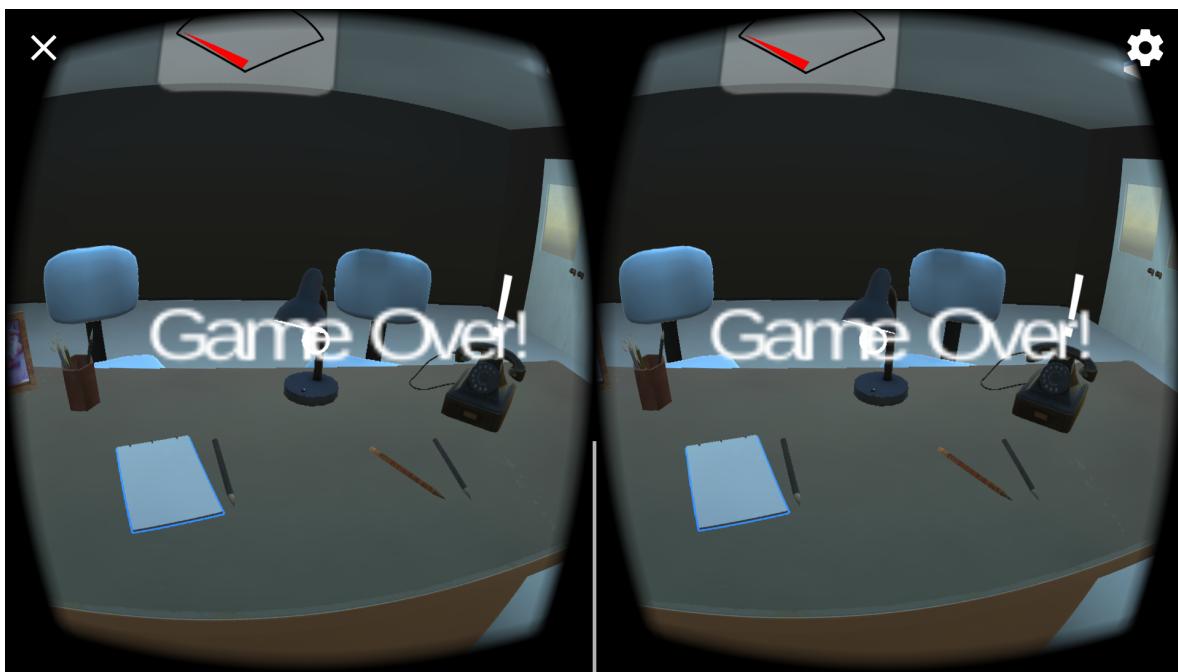
Gambar 5.4: Tampilan ketika *pop-up* pertanyaan muncul.



Gambar 5.5: Tampilan setelah pengguna mengangguk.



Gambar 5.6: Tampilan setelah pengguna menggeleng.



Gambar 5.7: Tampilan setelah permainan usai.

pada aplikasi. Pengujian Eksperimental bertujuan untuk menguji pengalaman pengguna dalam menggunakan aplikasi ini.

5.2.1 Pengujian Fungsional

Pengujian Fungsional ini dilakukan dengan mencoba menjalankan aplikasi pada perangkat-perangkat yang berbeda spesifikasi-nya.

Pada *smartphone* 1 terjadi kegagalan pada saat instalasi pada perangkat. Perangkat ini mengeluarkan *error* pada Google VR API for Unity, tetapi perangkat ini dapat berjalan dengan baik ketika menjalankan aplikasi untuk menganalisis algoritma pendekripsi gerakan kepala. Aplikasi-aplikasi Google VR lainnya yang tidak menggunakan Unity dalam membuatnya dapat berjalan dengan baik. Hal ini disebabkan karena pada Google VR SDK for Unity membutuhkan Sistem Operasi minimum pada versi 4.4 (KitKat), sehingga perangkat ini tidak dapat menjalankan aplikasi ini. Aplikasi-aplikasi Google Cardboard yang dibuat dengan menggunakan Google Cardboard SDK akan dapat berjalan dengan baik karena Google Cardboard SDK membutuhkan Sistem Operasi minimum pada versi 4.1 (Jelly Bean).

Pada *smartphone* 2 aplikasi dapat berjalan dengan baik.

Pada *smartphone* 3 aplikasi dapat berjalan dengan baik.

Pada *smartphone* 4 aplikasi dapat di-*install* dengan baik. Ketika membuka aplikasi tersebut menampilkan pesan *error* bahwa perangkat yang digunakan tidak memiliki sensor Gyroscope. Hal ini terjadi karena aplikasi mengecek terlebih dahulu apakah perangkat memiliki sensor Gyroscope. Setelah menampilkan pesan *error* tersebut aplikasi memberhentikan dirinya sendiri.

Pada *smartphone* 5 aplikasi dapat berjalan dengan baik, tetapi pergerakan kamera pada aplikasi terkadang tidak sesuai dengan pergerakan. Hal ini dikarenakan pada perangkat ini terdeteksi memiliki sensor Gryoscope, namun sensor gyroscope yang terbentuk pada perangkat ini merupakan-

an sensor yang menggunakan *software* untuk memenuhi fungsi gyroskopinya. Sensor Gyroscope yang terbentuk oleh *software* akan lebih tidak akurat dibandingkan dengan sensor Gyroscope yang terbentuk dari hardware. Oleh sebab itu pada perangkat ini aplikasi ini tidak berjalan begitu baik.

Pada *smartphone* 6 aplikasi dapat berjalan dengan baik.

5.2.2 Pengujian Eksperimental

Pengujian eksperimental dilakukan dengan menguji aplikasi ini kepada lima pengguna berbeda. Pada saat pencobaan kelima pengguna tersebut akan memainkan permainan ini. Kemudian mencatat pendapat, kritik, atau saran dari pengguna tentang permainan ini.

Berikut adalah pengguna-pengguna pada pengujian aplikasi ini:

1. Ricky Setiawan (Gambar 5.8)

"Aplikasi-nya bagus, untuk masukkan anggukan dan gelengan harus memperkirakan terlebih dahulu seberapa simpangan yang dibutuhkan agar dapat terdeteksi."



Gambar 5.8: Pengujian oleh Ricky Setiawan.

2. Harseto Pandityo (Gambar 5.9)

"Permainannya bagus dan menarik, khususnya pada fungsi mengangguk dengan menggeleng. Terkadang ada masalah ketika mengangguk tidak terdeteksi, tetapi harus mengangguk dengan lebih jauh lagi simpangan-nya agar dapat terdeteksi. Tidak ada masalah pada saat menggeleng."

3. Herfan Heryandi (Gambar 5.10)

"Untuk geleng sedikit kurang sensitif, tetapi untuk mengangguk sudah sensitif."

5.3 Masalah yang Dihadapi pada Saat Implementasi

Berikut adalah beberapa masalah yang dihadapi pada saat implementasi:



Gambar 5.9: Pengujian oleh Harseto Pandityo.



Gambar 5.10: Pengujian oleh Herfan Heryandi.

1. Sulit dalam mencari permainan *open source* yang telah ada. Rencana awal dalam meng-implementasi algoritma ini adalah dengan mencari permainan *open source* yang telah ada, namun permainan *open source* yang telah mengimplementasi Google VR sangatlah langka. Ketika telah menemukan permainan *open source* yaitu "Doom VR", SDK yang digunakan oleh permainan merupakan SDK dengan versi lama yang sudah tidak dikembangkan lagi oleh Google. Sehingga saya tidak dapat menemukan SDK Google VR versi lama. Jadi masalah ini diselesaikan dengan membuat suatu game baru dengan bantuan Game Engine Unity.
2. Penggunaan Quaternion untuk menyimpan orientasi GameObject-GameObject pada Unity membuat bingung pada saat implementasi. Hal ini terjadi karena penggunaan Quaternion yang di konversi menjadi sudut Euler membuat objek tersebut tidak dapat menyimpan sudut putar yang lebih besar dari 360 derajat dan lebih kecil dari 0 derajat.
3. Google VR for Unity belum sepenuhnya stabil. Pada saat instalasi Google VR for Unity terjadi *compile error*. Kesalahan *compile error* ini tidak diberikan penyelesaiannya pada website resmi Google VR maupun Unity. Sehingga saya mencari solusi-nya pada forum pengguna Unity. Satu-satunya solusi untuk menyelesaikan solusi ini adalah dengan mengganti satu baris kode pada Google VR for Unity. *Compile error* tersebut terjadi pada script C# "GvrVideo-PlayerTexture.cs" pada baris 595. Solusinya yaitu dengan menambahkan kode "yield break;" sebelum perintah "return;". Sekarang masalah ini sudah diperbaiki oleh Google VR sehingga tidak terjadi lagi.

BAB 6

KESIMPULAN

6.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan, diperoleh kesimpulan-kesimpulan sebagai berikut:

1. Untuk dapat mendeteksi gerakan kepala khususnya mengangguk dan menggeleng, hanya diperlukan sensor gyroscope saja. Sensor-sensor lainnya tidak diperlukan untuk membantu mendeteksian gerakan kepala. Selain dapat menggunakan gyroscope, berdasarkan pengujian fungsional sensor gabungan accelerometer dan magnetometer dapat juga digunakan sebagai pengganti sensor gyroscope (sensor gyroscope software) pada perangkat-perangkat tertentu.
2. Aplikasi telah dapat membaca gerakan kepala dengan baik. Hal ini ditunjukkan dengan tidak adanya kesalahan masukkan pada pengujian eksperimental.
3. Berdasarkan hasil pengujian eksperimental, algoritma pada aplikasi ini dapat berjalan dengan baik. Kelemahan pada algoritma ini berada pada simpangan anggukan dengan gelengan yang dilakukan, karena setiap orang mengangguk dan menggeleng dengan besar simpangan yang berbeda-beda.

6.2 Saran

Berdasarkan hasil penelitian yang dilakukan, berikut adalah beberapa saran untuk pengembangan:

1. Melakukan percobaan anggukan dan gelengan terlebih dahulu sebelum permainan dimulai, agar simpangan anggukan dengan gelengan sesuai dengan kriteria pengguna.
2. Menggunakan metode *machine learning* dalam mendeteksi anggukan dengan gelenggan.
3. Meningkatkan versi Google Cardboard menjadi Google Daydream.

DAFTAR REFERENSI

- [1] T. Parisi, *Learning virtual reality: developing immersive experiences and applications for desktop, web, and mobile*. O'Reilly Media, 1 ed., 2015.
- [2] G. J. Kim, *Designing virtual reality systems: the structured approach*. Springer, 2005.
- [3] J. Vince, *Introduction to virtual reality*. Springer, 2004.
- [4] “Google cardboard.” <https://vr.google.com/cardboard/>. [Online; diakses 10-September-2016].
- [5] “Sensor types.” <https://source.android.com/devices/sensors/sensor-types.html>. [Online; diakses 10-September-2016].
- [6] G. Bleser and D. Stricker, “Advanced tracking through efficient image processing and visual–inertial sensor fusion,” *Computers & Graphics*, vol. 33, no. 1, pp. 59–72, 2009.
- [7] A. Developers, “What is android,” 2011.
- [8] “Android 7.0 nougat!.” <https://developer.android.com/>. [Online; diakses 12-September-2016].
- [9] “Google vr | google developers.” <https://developers.google.com/vr/>. [Online; diakses 20-September-2016].
- [10] J. B. Kuipers, *Quaternions and Rotation Sequences : A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 1998.
- [11] “Unity - game engine.” <https://unity3d.com/>. [Online; diakses 16-Februari-2016].
- [12] “Unity - manual.” <https://docs.unity3d.com/Manual/index.html>. [Online; diakses 16-Februari-2016].
- [13] “Aldin dynamics.” <http://www.aldindynamics.com/>. [Online; diakses 22-November-2016].

LAMPIRAN A

KODE PROGRAM APLIKASI UNTUK ANALISIS

Listing A.1: MainActivity.java

```
1 package com.example.egaprianto.testingsensors;
2
3 import android.content.Intent;
4 import android.hardware.Sensor;
5 import android.hardware.SensorManager;
6 import android.support.v7.app.AppCompatActivity;
7 import android.os.Bundle;
8 import android.view.View;
9 import android.widget.TextView;
10
11 public class MainActivity extends AppCompatActivity {
12     TextView mTextView;
13     private SensorManager mSensorManager;
14     private Sensor mSensor;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20     }
21
22     @Override
23     public void onDestroy() {
24         super.onDestroy();
25         android.os.Debug.stopMethodTracing();
26     }
27
28     public void onClickLightSensorButton(View view){
29         Intent intent = new Intent(this, SensorAccelerometerActivity.class);
30         startActivity(intent);
31     }
32
33     public void onRotVecSensorButton(View view){
34         Intent intent = new Intent(this, QuaternionTheta.class);
35         startActivity(intent);
36     }
37
38     public void recordAllRWSSensorData(View view){
39         Intent intent = new Intent(this, RecordAllRWSSensorData.class);
40         startActivity(intent);
41     }
42
43     public void onGyroSensorClicked(View view){
44         Intent intent = new Intent(this, SensorGyroscopeActivity.class);
45         startActivity(intent);
46     }
47     public void onMagSensorClicked(View view){
48         Intent intent = new Intent(this, SensorGeomagneticRotationActivity.class);
49         startActivity(intent);
50     }
51 }
```

Listing A.2: RecordAllRWSSensorData.java

```
1 package com.example.egaprianto.testingsensors;
2
3 import android.Manifest;
4 import android.content.Context;
5 import android.content.pm.PackageManager;
6 import android.hardware.Sensor;
7 import android.hardware.SensorEvent;
8 import android.hardware.SensorEventListener;
9 import android.hardware.SensorManager;
10 import android.media.Ringtone;
11 import android.media.RingtoneManager;
12 import android.net.Uri;
13 import android.os.Build;
14 import android.os.Bundle;
15 import android.os.CountDownTimer;
16 import android.os.Environment;
```

```

17 import android.support.v4.app.ActivityCompat;
18 import android.support.v7.app.AppCompatActivity;
19 import android.text.format.DateFormat;
20 import android.util.Log;
21 import android.view.View;
22 import android.view.WindowManager;
23 import android.widget.TextView;
24
25
26 import java.io.BufferedWriter;
27 import java.io.File;
28 import java.io.FileWriter;
29 import java.io.IOException;
30 import java.util.ArrayList;
31 import java.util.List;
32
33 public class RecordAllRawSensorData extends AppCompatActivity implements SensorEventListener {
34     private static final String FILENAME = "sensorRecord-";
35     private SensorManager mSensorManager;
36     private ArrayList<Sensor> sensorArrayList;
37     private TextView textViewCountDown;
38     private boolean isCapturing;
39     private File file;
40     private ArrayList<double[]> acceleroData;
41     private ArrayList<double[]> gyroData;
42     private long startCaptureTime;
43     private long runningTime;
44     private ArrayList<double[]> rotData;
45     private ArrayList<double[]> rotVecData;
46
47     @Override
48     protected void onCreate(Bundle savedInstanceState) {
49         super.onCreate(savedInstanceState);
50         this.sensorArrayList = new ArrayList<Sensor>();
51         setContentView(R.layout.activity_record_all_rawsensor_data);
52         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
53         textViewCountDown = (TextView) findViewById(R.id.textViewCountDown);
54         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
55             this.sensorArrayList.add(sensorGetter(Sensor.TYPE_ACCELEROMETER));
56             this.sensorArrayList.add(sensorGetter(Sensor.TYPE_GYROSCOPE));
57             this.sensorArrayList.add(sensorGetter(Sensor.TYPE_ROTATION_VECTOR));
58         }
59
60         int REQUEST_EXTERNAL_STORAGE = 1;
61         String[] PERMISSIONS_STORAGE = {
62             Manifest.permission.READ_EXTERNAL_STORAGE,
63             Manifest.permission.WRITE_EXTERNAL_STORAGE
64         };
65         int permission = ActivityCompat.checkSelfPermission(this, Manifest.permission.
66             WRITE_EXTERNAL_STORAGE);
67         if (permission != PackageManager.PERMISSION_GRANTED) {
68             ActivityCompat.requestPermissions(
69                 this,
70                 PERMISSIONS_STORAGE,
71                 REQUEST_EXTERNAL_STORAGE
72             );
73         }
74         isCapturing = false;
75         for (Sensor sensor :
76             sensorArrayList) {
77             mSensorManager.registerListener(this, sensor, SensorManagerSENSOR_DELAY_GAME);
78         }
79     }
80
81     @Override
82     protected void onResume() {
83         super.onResume();
84         for (Sensor sensor :
85             sensorArrayList) {
86             mSensorManager.registerListener(this, sensor, SensorManagerSENSOR_DELAY_GAME);
87         }
88     }
89
90     @Override
91     protected void onPause() {
92         super.onPause();
93         mSensorManager.unregisterListener(this);
94     }
95
96     @Override
97     public void onSensorChanged(SensorEvent event) {
98         Sensor sensor = event.sensor;
99         if (isCapturing) {
100             float[] copyData = new float[4];
101             System.arraycopy(event.values, 0, copyData, 1, 3);
102             copyData[0] = System.currentTimeMillis() - startCaptureTime;
103             if (sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
104                 this.acceleroData.add(convertFloatsToDoubles(copyData));
105             } else if (sensor.getType() == Sensor.TYPE_GYROSCOPE) {
106                 this.gyroData.add(convertFloatsToDoubles(copyData));
107             } else if (sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {
108                 copyData = new float[6];
109                 System.arraycopy(event.values, 0, copyData, 1, event.values.length);
110                 copyData[0] = System.currentTimeMillis() - startCaptureTime;
111                 this.rotData.add(convertFloatsToDoubles(copyData));
112                 double theta = (Math.acos(event.values[3]))*2;
113                 double x = event.values[0]/ Math.sin(theta/2);
114                 double y = event.values[1]/ Math.sin(theta/2);

```

```

115    double [] vecData = new double[5];
116    vecData[0] = copyData[0];
117    vecData[1] = x;
118    vecData[2] = y;
119    vecData[3] = z;
120    vecData[4] = theta;
121    this.rotVecData.add(vecData);
122 }
123 runningTime = (System.currentTimeMillis() - startCaptureTime);
124 this.textViewCountDown.setText("Time:" + runningTime / 1000 + "sec");
125 }
126
127
128 @Override
129 public void onAccuracyChanged(Sensor sensor, int accuracy) {
130 }
131
132
133 public void onStartButtonClicked(View view) throws InterruptedException {
134     Uri notification = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
135     Ringtone r = RingtoneManager.getRingtone(getApplicationContext(), notification);
136     acceleroData = new ArrayList<double[]>();
137     gyroData = new ArrayList<double[]>();
138     rotData = new ArrayList<double[]>();
139     rotVecData = new ArrayList<double[]>();
140     String time = DateFormat.format("MM-dd-yyyy-h|mm|ss-aa", System.currentTimeMillis()).toString();
141     File root = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS)
142         ) + File.separator + "DataSensors");
143     root.mkdirs();
144     file = new File(root, FILENAME + time + ".csv");
145     try {
146         file.createNewFile();
147     } catch (IOException e) {
148         e.printStackTrace();
149     }
150     new CountDownTimer(10000, 1000) {
151
152         public void onTick(long millisUntilFinished) {
153             textViewCountDown.setText("Count_Down=" + millisUntilFinished / 1000);
154         }
155
156         public void onFinish() {
157             textViewCountDown.setText("Capturing Data!");
158             Uri notification = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
159             Ringtone r = RingtoneManager.getRingtone(getApplicationContext(), notification);
160             r.play();
161             isCapturing = true;
162             getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
163             startCaptureTime = System.currentTimeMillis();
164         }
165     }.start();
166 }
167
168 private Sensor sensorGetter(int input) {
169
170     List<Sensor> sensorList = mSensorManager.getSensorList(input);
171     Sensor sensor = null;
172     for (int i = 0; i < sensorList.size(); i++) {
173         sensor = sensorList.get(i);
174     }
175     return sensor;
176 }
177
178 private void writeData(BufferedWriter bw, ArrayList<double[]> data, String title, String columnTitle)
179     throws IOException {
180     bw.write(title);
181     bw.newLine();
182     bw.write(columnTitle);
183     bw.newLine();
184     for (int i = 0; i < data.size(); i++) {
185         for (int j = 0; j < data.get(i).length; j++) {
186             bw.write(data.get(i)[j] + ",");
187         }
188         bw.newLine();
189     }
190     bw.newLine();
191 }
192
193 private double[] convertFloatsToDoubles(float[] input) {
194     if (input == null)
195     {
196         return null;
197     }
198     double[] output = new double[input.length];
199     for (int i = 0; i < input.length; i++)
200     {
201         output[i] = input[i];
202     }
203     return output;
204 }
205
206 public void onStopButtonClicked(View view) {
207     isCapturing = false;
208     getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
209     try {
210         BufferedWriter bw = new BufferedWriter(new FileWriter(this.file));
211         writeData(bw, acceleroData, "Accelerometer", "time,x,y,z");

```

```

212     writeData(bw, gyroData, "Gyroscope", "time,x,y,z");
213     writeData(bw, rotData, "Rotation_Vector", "time,x*sin(theta/2),y*sin(theta/2),z*sin(theta/2),
214         cos(theta/2),akurasi_(dalam_radians)_(-1_jika_tidak_ada)");
215     writeData(bw, rotVecData, "Rotation_Vector_by_Vector_and_Angle", "time,x,y,z,theta");
216     bw.newLine();
217     bw.write("Running_Time:" + runningTime + "ms");
218     bw.flush();
219     bw.close();
220     file.createNewFile();
221     textViewCountDown.setText("Data_Captured!_at_" + file.getAbsolutePath());
222     Log.d("File", "File_is_located_at_" + file.getAbsolutePath());
223 } catch (IOException e) {
224     e.printStackTrace();
225 }
226 }
227 }
```

Listing A.3: SensorAccelerometerActivity.java

```

1 package com.example.egaprianto.testingsensors;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.hardware.Sensor;
6 import android.hardware.SensorEvent;
7 import android.hardware.SensorEventListener;
8 import android.hardware.SensorManager;
9 import android.os.Build;
10 import android.os.Bundle;
11 import android.widget.TextView;
12
13 import java.util.List;
14
15 public class SensorAccelerometerActivity extends Activity implements SensorEventListener {
16     private SensorManager mSensorManager;
17     private Sensor mAccelSensor;
18     private TextView textViewXData;
19     private TextView textViewYData;
20     private TextView textViewZData;
21     private TextView textViewDebug;
22     private TextView textViewAccuracy;
23
24     @Override
25     public final void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_accelero_sensor);
28         textViewXData = (TextView) findViewById(R.id.textViewXData);
29         textViewYData = (TextView) findViewById(R.id.textViewYData);
30         textViewZData = (TextView) findViewById(R.id.textViewZData);
31         textViewDebug = (TextView) findViewById(R.id.textViewDebug);
32         textViewAccuracy = (TextView) findViewById(R.id.textViewAccuracy);
33         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
34         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM SANDWICH_MR1) {
35             mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
36             List<Sensor> accelerometerSensors = mSensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
37             mAccelSensor = null;
38             if (accelerometerSensors != null) {
39                 for (int i = 0; i < accelerometerSensors.size(); i++) {
40                     mAccelSensor = accelerometerSensors.get(i);
41                 }
42             }
43         }
44     }
45
46     @Override
47     public final void onAccuracyChanged(Sensor sensor, int accuracy) {
48     }
49
50     @Override
51     public final void onSensorChanged(SensorEvent event) {
52         float x = event.values[0];
53         float y = event.values[1];
54         float z = event.values[2];
55         textViewXData.setText("x=_" + x);
56         textViewYData.setText("y=_" + y);
57         textViewZData.setText("z=_" + z);
58     }
59
60     @Override
61     protected void onResume() {
62         super.onResume();
63         mSensorManager.registerListener(this, mAccelSensor, SensorManager.SENSOR_DELAY_NORMAL);
64     }
65
66     @Override
67     protected void onPause() {
68         super.onPause();
69         mSensorManager.unregisterListener(this);
70     }
71
72 }
73 }
```

Listing A.4: SensorGyroscopeActivity.java

```
1 package com.example.egaprianto.testingsensors;
```

```

2|
3 import android.app.Activity;
4 import android.content.Context;
5 import android.hardware.Sensor;
6 import android.hardware.SensorEvent;
7 import android.hardware.SensorEventListener;
8 import android.hardware.SensorManager;
9 import android.os.Build;
10 import android.os.Bundle;
11 import android.widget.TextView;
12
13 import java.util.List;
14
15 public class SensorGyroscopeActivity extends Activity implements SensorEventListener {
16     private SensorManager mSensorManager;
17     private Sensor mGryoSensor;
18     private TextView textViewXData;
19     private TextView textViewYData;
20     private TextView textViewZData;
21     private TextView textViewDebug;
22     private TextView textViewAccuracy;
23
24     @Override
25     public final void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_gyroscope_sensor);
28         textViewXData = (TextView) findViewById(R.id.textViewXData);
29         textViewYData = (TextView) findViewById(R.id.textViewYData);
30         textViewZData = (TextView) findViewById(R.id.textViewZData);
31         textViewDebug = (TextView) findViewById(R.id.textViewDebug);
32         textViewAccuracy = (TextView) findViewById(R.id.textViewAccuracy);
33         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
34         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
35             mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
36             List<Sensor> gyroscopeSensors = mSensorManager.getSensorList(Sensor.TYPE_GYROSCOPE);
37             mGryoSensor = null;
38             if (gyroscopeSensors != null) {
39                 for (int i = 0; i < gyroscopeSensors.size(); i++) {
40                     mGryoSensor = gyroscopeSensors.get(i);
41                 }
42             }
43         }
44     }
45
46     @Override
47     public final void onAccuracyChanged(Sensor sensor, int accuracy) {
48     }
49
50     @Override
51     public final void onSensorChanged(SensorEvent event) {
52         float x = event.values[0];
53         float y = event.values[1];
54         float z = event.values[2];
55         textViewXData.setText("x=" + x);
56         textViewYData.setText("y=" + y);
57         textViewZData.setText("z=" + z);
58     }
59
60     @Override
61     protected void onResume() {
62         super.onResume();
63         mSensorManager.registerListener(this, mGryoSensor, SensorManager.SENSOR_DELAY_NORMAL);
64     }
65
66     @Override
67     protected void onPause() {
68         super.onPause();
69         mSensorManager.unregisterListener(this);
70     }
71
72
73 }

```


LAMPIRAN B

KODE PROGRAM APLIKASI *GAME*

Listing B.1: GyroscopeChecker.cs

```
1  iżłusing System.Collections;
2  using System.Collections.Generic;
3  using TheNextFlow.UnityPlugins;
4  using UnityEngine;
5
6  public class GyroscopeChecker : MonoBehaviour {
7
8      // Use this for initialization
9      private void Awake()
10     {
11         //Debug.Log("init");
12
13         if (!SystemInfo.supportsGyroscope)
14         {
15             MobileNativePopups.OpenAlertDialog(
16                 "No Gyroscope", "Your device didn't have Gyroscope sensor, you can't play this game!",
17                 "OK", null);
18             Application.Quit();
19         }
20     }
21 }
```

Listing B.2: GameplayStatistic.cs

```
1  iżł
2  public class GameplayStatistic {
3
4      public static bool isAnswering = false;
5  }
```

Listing B.3: GameOverController.cs

```
1  iżłusing System;
2  using UnityEngine;
3  using UnityEngine.UI;
4
5  public class GameOverController : MonoBehaviour, IGvrGazeResponder {
6
7      public QuestionMessageController[] itemMessages;
8      public Image pointer;
9      public Animator animator;
10
11     public void OnGazeEnter()
12     {
13         //Do nothing
14     }
15
16     public void OnGazeExit()
17     {
18         //Do nothing
19     }
20
21     public void OnGazeTrigger()
22     {
23         //print("trigger");
24         restartGame();
25     }
26
27     public void restartGame()
28     {
29         //print("restart");
30         foreach(QuestionMessageController itemMessage in itemMessages)
31         {
32             itemMessage.resetAll();
33             animator.SetBool("poppedOut", false);
34             pointer.transform.Rotate(new Vector3(0,0,(-1*pointer.transform.rotation.eulerAngles.z)));
35         }
36     }
37
38     // Use this for initialization
39     void Start () { }
```

```

39 }
40
41 // Update is called once per frame
42 void Update ()
43 {
44     //print(Math.Abs(pointer.transform.rotation.eulerAngles.z));
45     //RectTransform gameOverTransform= this.GetComponent<RectTransform>();
46     //gameOverTransform.rotation.eulerAngles.Set(gameOverTransform.rotation.eulerAngles.x,
47         //gameOverTransform.rotation.eulerAngles.y,0);
48     if (isGameOver())
49     {
50         animator.SetBool("poppedOut", true);
51         haltAllGameplay ();
52     }
53 }
54 void LateUpdate()
55 {
56     GvrViewer.Instance.UpdateState();
57     if (GvrViewer.Instance.BackButtonPressed)
58     {
59         Application.Quit();
60     }
61 }
62 public bool isGameOver()
63 {
64     return Math.Abs(pointer.transform.rotation.eulerAngles.z) >= 59 && Math.Abs(pointer.transform.
65         rotation.eulerAngles.z) <= 309;
66 }
67 public void haltAllGameplay()
68 {
69     foreach (QuestionMessageController itemMessage in itemMessages)
70     {
71         itemMessage.stopWorking();
72     }
73 }

```

Listing B.4: QuestionMessageController.cs

```

1 izzf
2 using System;
3 using TheNextFlow.UnityPlugins;
4 using UnityEngine;
5 using UnityEngine.UI;
6
7 public class QuestionMessageController : MonoBehaviour, IGvrGazeResponder, HeadMotionListener
8 {
9     public Canvas messageCanvas;
10    public Canvas notificationCanvas;
11    public String[] questions;
12    public bool[] answers;
13    public Image pointer;
14    private Animator animatorMessage;
15    private Animator animatorNotification;
16    private Text messageText;
17    private bool startCounting;
18    private bool isDetecting;
19    private bool clickable;
20    private float timeAvailable;
21    private MotionRecorder mMotionRecorder;
22    private int selectedIndexQuestions;
23
24    public void OnGazeEnter()
25    {
26    }
27
28    public void OnGazeExit()
29    {
30    }
31
32    public void OnGazeTrigger()
33    {
34        if (clickable && !GameplayStatistic.isAnswering)
35        {
36            Input.gyro.enabled = true;
37            Input.gyro.updateInterval = 0.00002F;
38            mMotionRecorder = new MotionRecorder();
39            mMotionRecorder.addHeadMotionListener(this);
40            isDetecting = true;
41            selectedIndexQuestions = (int)(UnityEngine.Random.Range(0.0f, 0.9999f) * questions.Length);
42            messageText.text = questions[selectedIndexQuestions];
43            animatorMessage.SetBool("poppedOut", !animatorMessage.GetBool("poppedOut"));
44            animatorNotification.SetBool("poppedOut", !animatorNotification.GetBool("poppedOut"));
45            clickable = false;
46            GameplayStatistic.isAnswering = true;
47        }
48    }
49
50    // Use this for initialization
51    void Start () {
52        timeAvailable = UnityEngine.Random.Range(5.0f, 10.0f);
53        clickable = false;
54        animatorMessage = messageCanvas.GetComponentInChildren<Animator>();
55        animatorNotification = notificationCanvas.GetComponentInChildren<Animator>();
56        messageText = messageCanvas.GetComponentInChildren<Text>();
57        startCounting = true;
58    }

```

```

59 // Update is called once per frame
60 void Update() {
61     if (startCounting)
62     {
63         timeAvailable -= Time.deltaTime;
64     }
65     if (timeAvailable < 0)
66     {
67         startQuestion();
68     }
69     if (isDetecting)
70     {
71         this.mMotionRecorder.inputGryoData(Input.gyro.rotationRate.y, Input.gyro.rotationRate.x);
72         if (Input.GetKeyDown("d")) debugOnNodMotionDetected();
73         if (Input.GetKeyDown("f")) debugOnShookMotionDetected();
74     }
75 }
76
77 public void resetAll()
78 {
79     print("resetting object");
80     animatorMessage.SetBool("poppedOut", false);
81     animatorNotification.SetBool("poppedOut", false);
82     startCounting = true;
83     isDetecting = false;
84     clickable = false;
85     timeAvailable = UnityEngine.Random.Range(5.0f, 10.0f);
86     mMotionRecorder = null;
87 }
88
89 public void stopWorking()
90 {
91     isDetecting = false;
92     clickable = false;
93 }
94
95 public void debugOnNodMotionDetected()
96 {
97     this.onMotionDetected(Motion.NOD);
98 }
99
100 public void debugOnShookMotionDetected()
101 {
102     this.onMotionDetected(Motion.SHOOK);
103 }
104
105 public void startQuestion()
106 {
107     startCounting = false;
108     animatorNotification.SetBool("poppedOut", !animatorNotification.GetBool("poppedOut"));
109     clickable = true;
110     timeAvailable = UnityEngine.Random.Range(5.0f, 10.0f);
111 }
112
113 public void onMotionDetected(Motion motion)
114 {
115     if (isDetecting)
116     {
117         if (motion == Motion.NOD)
118         {
119             messageText.text = "YES";
120             if (answers[selectedIndexQuestions])
121             {
122                 pointer.transform.Rotate(new Vector3(0, 0, 10));
123             }
124             else
125             {
126                 pointer.transform.Rotate(new Vector3(0, 0, -10));
127             }
128         }
129         else
130         {
131             messageText.text = "NO";
132             if (!answers[selectedIndexQuestions])
133             {
134                 pointer.transform.Rotate(new Vector3(0, 0, 10));
135             }
136             else
137             {
138                 pointer.transform.Rotate(new Vector3(0, 0, -10));
139             }
140         }
141         clickable = false;
142         startCounting = true;
143         GameplayStatistic.isAnswering = false;
144         animatorMessage.SetBool("poppedOut", false);
145         Input.gyro.enabled = false;
146         isDetecting = false;
147         mMotionRecorder.removeHeadMotionListener(this);
148         mMotionRecorder = null;
149     }
150 }
151 }
```

Listing B.5: Axis.cs

```

1 //public enum Axis
2 {
```

```

3|     X,
4|     Y
5| }
```

Listing B.6: Detector.cs

```

1| if
2| public abstract class Detector {
3|
4|     protected bool mightHaveActivites;
5|
6|     protected Pulse[] listPulseToDetect;
7|     protected double limitTime;
8|     protected float limitSpeed;
9|     protected double limitSimpangan;
10|
11|    public Detector(double limitTime, float limitSpeed, double limitSimpangan)
12|    {
13|        this.limitTime = limitTime;
14|        this.limitSpeed = limitSpeed;
15|        this.limitSimpangan = limitSimpangan;
16|    }
17|
18|    public bool isMightHaveActivites()
19|    {
20|        return mightHaveActivites;
21|    }
22|
23|
24|    abstract public bool addPulse(Pulse yPulse);
25| }
```

Listing B.7: HeadMotionListener.cs

```

1| if
2| public interface HeadMotionListener
3| {
4|     void onMotionDetected(Motion motion);
5| }
```

Listing B.8: Motion.cs

```

1| if
2| public enum Motion
3| {
4|     NOD,
5|     SHOOK
5| }
```

Listing B.9: MotionRecorder.cs

```

1| if
2| using System.Collections.Generic;
3| using UnityEngine;
4|
5| public class MotionRecorder {
6|
7|     private PulseFactory pulseFactoryXAxis;
8|     private PulseFactory pulseFactoryYAxis;
9|     private Detector nodDetector;
10|    private Detector shookDetector;
11|
12|    private LinkedList<HeadMotionListener> listListener;
13|
14|    public MotionRecorder()
15|    {
16|        listListener = new LinkedList<HeadMotionListener>();
17|        pulseFactoryXAxis = new PulseFactory(Axis.X);
18|        pulseFactoryYAxis = new PulseFactory(Axis.Y);
19|        nodDetector = new NodDetector(700, 2, 0.25);
20|        shookDetector = new ShookDetector(700, 2, 0.25);
21|    }
22|
23|    public void addHeadMotionListener(HeadMotionListener listener)
24|    {
25|        listListener.AddFirst(listener);
26|    }
27|
28|    public void removeHeadMotionListener(HeadMotionListener listener)
29|    {
30|        listListener.Remove(listener);
31|    }
32|
33|    public void inputGryoData(float x, float y)
34|    {
35|        Pulse xPulse = pulseFactoryXAxis.inputData(x);
36|        Pulse yPulse = pulseFactoryYAxis.inputData(y);
37|        if (xPulse != null)
38|        {
39|            Debug.Log("PULSE");
40|            bool shookDetected = shookDetector.addPulse(xPulse);
41|            if (shookDetected && !nodDetector
42|                .isMightHaveActivites())
43|                // terdeteksi geleng tanpa ada pergerakan pada sumbu y
44|                notifyAllMotionListener(Motion.SHOOK);
```

```

45         //Log.d("MotionDetectedPulsesX", shookDetector.toString());
46     }
47   }
48   if (yPulse != null)
49   {
50     bool nodDetected = nodDetector.addPulse(yPulse);
51     if (nodDetected && !shookDetector
52       .isMightHaveActivites())
53     { // terdeteksi gelengan tanpa ada pergerakan pada sumbu x
54       notifyAllMotionListener(Motion.NOD);
55     }
56   }
57 }
58
59 public void notifyAllMotionListener(Motion motion)
60 {
61   foreach (HeadMotionListener motionListener in listListener)
62   {
63     motionListener.onMotionDetected(motion);
64   }
65 }
66 }
```

Listing B.10: NodDetector.cs

```

1  using System;
2
3  public class NodDetector : Detector
4  {
5    /*
6    private double limitTime;// in milliseconds
7    private float limitSpeed = 2;
8    private double limitSimpangan = 0.25;
9    */
10   public NodDetector(double limitTime, float limitSpeed, double limitSimpangan) : base(limitTime,
11     limitSpeed, limitSimpangan)
12   {
13
14     this.mightHaveActivites = false;
15     this.listPulseToDetect = new Pulse[2];
16     this.listPulseToDetect[0] = PulseFactory.STANDARD_PULSE; //firstPulse
17     this.listPulseToDetect[1] = PulseFactory.STANDARD_PULSE; //lastPulse
18
19   public override bool addPulse(Pulse yPulse)
20   {
21     bool result = false;
22     this.listPulseToDetect[0] = yPulse;
23     if (Math.Abs(this.listPulseToDetect[0].getPeak()) > limitSpeed)
24     {
25       mightHaveActivites = true;
26       bool isPassLimitSimpangan = Math.Abs(this.listPulseToDetect[0].getSimpangan()) >
27         limitSimpangan
28       && Math.Abs(listPulseToDetect[1].getSimpangan()) > limitSimpangan;
29       bool isPassLimitTimeRange =
30         listPulseToDetect[0].getRangeTime() < limitTime && listPulseToDetect[1].getRangeTime() <
31         limitTime;
32       bool isDifferentTypePulse = this.listPulseToDetect[0].getType() != listPulseToDetect[1].
33         getType();
34       result = isPassLimitSimpangan && isPassLimitTimeRange && isDifferentTypePulse;
35     }
36     else
37     {
38       if (Math.Abs(listPulseToDetect[1].getPeak()) < limitSpeed)
39       {
40         mightHaveActivites = false;
41       }
42     }
43     this.listPulseToDetect[1] = this.listPulseToDetect[0];
44     return result;
45   }
46 }
```

Listing B.11: Pulse.cs

```

1  using System;
2
3  public class Pulse
4  {
5    private double simpangan;
6    private double rangeTime;
7    private double startTime;
8    private double endTime;
9    private float peak;
10   private PulseType type;
11   private Axis axis;
12
13   public Pulse(double simpangan, double startTime, double endTime, float peak, PulseType type,
14     Axis axis)
15   {
16     this.simpangan = simpangan;
17     this.startTime = startTime;
18     this.endTime = endTime;
19     this.rangeTime = endTime - startTime;
20     this.peak = peak;
21     this.type = type;
22     this.axis = axis;
```

```

23    }
24
25    public double getSimpanan()
26    {
27        return simpanan;
28    }
29
30    public double getRangeTime()
31    {
32        return rangeTime;
33    }
34
35    public double getStartTime()
36    {
37        return startTime;
38    }
39
40    public double getEndTime()
41    {
42        return endTime;
43    }
44
45    public float getPeak()
46    {
47        return peak;
48    }
49
50    public PulseType getType()
51    {
52        return type;
53    }
54
55    public Axis getAxis()
56    {
57        return axis;
58    }
59}

```

Listing B.12: PulseFactory.cs

```

1  using System;
2
3  public class PulseFactory {
4
5      private long lastTime;
6      private float lastData;
7      private double simpanan;
8      private long rangeTime;
9      private double startTime;
10     private double endTime;
11     private float peak;
12     private PulseType currentPulseType;
13     private Axis axis;
14
15     public static Pulse STANDARD_PULSE = new Pulse(0, 0, 0, 0, PulseType.HILL, Axis.X);
16
17     public PulseFactory(Axis axis)
18     {
19         this.axis = axis;
20     }
21
22     private void resetAttributes(PulseType type)
23     {
24         this.simpanan = 0;
25         this.peak = 0;
26         this.currentPulseType = type;
27     }
28
29     private Pulse finishPulse(float data, double currentTime)
30     {
31         this.endTime = ((-lastData / (data - lastData)) * (currentTime - lastTime)) + lastTime;
32         double areaBeforeIntersect = ((endTime - lastTime) / 1000) * lastData / 2;
33         this.simpanan += areaBeforeIntersect;
34         Pulse newPulse = new Pulse(this.simpanan, this.startTime, this.endTime, this.peak,
35             this.currentPulseType,
36             this.axis);
37         this.startTime = endTime;
38         return newPulse;
39     }
40
41     public Pulse inputData(float data)
42     {
43         Pulse result = null;
44         long currentTime = DateTime.Now.Millisecond;
45
46         if (currentPulseType == PulseType.HILL)
47         {
48             if (peak < data)
49             {
50                 peak = data;
51             }
52             if (data < 0)
53             { //if values intersect data=0
54                 result = finishPulse(data, currentTime);
55                 resetAttributes(PulseType.VALLEY);
56             }
57             else
58             {

```

```

59         this.simpangan += ((lastData + data) / 1000) * (currentTime - lastTime) / 2;
60     }
61     else if (currentPulseType == PulseType.VALLEY)
62     {
63         if (peak > data)
64         {
65             peak = data;
66         }
67         if (data > 0)
68         {//if values intersect data=0
69             result = finishPulse(data, currentTime);
70             resetAttributes(PulseType.HILL);
71         }
72         else
73         {
74             this.simpangan += ((lastData + data) / 1000) * (currentTime - lastTime) / 2;
75         }
76     }
77 }
78 lastData = data;
79 lastTime = currentTime;
80 return result;
81 }
82 }
83 }
```

Listing B.13: PulseType.cs

```

1 izfpublic enum PulseType
2 {
3     HILL,
4     VALLEY,
5     INIT
6 }
```

Listing B.14: ShookDetector.cs

```

1 izfusing System;
2 public class ShookDetector :Detector
3 {
4     /*
5      * private double limitTime;//in millisecons
6      * private float limitSpeed = 2;
7      * private double limitSimpangan = 0.25;
8      */
9     public ShookDetector(double limitTime, float limitSpeed, double limitSimpangan) : base(limitTime,
10                         limitSpeed, limitSimpangan)
11     {
12         mightHaveActivites = false;
13         listPulseToDetect = new Pulse[3];
14         listPulseToDetect[0] = PulseFactory.STANDARD_PULSE;
15         listPulseToDetect[1] = PulseFactory.STANDARD_PULSE;
16         listPulseToDetect[2] = PulseFactory.STANDARD_PULSE;
17     }
18
19     public override bool addPulse(Pulse xPulse)
20     {
21         bool result = false;
22         listPulseToDetect[2] = listPulseToDetect[1];
23         listPulseToDetect[1] = listPulseToDetect[0];
24         listPulseToDetect[0] = xPulse;//currentPulse
25         if (Math.Abs(listPulseToDetect[0].getPeak()) > limitSpeed)
26         {
27             mightHaveActivites = true;
28             bool isPassLimitSimpangan = Math.Abs(listPulseToDetect[0].getSimpangan()) > limitSimpangan
29             && Math.Abs(listPulseToDetect[1].getSimpangan()) > limitSimpangan
30             && Math.Abs(listPulseToDetect[2].getSimpangan()) > limitSimpangan;
31             bool isPassLimitTimeBetween =
32             (listPulseToDetect[1].getStartTime() - listPulseToDetect[2].getEndTime() == 0) &&
33             listPulseToDetect[0].getStartTime() - listPulseToDetect[1].getEndTime() == 0);
34             bool isPassLimitTimeRange =
35             listPulseToDetect[0].getRangeTime() < limitTime && listPulseToDetect[1].getRangeTime() <
36             limitTime
37             && listPulseToDetect[2].getRangeTime() < limitTime;
38             bool isDifferentTypePulse =
39             (listPulseToDetect[1].getType() != listPulseToDetect[2].getType()) && (listPulseToDetect
40             [0].getType() != listPulseToDetect[1]
41             .getType());
42             result = isPassLimitSimpangan && isPassLimitTimeBetween && isDifferentTypePulse
43             && isPassLimitTimeRange;
44         }
45         else
46         {
47             if (Math.Abs(listPulseToDetect[2].getPeak()) < limitSpeed
48                 && Math.Abs(listPulseToDetect[1].getPeak()) < limitSpeed)
49             {
50                 mightHaveActivites = false;
51             }
52         }
53     }
54 }
```