

SKRIPSI

**ALGORITMA 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 ALGORITHM USING GOOGLE
CARDBOARD**



EGA PRIANTO

NPM: 2013730047

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2016**

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 are 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

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
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	2
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	9
2.2 Google VR SDK	16
2.2.1 API Audio	17
2.2.2 API Base	18
2.3 Teori Kuaternion	20
2.3.1 Struktur Ajabar	20
2.3.2 <i>Aljabar Kuaternion</i> dan Operasi-operasi pada Kuaternion	22
2.4 Unity Game Engine	24
2.4.1 Struktur Hierarki GameObject	24
2.4.2 Prefabs	25
2.4.3 Scene	25
2.4.4 GameObject	25
2.4.5 Transformasi Rotasi dan Orientasi	27
2.5 Google VR SDK for Unity	28
3 ANALISIS	29
3.1 Analisis Aplikasi Sejenis	29
3.2 Perekaman Data Sensor	30
3.2.1 Perekaman Grafik Sensor <i>Accelerometer</i>	32
3.2.2 Perekaman Grafik Sensor <i>Gyroscope</i>	34
3.2.3 Perekaman Grafik Sensor <i>Rotation Vector</i>	36
3.3 Analisis Data Sensor untuk Mendeteksi Gerakan Kepala	37
3.4 Analisis Metode Pendekripsi Gerakan Kepala	39
3.4.1 Algoritma Mendekripsi Gerakan Mengangguk	39

3.4.2	Algoritma Mendeteksi Gerakan Menggeleng	42
4	PERANCANGAN	45
4.1	Perancangan Kelas Algoritma Pendekripsi Gerakan Kepala	45
4.2	Perancangan Hierarki GameObject pada Unity	49
4.3	Perancangan <i>Gameplay</i> dan Perancangan Antarmuka	49
5	IMPLEMENTASI DAN PENGUJIAN	53
5.1	Implementasi	53
5.1.1	Lingkungan Implementasi	53
5.1.2	Hasil Implementasi	53
5.2	Pengujian	55
5.2.1	Pengujian Fungsional	55
5.2.2	Pengujian Eksperimental	58
5.3	Masalah yang Dihadapi pada Saat Implementasi	61
6	KESIMPULAN DAN SARAN	63
6.1	Kesimpulan	63
6.2	Saran	63
DAFTAR REFERENSI		65
A	KODE PROGRAM APLIKASI UNTUK ANALISIS	67
B	KODE PROGRAM APLIKASI <i>Game</i>	73

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	8
2.4	Sistem koordinat (relatif dengan perangkatnya) yang digunakan oleh Sensor API	11
2.5	Sistem koordinat sensor rotasi vektor terhadap Bumi	17
2.6	Contoh perputaran dengan bilangan kompleks	21
2.7	Contoh perputaran tiga puluh derajat dengan bilangan kompleks	22
2.8	Right-hand rule dalam <i>cross product</i> vektor	23
2.9	Contoh Hierarchy Parenting.	25
2.10	Contoh <i>scene</i> kosong.	26
2.11	Contoh penggunaan komponen pada GameObject.	26
2.12	Contoh komponen pada GameObject kubus.	27
3.1	<i>Screenshot</i> task yang diberikan aplikasi InMind VR.	29
3.2	<i>Screenshot</i> aplikasi InMind VR ketika permainan berlangsung.	30
3.3	<i>Screenshot</i> aplikasi InMind VR ketika meminta pengguna untuk mengangguk jika telah siap.	31
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.	32
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.	33
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.	34
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.	35
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.	35
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.	36
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.	37
3.11	Dua buah rotasi yang identik dengan nilai kuaternion yang berbeda.	38

3.12 Deskripsi grafik pada saat pengguna menggeleng. Yang ditunjuk oleh panah ber-warna hitam adalah selang waktu yang terjadi saat pengguna melawan arah gerakan kepala.	38
4.1 Diagram Kelas Algoritma Pendekripsi Gerakan Kepala.	45
4.2 Perancangan hierarki Game Object	50
4.3 Meteran pada permainan.	51
4.4 Desain ruangan untuk permainan yang akan dibuat.	51
4.5 Desain User Interface	52
5.1 Tampilan ketika aplikasi baru dimulai.	54
5.2 Tampilan ketika perangkat tidak memiliki sensor Gyroscope.	54
5.3 Tampilan ketika notifikasi muncul.	55
5.4 Tampilan ketika <i>pop-up</i> pertanyaan muncul.	56
5.5 Tampilan setelah pengguna mengangguk.	56
5.6 Tampilan setelah pengguna menggeleng.	57
5.7 Tampilan setelah permainan usai.	57
5.8 Pengujian oleh Ricky Setiawan.	59
5.9 Pengujian oleh Harseto Pandityo.	59
5.10 Pengujian oleh Herfan Heryandi.	60
5.11 Pengujian oleh Kevin Antonius.	60
5.12 Pengujian oleh Antonius Kurnia.	61

DAFTAR TABEL

2.1 Tipe-tipe Sensor pada Android	10
5.1 Tabel Perangkat yang digunakan	58

1

BAB 1

2

PENDAHULUAN

3

1.1 Latar Belakang

4 *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]

5 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 *template*-nya di situs 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. Perusahaan-perusahaan tersebut diantaranya BoboVR, ShineconVR, VRBox, dan lain-lain.

6 Pada gawai Google Cardboard cara pengguna memberikan masukkan 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*.

7 Skripsi ini akan membuat aplikasi untuk menambahkan cara baru memberikan masukkan 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 masukkan baru kepada program VR. Jenis masukkan yang diberikan kepada komputer hanya ya(mengangguk) atau tidak(menggeleng).

8 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 1.2 Rumusan Masalah

- 2 • Bagaimana cara menampilkan grafik data yang diambil dari sensor-sensor pada *smartphone*?
3
- 4 • Bagaimana cara mendeteksi gerakan kepala dari data yang didapat dari sensor-sensor
5 pada *smartphone*?
6

6 1.3 Tujuan

- 7 • Mengetahui cara untuk menampilkan grafik data dari sensor-sensor pada *smartphone*.
8
- 9 • Mengetahui cara mendeteksi gerakan kepala dari data yang didapat dari sensor-sensor
pada *smartphone*.
10

10 1.4 Batasan Masalah

11 Penelitian ini dibuat berdasarkan batasan-batasan sebagai berikut:

- 12 1. Program pertama yang akan dibuat dalam skripsi ini hanya akan digunakan untuk
13 membantu dalam menganalisis sensor.
14
- 15 2. Program kedua yang akan dibuat hanya dapat melakukan pendeksi gerakan kepala
khusus untuk mengangguk dan menggeleng saja.
16

16 1.5 Metode Penelitian

17 Berikut adalah metode penelitian yang digunakan dalam penelitian ini:

- 18 1. Melakukan studi literatur tentang Android SDK, Google VR SDK, Quaternion, *Sensor Fusion*, dan algoritma *Head Motion Detection*.
19
- 20 2. Merancang dan membuat aplikasi untuk menampilkan grafik sensor-sensor pada *smartphone* Android.
21
- 22 3. Menganalisis aplikasi-aplikasi sejenis.
23
- 24 4. Merekam dan menganalisis grafik dari sensor-sensor pada *smartphone* ketika meng-
angguk dan menggeleng.
25
- 26 5. Menganalisis metode pendeksi gerakan kepala.
27
- 28 6. Merancang aplikasi untuk mendeksi gerakan kepala
29
- 30 7. Mengimplementasi algoritma pendeksi gerakan kepala ke aplikasi *virtual reality*.
31

28 1.6 Sistematika Penulisan

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

- 31 1. Bab 1: Pendahuluan, yaitu membahas mengenai gambaran umum penelitian ini. Berisi
32 tenang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian,
33 dan sistematika penulisan.
34
- 35 2. Bab 2: Dasar Teori, yaitu membahas teori-teori yang mendukung berjalannya peneli-
tian ini. Berisi tentang Android SDK, Google VR SDK, Quaternion.

- 1 3. Bab 3: Analisis, yaitu membahas mengenai analisa masalah. Berisi tentang analisis
2 aplikasi-aplikasi sejenis, perekaman data sensor, analisis grafik dari sensor-sensor pada
3 smartphone ketika mengangguk dan menggeleng, analisis metode pendekripsi gerakan
4 kepala.
- 5 4. Bab 4: Perancangan, yaitu membahas mengenai perancangan aplikasi VR yang dapat
6 mendeteksi gerakan menggeleng dan mengangguk.
- 7 5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dan pe-
8 ngujian aplikasi yang telah dilakukan. Berisi tentang implementasi dan hasil pengujian
9 aplikasi.
- 10 6. Bab 6: Kesimpulan dan Saran, yaitu membahas mengenai kesimpulan dari keseluruhan
11 penelitian ini dan saran-saran yang dapat diberikan untuk penelitian selanjutnya.

BAB 2

DASAR TEORI

³ Pada bab ini akan dijelaskan dasar-dasar teori mengenai Android SDK, Google VR SDK, Quaternion,
⁴ *Sensor Fusion*, dan algoritma *Head Motion Detection*.

5 2.1 Android SDK

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

11 2.1.1 Struktur File Android Studio Project

12 Dalam membuat aplikasi pada perangkat Android, dibutuhkan Android SDK.^[8] Android
13 SDK(*software development kit*) adalah kumpulan *source code*, *development tools*, *emulator*,
14 dan semua *libraries* untuk membuat suatu aplikasi untuk *platform* Android.^[7] IDE (*inte-*
15 *grated development environment*) yang resmi untuk Android SDK adalah Android Studio.
16 Android Studio dapat di download di halaman situs web Google Developer, sekaligus de-
17 ngan Android SDKnya. **Struktur File Android Studio Project**
18 Pada saat **project** baru telah dibuat, Android Studio akan membuatkan folder-folder stan-
19 dar seperti pada Gambar 2.1.

20 2.1.2 Membuat User Interface

²¹ Pada subbab ini akan dijelaskan bagaimana membuat layout di XML termasuk *text field*
²² dan *button*[8]

²³ Hierarki *Graphical User Interface* (GUI) untuk Aplikasi Android

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

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

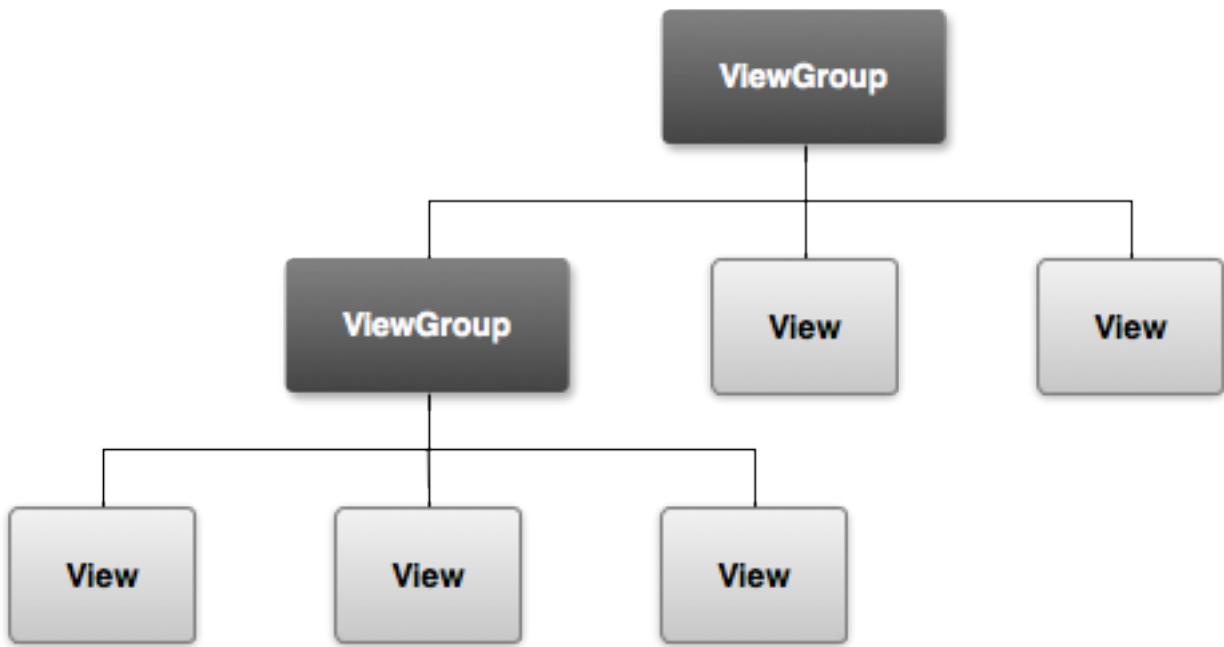
31 Attribut-attribut Objek View

³² Pada subbab ini akan dijelaskan attribut-attribut object View yang digunakan dalam membuat GUI pada file activity_main.xml

- 34 • **android:id** Attribut ini merupakan pengidentifikasi dari suatu view. Attribut ini
35 dapat digunakan untuk menjadi referensi object dari kode aplikasi seperti membaca

```
module
├── build ... Folder yang mengandung
    file-file dari hasil
    pembangunan project.
├── libs ... Folder yang mengandung
    libraries privat.
└── src ... Folder yang mengandung
    semua kode dan file
    sumber untuk suatu modul.
    ├── androidTest ... Folder yang mengandung
        kode untuk mengetes yang
        berjalan di perangkat
        Android.
    └── main ... Folder yang mengandung
        file-file kode dan file
        sumber inti dari suatu
        module.
        ├── AndroidManifest.xml ... File yang mendeskripsikan
            sifat dari aplikasi dan
            setiap komponennya.
        ├── java ... Folder yang mengandung
            kode-kode java.
        ├── jni ... Folder yang mengandung
            kode-kode yang
            menggunakan Java Native
            Interface (JNI).
        ├── gen ... Folder yang mengandung
            file-file java yang
            dihasilkan oleh Android
            Studio.
        ├── res ... Folder yang mengandung
            file-file sumber untuk
            aplikasi, seperti file
            drawable, layout, dan UI
            String.
        └── assets ... Folder yang mengandung
            file yang akan di kompile
            menjadi file .apk.
    └── build.gradle ... File yang mendefinisikan
        konfigurasi modul untuk
        proses build.
└── build.gradle ... File yang mendefinisikan
    konfigurasi proses
    build untuk project
    yang berlaku untuk semua
    modul.
```

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.

1 dan memanipulasi objek tersebut (Akan dijelaskan lebih lanjut pada subbab 2.1.3).
 2 Tanda '@' dibutuhkan ketika mereferensi object dari suatu XML. Tanda '@' tersebut
 3 diikuti dengan tipe (id pada kasus ini), slash, dan nama (edit_message pada Listing
 4 2.2). Tanda tambah (+) sebelum tipe hanya dibutuhkan jika ingin mendefinisikan
 5 resource ID untuk pertama kalinya.

- 6 • **android:layout_width** dan **android:layout_height** Attribut ini digunakan untuk
 7 mendefinisikan panjang dan lebar dari suatu objek View. Daripada menggunakan besar
 8 spesifik untuk panjang dan lebarnya, lebih baik menggunakan "wrap_content"
 9 yang menspesifikasi viewnya hanya akan sebesar yang dibutuhkan untuk memuat
 10 konten-konten dari View. Jika menggunakan "match_parent" pada kasus Listing 2.2
 11 View akan memenuhi layar, karena besarnya akan mengikuti besar dari paretnya Li-
 12 nearLayout.
- 13 • **android:hint** Attribut ini merupakan *default string* untuk ditampilkan ketika objek
 14 View kosong. Daripada menggunakan *hard-coded string* sebagai *nilai* untuk ditam-
 15 pilkan, *value "@string/edit_message"* mereferensi ke sumber string pada file yang
 16 berbeda. Karena mereferensi ke sumber konkret, maka tidak dibutuhkan tanda tam-
 17 bah (+). Nilai string ini akan disimpan pada file Strings.xml yang ditunjukkan pada
 18 Listing 2.1.

Listing 2.1: Contoh kode pada string.xml

```

19 <resources>
20   <string name="app_name">MyFirstAndroidApp</string>
21   <string name="edit_message">Ini adalah hint</string>
22   <string name="button_send">Send</string>
23 </resources>
  
```

- 24 • **android:onClick** Attribut ini akan memberitahu *system* untuk memanggil method
 25 yang sesuai namanya (contoh pada Listing 2.2 adalah **sendMessage()**) di Activity
 26 ketika pengguna melakukan klik pada button tersebut. Agar *system* dapat memanggil
 27 method yang tepat, method tersebut harus memenuhi kriteria berikut.
 28 – Access Modifier haruslah *public*.

- Harus *void return valuenya*.
 - Mempunyai View sebagai parameter satu-satunya. View ini akan diisi dengan View yang di klik.

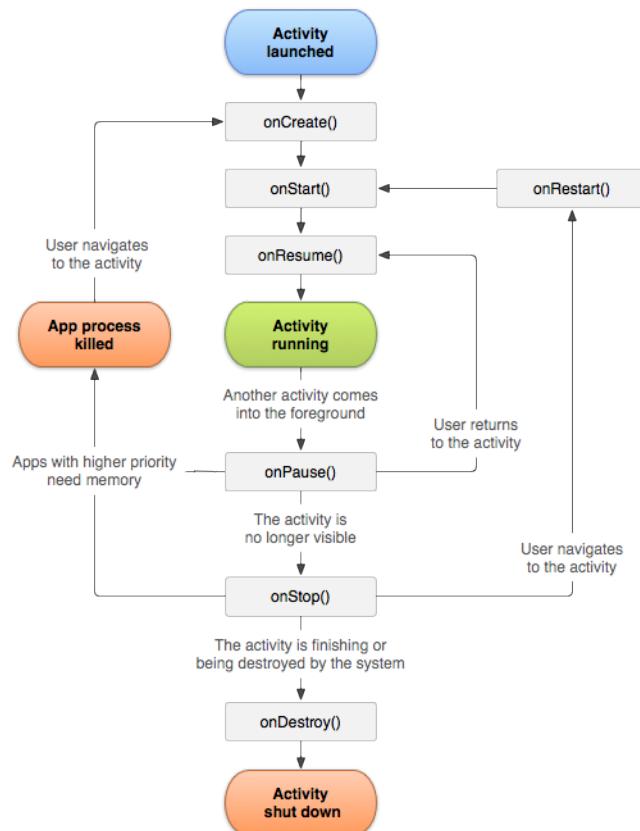
Listing 2.2: Contoh kode file XML pada folder layout

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

21 2.1.3 Activity

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

27 Activity Lifecycle



Gambar 2.3: *State diagram* siklus Activity

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

1 aktif. Activity sebelumnya akan tetap berada di bawah stack, dan tidak akan muncul lagi
2 sampai Activity yang baru berakhir.

3 Activity didasari dari empat kondisi:

- 4 • Jika Activity berada di latar depan pada layar, Activity tersebut sedang aktif.
- 5 • Jika Activity sudah tidak terfokuskan tetapi masih dapat terlihat, Activity tersebut
6 sedang berhenti sementara. Pada kondisi ini Activity tersebut masih berjalan, tapi
7 bisa diberhentikan ketika sistem berada dalam situasi kekurangan memori.
- 8 • Jika suatu Activity benar-benar dihalangi oleh Activity lain, Activity tersebut telah
9 berhenti. Activity tersebut akan tetap mengingat seluruh keadaan dan infomasi anggo-
10 ta tetapi, Activity tersebut tidak lagi terlihat oleh pengguna jadi tampilan jendelanya
11 akan tersembunyi dan seringkali akan diberhentikan Activitynya ketika system mem-
12 butuhkan memori.
- 13 • Jika suatu Activity sedang berhenti sementara atau berhenti total, sistem dapat mem-
14 buang Activity dari memory dengan cara menanyakan kepada pengguna untuk mem-
15 berhentikannya atau langsung diberhentikan oleh sistem. Jika Activity tersebut ditam-
16 pilkan lagi kepada pengguna, Activity tersebut harus memulai dari awal dan kembali
17 ke keadaan sebelumnya.

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

- 22 • *Entire lifetime* terjadi diantara pemanggilan pertama pada onCreate(Bundle) sam-
23 pai ke satu pemanggilan akhir onDestroy(). Suatu Activity akan melakukan semua
24 persiapan pada kondisi umum pada method onCreate(), dan melepaskan seluruh sisa
25 pemrosesan pada method onDestroy().
- 26 • *Visible lifetime* terjadi antara pemanggilan onStart() sampai pemanggilan yang sesuai
27 pada onStop(). Pada tahap ini pengguna dapat melihat Activity pada layar meskipun
28 tidak berada pada *foreground* dan berinteraksi dengan pengguna.
- 29 • *Foreground lifetime* terjadi antara pemanggilan onResume() sampai ke sa-
30 tu pemanggilan akhir onDestroy(). Pada tahap ini Activity berada di depan semua
31 Activity lainnya dan sedang berinteraksi dengan user.

32 2.1.4 Android Sensor Framework

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

- 38 • **Sensor gerak** Sensor-sensor ini mengukur akselerasi dan rotasi pada tiga sumbu.
39 Kategori sensor ini meliputi *accelerometers*, sensor gravitasi, *gyroscope*, dan *rotation
40 vector*.
- 41 • **Sensor keadaan lingkungan** Sensor-sensor ini mengukur berbagai keadaan ling-
42 kungan seperti suhu udara, tekanan, pencahayaan, dan kelembaban. Kategori sensor
43 ini termasuk barimeter, fotometer dan termometer.
- 44 • **Sensor posisi** Sensor-sensor ini mengukur posisi perangkat. Kategori sensor ini me-
45 liputi sensor orientasi dan magnetometer.

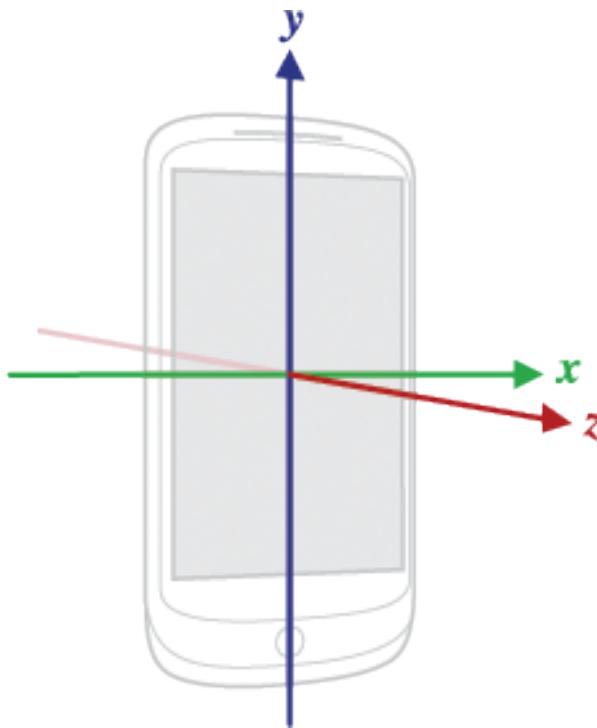
- 1 Android Sensor Framework membantu developers untuk mengakses berbagai jenis sensor.
 2 Beberapa sensor berbasis perangkat keras dan beberapa sensor berbasis perangkat lunak.
 3 Sensor berbasis perangkat keras mendapatkan data dengan langsung mengukur sifat ling-
 4 kungan tertentu, seperti percepatan,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

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(goncangan, 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(goncangan, 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.

9 Sistem Koordinat Sensor

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



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

- 1 • Accelerometer
- 2 • Sensor Gravitasi
- 3 • Gyroscope
- 4 • Sensor Percepatan Linear
- 5 • Sensor Medan Geomagnetik

6 Koordinat sistem yang sumbunya tidak tertukar ketika orientasi perangkat berubah. Sis-
7 tem koordinat sensor tidak pernah berubah seiring perangkatnya bergerak. Dalam aplikasi
8 android tidak dapat diassumsikan bahwa standar orientasi perangkat android adalah *portra-
9 it*. Kebanyakan perangkat *Tablet* standar orientasinya adalah *landscape*. Sistem koordinat
10 sensor selalu di dasarkan pada orientasi dasar dari suatu perangkat android.

11 Mengidentifikasi Sensor dan Kapabilitas Sensor

12 Android Sensor Framework menyediakan beberapa method yang dapat membuat developer
13 mudah untuk menentukan sensor mana yang akan digunakan. APInya juga dapat menye-
14 diakan method yang memungkinkan pengguna menentukan kapabilitas masing-masing
15 sensor, seperti jangkauan maksimum, resolusi, dan kebutuhan dayanya. Untuk mengidentifi-
16 kasi sensor-sensor yang ada pada perangkat, hal pertama yang perlu dilakukan adalah men-
17 dapatkan referensi sensor tersebut. Untuk mendapatkan referensi tersebut, dapat dilakukan
18 dengan membuat instansiasi dari kelas SensorManager dan memanggil method getSystemService()
19 dan memasukkan isi Parameternya dengan SENSOR_SERVICE. Contohnya pada
20 Listing 2.3.

Listing 2.3: Contoh inisialisasi kelas SensorManager

```
21 | private SensorManager mSensorManager;
22 | ...
23 | mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

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

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

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

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

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

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

```
15 | private SensorManager mSensorManager;  

16 | ...  

17 | mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  

18 | if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){  

19 |     // Sukses! Perangkat ini memiliki sensor magnetometer.  

20 | }  

21 | else {  

22 |     // Gagal! Perangkat ini tidak memiliki sensor magnetometer.  

23 | }
```

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

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

```
28 | private SensorManager mSensorManager;  

29 | private Sensor mSensor;  

30 | ...  

31 |  

32 | mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  

33 | mSensor = null;  

34 |  

35 | if (mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY) != null){  

36 |     List<Sensor> gravSensors = mSensorManager.getSensorList(Sensor.TYPE_GRAVITY);  

37 |     for(int i=0; i<gravSensors.size(); i++) {  

38 |         if ((gravSensors.get(i).getVendor().contains("Google Inc.")) &&  

39 |             (gravSensors.get(i).getVersion() == 3)){  

40 |             // menggunakan sensor gravitasi versi 3.  

41 |             mSensor = gravSensors.get(i);  

42 |         }  

43 |     }  

44 | }  

45 | if (mSensor == null){  

46 |     // Use the accelerometer.  

47 |     if (mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null){  

48 |         mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);  

49 |     }  

50 |     else{  

51 |         // Tidak ada sensor gravitasi dan sensor accelerometer!  

52 |     }  

53 | }  

54 | }
```

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

59 Memonitor Nilai Sensor

60 Untuk memonitor data mentah dari sensor, dibutuhkan untuk mengimplement dua buah
 61 method callback yang berada pada interface `SensorEventListener`. Kedua method tersebut

1 adalah onAccuracyChanged(Sensor sensor, int accuracy) dan onSensorChanged(SensorEvent
 2 event). Sistem android akan memanggil kedua method ini ketika terjadi salah satu kondisi
 3 ini:

4 ● **Perubahan akurasi sensor.**

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

8 ● **Sensor memberitahu adanya nilai baru.**

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

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

Listing 2.7: Contoh memonitor data mentah pada sensor cahaya

```
18 public class SensorActivity extends Activity implements SensorEventListener {
19     private SensorManager mSensorManager;
20     private Sensor mLight;
21
22     @Override
23     public final void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.main);
26
27         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
28         mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
29     }
30
31     @Override
32     public final void onAccuracyChanged(Sensor sensor, int accuracy) {
33         // Hal yang perlu dilakukan aplikasi ketika akurasinya berubah.
34     }
35
36     @Override
37     public final void onSensorChanged(SensorEvent event) {
38         // Sensor cahaya akan mengembalikan 1 nilai saja.
39         // Banyak sensor lain yang akan mengembalikan lebih dari 1 nilai.
40         float lux = event.values[0];
41         // Hal yang perlu dilakukan ketika ada perubahan data.
42     }
43
44     @Override
45     protected void onResume() {
46         super.onResume();
47         mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
48     }
49
50     @Override
51     protected void onPause() {
52         super.onPause();
53         mSensorManager.unregisterListener(this);
54     }
55 }
```

56 Pada method onSensorChanged(SensorEvent event), struktur nilai-nilai yang dikembalikan
 57 akan dijelaskan pada subbab 2.1.4. Pada method onResume(), ada pemanggilan method re-
 58 gisterListener(). Method registerListener() ini berguna untuk menspesifikasi waktu delay
 59 pada pemanggilan onSensorChanged(). Untuk menspesifikasi delaynya dapat menggu-
 60 nakan konstanta yang ada pada kelas SensorManager. Konstanta-konstanta tersebut adalah
 61 SENSOR_DELAY_NORMAL (200.000 mikrodetik),SENSOR_DELAY_GAME (20.000 mik-
 62 rodetik),SENSOR_DELAY_UI (60.000 mikrodetik), atau SENSOR_DELAY_FASTEST
 63 (0 mikrodetik). Pengaturan dasar untuk waktu delay ini menggunakan konstanta SEN-
 64 SOR_DELAY_NORMAL. Perlu diperhatikan juga pemesanan sensor ketika activity sedang
 65 di berhentikan sementara maupun dilanjutkan kembali. Sistem tidak boleh tetap merekam
 66 sensor ketika activity tidak aktif. Hal ini diperlukan karena pada saat perangkat android
 67 menggunakan sensor, perangkat akan menggunakan tenaga yang banyak. Akan lebih baik
 68 jika penggunaan sensor diberhentikan ketika activitynya sudah tidak lagi digunakan.

1 Struktur Nilai yang Dikembalikan oleh Sensor

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

7

8 • TYPE_ACCELEROMETER

9 Semua nilai didefinisikan sebagai satuan m/s^2

- 10 – values[0]: Percepatan yang terjadi pada sumbu x dikali -1
 11 – values[1]: Percepatan yang terjadi pada sumbu y dikali -1
 12 – 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(*Fs*) yang terjadi pada sensor menggunakan relasi berikut:

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

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

```
21 public void onSensorChanged(SensorEvent event)
22 {
23     // alpha dikalkulasikan sebagai t / (t + dT)
24     // dengan t adalah low-pass filter's time-constant
25     // dan dT, rata-rata event tersampaikan
26
27     final float alpha = 0.8;
28
29     gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
30     gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
31     gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];
32
33     linear_acceleration[0] = event.values[0] - gravity[0];
34     linear_acceleration[1] = event.values[1] - gravity[1];
35     linear_acceleration[2] = event.values[2] - gravity[2];
36 }
```

Low-pass filter dapat diimplementasikan pada Listing 2.8

39 • TYPE_MAGNETIC_FIELD

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

43 • TYPE_GYROSCOPE

44 Sensor ini mengukur rata-rata perputaran pada perangkat yang berputar di sumbu
 45 X,Y, dan Z dalam satuan radians/second. Sistem koordinat yang digunakan sama
 46 dengan sistem koordinat pada sensor percepatan(Accelerometer). Jika perangkat ber-
 47 putar 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

```

9   private static final float NS2S = 1.0f / 1000000000.0f;
10  private final float[] deltaRotationVector = new float[4]();
11  private float timestamp;
12
13  public void onSensorChanged(SensorEvent event) {
14      // Pada tahapan ini delta rotasi akan dikalikan dengan rotasi saat ini
15      // setelah mengomputasinya dari data gyro.
16      if (timestamp != 0) {
17          final float dT = (event.timestamp - timestamp) * NS2S;
18          // Sumbu dari rotasi, masih belum di normalisasi.
19          float axisX = event.values[0];
20          float axisY = event.values[1];
21          float axisZ = event.values[2];
22
23          // Menghitung percepatan angular
24          float omegaMagnitude = sqrt(axisX*axisX + axisY*axisY + axisZ*axisZ);
25
26          // Normalisasi rotasi vektor jika cukup besar untuk mendapatkan sumbunya.
27          if (omegaMagnitude > EPSILON) {
28              axisX /= omegaMagnitude;
29              axisY /= omegaMagnitude;
30              axisZ /= omegaMagnitude;
31          }
32
33          // Integrate around this axis with the angular speed by the time step
34          // in order to get a delta rotation from this sample over the time step
35          // We will convert this axis-angle representation of the delta rotation
36          // into a quaternion before turning it into the rotation matrix.
37          float thetaOverTwo = omegaMagnitude * dT / 2.0f;
38          float sinThetaOverTwo = sin(thetaOverTwo);
39          float cosThetaOverTwo = cos(thetaOverTwo);
40          deltaRotationVector[0] = sinThetaOverTwo * axisX;
41          deltaRotationVector[1] = sinThetaOverTwo * axisY;
42          deltaRotationVector[2] = sinThetaOverTwo * axisZ;
43          deltaRotationVector[3] = cosThetaOverTwo;
44      }
45      timestamp = event.timestamp;
46      float[] deltaRotationMatrix = new float[9];
47      SensorManager.getRotationMatrixFromVector(deltaRotationMatrix, deltaRotationVector
48          );
49      // User code should concatenate the delta rotation we computed with the current
50      // rotation
51      // in order to get the updated rotation.
52      // rotationCurrent = rotationCurrent * deltaRotationMatrix;
53  }
54

```

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:

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

1
2
3 • **TYPE_ORIENTATION**

4 Semua nilai adalah sudut dalam derajat.

- 5 – values[0]: Azimuth, sudut diantara arah magnetik utara dengan sumbu y, sekitar
6 sumbu z (0 sampai 359). 0 = Utara, 90 = Timur, 180 = Selatan, 270 = Barat
7 – values[1]: Pitch, rotasi sekitar sumbu x (-180 sampai 180), dengan nilai positif
8 ketika sumbu x bergerak menuju sumbu y.
9 – values[2]: Roll, perputaran sekitar sumbu y (-90 sampai 90) pada kondisi portrait,
10 sensor akan bernilai 0. Pada kondisi landscape ke kanan sensor akan bernilai 90
11 dan sebaliknya yaitu kondisi landscape ke kiri sensor akan bernilai -90.

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

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

16
17 • **TYPE_ROTATION_VECTOR**

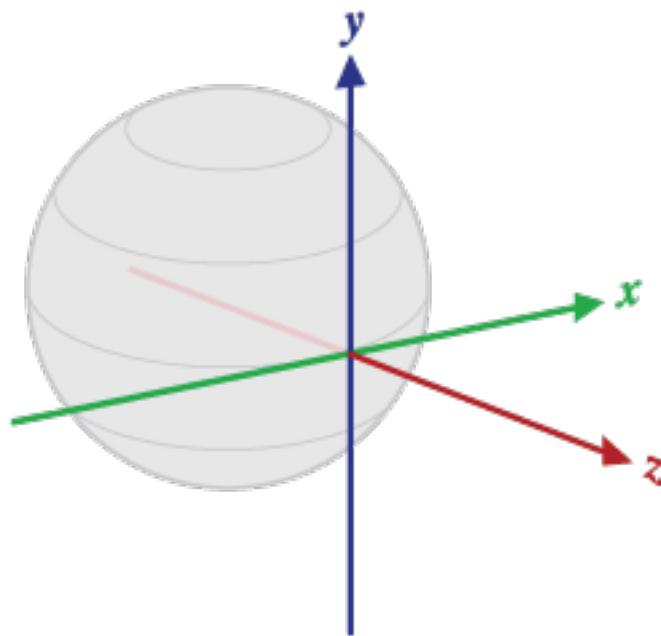
18 Sensor ini merepresentasikan orientasi perangkat dengan kombinasi dari sumbu dan
19 sudut. Perangkat akan di putar sebesar sudut θ mengelilingi sumbu x, y, z . Ti-
20 ga elemen dari vektor rotasi adalah $(x \sin(\frac{\theta}{2}), y \sin(\frac{\theta}{2}), z \sin(\frac{\theta}{2}))$, sehingga besarnya
21 vektor rotasi sama dengan $\sin(\frac{\theta}{2})$, dan arah vektor rotasi sama dengan sumbu rota-
22 si. Tiga elemen dari vektor rotasi sama dengan tiga komponen terakhir pada unit
23 quaternion($\cos(\frac{\theta}{2}), x \sin(\frac{\theta}{2}), y \sin(\frac{\theta}{2}), z \sin(\frac{\theta}{2})$) yang dijelaskan pada subbab 2.3. Ele-
24 men dari vektor rotasi tak memiliki satuan. Sistem koordinat yang digunakan sama
25 dengan sistem koordinat yang digunakan pada sensor percepatan. Referensi koordinat
26 didefinisikan sebagai basis orthonormal, yaitu:

- 27 – X didefinisikan sebagai perkalian dot product $\mathbf{Y} \cdot \mathbf{Z}$
28 – Y merupakan tangensial ke tanan pada lokasi perangkat saat ini dan menunjuk
29 ke arah utara.
30 – Z menghadap ke langit dan tegak lurus dengan tanah. Untuk lebih jelasnya dapat
31 dilihat pada Gambar 2.5
32 – values[0]: $x \sin(\frac{\theta}{2})$
33 – values[1]: $y \sin(\frac{\theta}{2})$
34 – values[2]: $z \sin(\frac{\theta}{2})$
35 – values[3]: $\cos(\frac{\theta}{2})$
36 – values[4]: Perkiraan akurasi (dalam radians) (-1 jika tidak tersedia)

37 **2.2 Google VR SDK**

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

- 40 • **Binocular rendering:** Fitur untuk tampilan layar terpisah untuk masing-masing
41 dalam pandangan VR.



Gambar 2.5: Sistem koordinat sensor rotasi vektor terhadap Bumi

- 1 • **Spatial audio:** Fitur untuk mengeluarkan suara yang datang dari daerah-daerah
2 tertentu dari dunia VR.
- 3 • **Head movement tracking:** Fitur untuk mendapatkan memperbaharui pengelihatan
4 dunia VR yang sesuai dengan gerakan kepala pengguna.
- 5 • **Trigger input:** Fitur untuk memberikan input pada dunia VR dengan menekan
6 tombol.

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

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

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

- 17 • API audio: API untuk mengimplementasikan ***Spatial Audio*** (Metode untuk mens-
18 pasialisasikan sumber suara dalam ruang tiga dimensi).
- 19 • API base: API untuk fondasi dari suatu aplikasi Google VR.

20 2.2.1 API Audio

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

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

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

5
6 API ini juga dapat memutarkan suara secara *stereo*. Kelas **GvrAudioEngine** memiliki tiga
7 buah *nested class* yaitu:

8 ● **GvrAudioEngine.DistanceRolloffModel**: Kelas ini mendefinisikan konstanta-konstanta
9 yang merepresentasikan perbedaan jarak dari efek model-model rolloff.

10 ● **GvrAudioEngine.MaterialName**: Kelas ini mendefinisikan konstanta-konstanta yang
11 merepresentasikan bahan permukaan ruangan untuk disesuaikan dengan efek suara
12 pada suatu ruangan.

13 ● **GvrAudioEngine.RenderingMode**: Kelas ini mendefinisikan konstanta-konstanta untuk
14 menyesuaikan dengan mode rendering. Semakin baik kualitas renderin akan semakin besar
15 penggunaan CPU(Central Processing Unit).

16

2.2.2 API Base

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

21 ● **AndroidCompat**

22 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:

23 – `setSustainedPerformanceMode(Activity activity, boolean enabled)`:

24 Method ini digunakan untuk mengubah window android ke mode performa secara berkelanjutan.

25 – `public static void setVrModeEnabled (Activity activity, boolean enabled)`:

26 Mengatur pengaturan yang tepat untuk "mode VR" pada suatu Activity. Method ini tidak digunakan karena hanya dapat digunakan pada Android N+.

27 – `public static boolean trySetVrModeEnabled (Activity activity, boolean enabled)`:

28 Method ini kegunaanya sama dengan method **setVrModeEnabled (Activity activity, boolean enabled)**, namun mengembalikan boolean true jika berhasil dan sebaliknya.

29 ● **Eye**

30 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).

31 ● **GvrActivity**

32 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

1 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).

4 • GvrView

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

- 9 – GvrView.StereoRenderer: Interface untuk perenderan detil stereoskopik secara
10 abstrak oleh perender.
- 11 – GvrView.Renderer: Interface untuk mesin yang kompleks yang membutuhkan
12 untuk menangai semua detil perenderan.

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

16 – **public void onNewFrame(HeadTransform headTransform)**

17 method ini terpanggil ketika Frame baru akan digambar. Method ini memungkinkan
18 untuk membedakan antara perenderan pandangan mata dan frame-frame
19 yang berbeda. Setiap operasi per-frame harus tidak spesifik pada satu tampilan
20 saja.

21 – **public abstract void onDrawEye (Eye eye)**

22 Method ini meminta untuk menggambar suatu konten dari sudut pandang mata.

23 – **public abstract void onFinishFrame (Viewport viewport)**

24 Method ini dipanggil ketika suatu frame telah selesai. Dengan pemanggilan ini,
25 konten frame telah di gambar dan jika koreksi distorsi diaktifkan, koreksi distorsi
26 akan diterapkan. Setiap perenderan pada tahap ini relatif terhadap seluruh
27 permukaan, tidak terhadap satu pandangan mata tunggal.

28 – **public abstract void onRendererShutdown ()**

29 Method ini dipanggil ketika thread perender sedang menutup. Melepaskan sumber
30 GL(Graphics Library) dan sedang melakukan penutupan operasi pada thread
31 perender. Dipanggil hanya jika sebelumnya ada pemanggilan method onSurfaceCreated.

33 – **public abstract void onSurfaceChanged (int width, int height)**

34 Dipanggil ketika ada perubahan dimensi permukaan. Semua nilai adalah relatif
35 ke ukuran yang dibutuhkan untuk merender sebuah mata.

36 – **public abstract void onSurfaceCreated (EGLConfig config)**

37 Method ini dipanggil ketika suatu permukaan dibangun atau dibangun ulang.

38 • HeadTransform

39 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:

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

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

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

48 Method ini digunakan untuk mendapatkan kuaternion yang merepresentasikan
49 rotasi kepala.

- 1 • Viewport

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

3 2.3 Teori Kuaternion

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

10 2.3.1 Struktur Ajabar

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

15 Bilangan Kompleks

16 [10] Bilangan kompleks adalah bilangan yang merupakan gabungan dari bilangan imajiner
 17 dengan bilangan riil. Notasi umum dari bilangan kompleks adalah :

$$a + bi \quad (2.1)$$

18 Pada notasi 2.1 bilangan a dengan b merupakan bilangan riil, dan i merupakan bilangan
 19 imajiner tertentu yang memiliki sifat $i^2 = -1$. Bilangan kompleks juga dapat beroperasi
 20 dengan bilangan kompleks lainnya seperti penjumlahan, perkalian, pengurangan, dan pem-
 21 bagian. Berikut adalah contoh-contoh operasi pada bilangan kompleks 2.1 dengan bilangan
 22 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.

$$(a + ib) + (c + id) = (a + c) + i(b + d) = \quad (2.2)$$

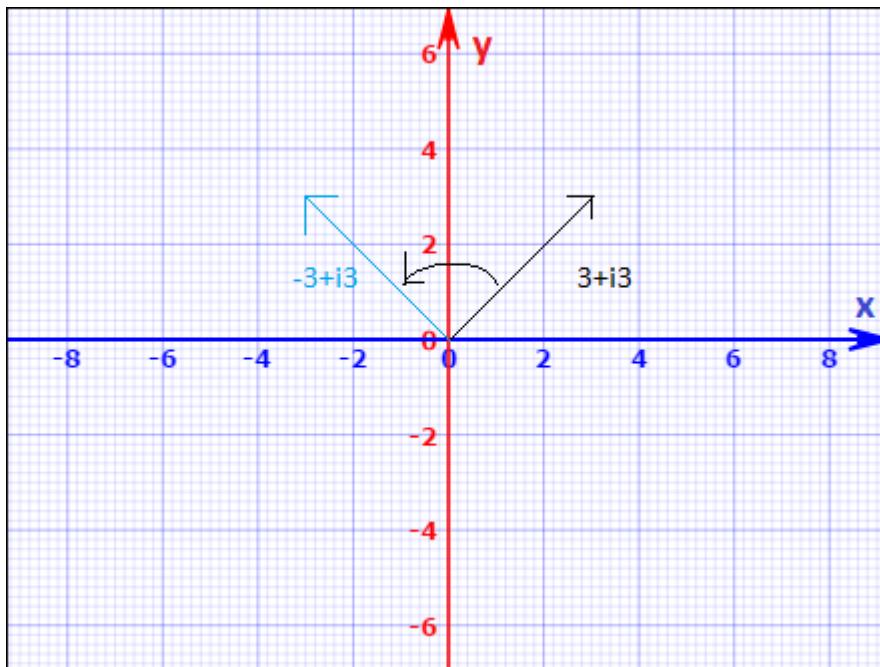
$$(c + id) + (a + ib) = (c + a) + i(d + b)$$

- 1 Suatu bilangan dapat dikatakan konjugasi kompleks dari suatu bilangan kompleks jika
 2 nilai bilangan riilnya sama, tetapi nilai bilangan imajinernya berlawanan dengan nilai pada
 3 bilangan kompleks tersebut. Maka konjugasi kompleks dari bilangan kompleks 2.1 adalah
 4 $a - bi$.

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



Gambar 2.6: Contoh perputaran dengan bilangan kompleks

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:

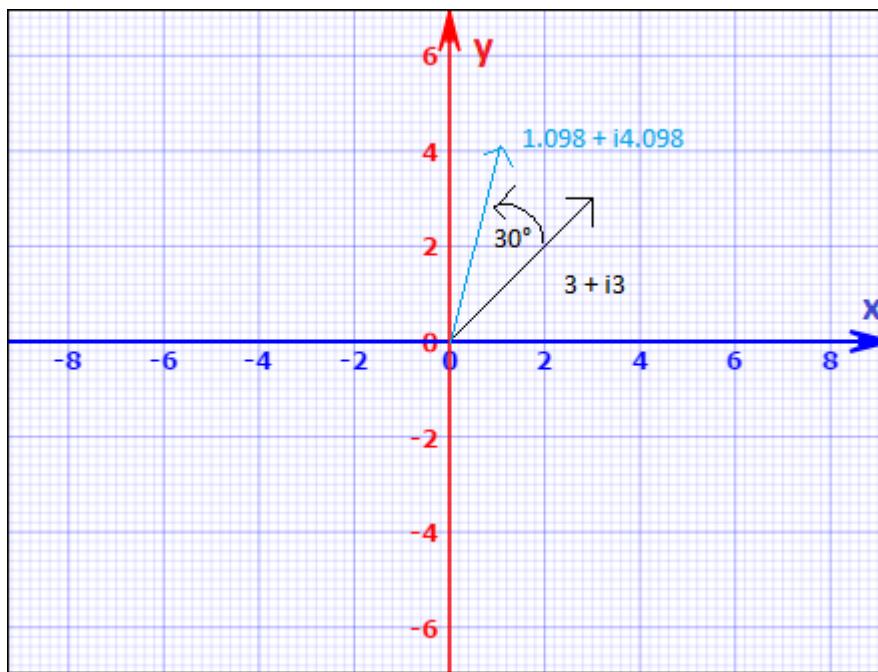
$$v' = v \times (\cos \theta + i \sin \theta)$$

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

- 1 arah jarum jam menggunakan konsep diatas, akan menghasilkan vektor berikut:

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

- 2 Dari persamaan tersebut dapat divisualisasikan pada Gambar 2.7.



Gambar 2.7: Contoh perputaran tiga puluh derajat dengan bilangan kompleks

3 2.3.2 Aljabar Kuaternion dan Operasi-operasi pada Kuaternion

- 4 Kuaternion ditemukan oleh ahli matematika dan astronomi Inggris, William Rowan Hamilton, dengan memperpanjang aritmatika dari bilangan kompleks.[\[10\]](#) Dari penemuan tersebut 5 William Rowan Hamilton menemukan bahwa dia tidak hanya membutuhkan bilangan imajiner i saja untuk melakukan rotasi pada ruang tiga dimensi. Dia menemukan bahwa dia 6 juga membutuhkan tiga komponen imajiner lainnya yaitu i, j dan k . Persamaan umum 7 Kuaternion memiliki empat bilangan riil atau skalar. Persamaan tersebut adalah 8

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

Ketiga komponen tersebut memiliki relasi sebagai berikut:

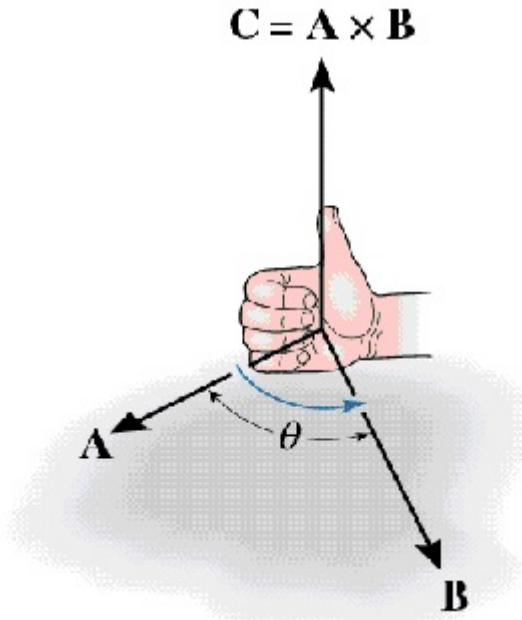
$$i^2 = j^2 = k^2 = ijk = -1$$

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

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

¹

- ² 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 .



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

⁵⁶

Seperti pada bilangan kompleks, kuaternion juga memiliki konjugasinya. Konjugasi kuaternion ini akan digunakan untuk melakukan operasi rotasi tiga dimensi(Akan dijelaskan pada subbab 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:

$$q = q_0 - iq_1 - jq_2 - kz_3$$

7 Operasi pada Kuaternion

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

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

dan,

$$q = q_0 + iq_1 + jq_2 + kz_3$$

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:

$$v' = qvq^*$$

¹ dengan,

- ² • $v = 1 + x_v i + y_v j + z_v k$
- ³ • $q = q_0 + iq_1 + jq_2 + kq_3$
- ⁴ • $q^* = q_0 - iq_1 - jq_2 - kq_3$
- ⁵ • $v' = 1 + x_{v'} i + y_{v'} j + z_{v'} k$

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:

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

⁶ dengan vektor f (x_f, y_f, z_f) merupakan sumbu perputaran dan θ merupakan besar sudut
⁷ putar berlawanan arah dengan jarum jam.

⁸ 2.4 Unity Game Engine

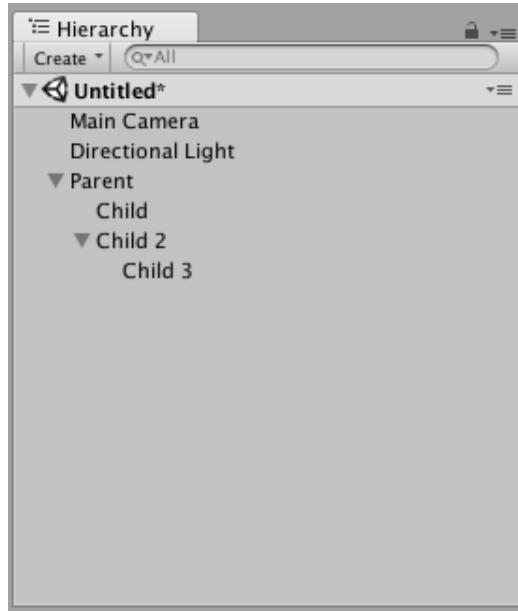
⁹ Unity merupakan salah satu *Game Engine* dalam pembuatan permainan dua dimensi atau
¹⁰ tiga dimensi.^[11] Unity merupakan *Game Engine multi-platform* sehingga permainan yang
¹¹ dibuat menggunakan Unity akan dapat berjalan diberbagai macam perangkat seperti *smartphone*,
¹² komputer, *console*, dan lain-lain. *Game Engine* ini menggunakan bahasa C# atau
¹³ Javascript.

¹⁴ Game Engine ini dapat diperoleh secara gratis, yaitu dengan menggunakan Unity Personal
¹⁵ Edition. Game Engine edisi Personal ini tidak akan berlaku jika penggunaan unity
¹⁶ mendapatkan kotor tahunan sebesar USD 100.000. Jika pendapatan kotor ta-
¹⁷ hunan sudah melebihi USD 100.000, pengguna unity personal harus segera mengganti edisi
¹⁸ Unity menjadi Unity Plus, Unity Pro, atau Unity Enterprise. Jika menggunakan Unity
¹⁹ Personal, *splashscreen* dan logo *icon* pada permainan yang dibuat tidak diganti maupun
²⁰ dihapus.

²¹ 2.4.1 Struktur Hierarki GameObject

²² Hierarki pada unity merupakan kumpulan dari GameObject. ^[12] GameObject akan dijelaskan
²³ lebih lanjut pada subbab 2.4.4. Beberapa dari GameObject pada hierarki ini merupakan
²⁴ instansi dari file *assets*. Setiap *scene* akan memiliki hierarki masing-masing. Setiap ob-
²⁵ jek yang di tambahkan pada suatu *scene*(dijelaskan pada subbab 2.4.3) akan muncul pada
²⁶ hierarki juga.

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



Gambar 2.9: Contoh Hierarchy Parenting.

6 2.4.2 Prefabs

7 Unity memiliki suatu jenis assets yang dinamakan Prefab. Prefab ini adalah suatu Ga-
 8 meObject yang disimpan menjadi assets. Prefab berperan sebagai contoh/*blueprint* dari
 9 suatu GameObject. Prefab ini juga dapat dibentuk dari GameObject yang telah dibuat dan
 10 menjadi assets, sehingga dapat digunakan pada proyek lain. Prefab juga data di ubah dan
 11 dimodif sehingga menjadi sesuai dengan yang diinginkan.

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

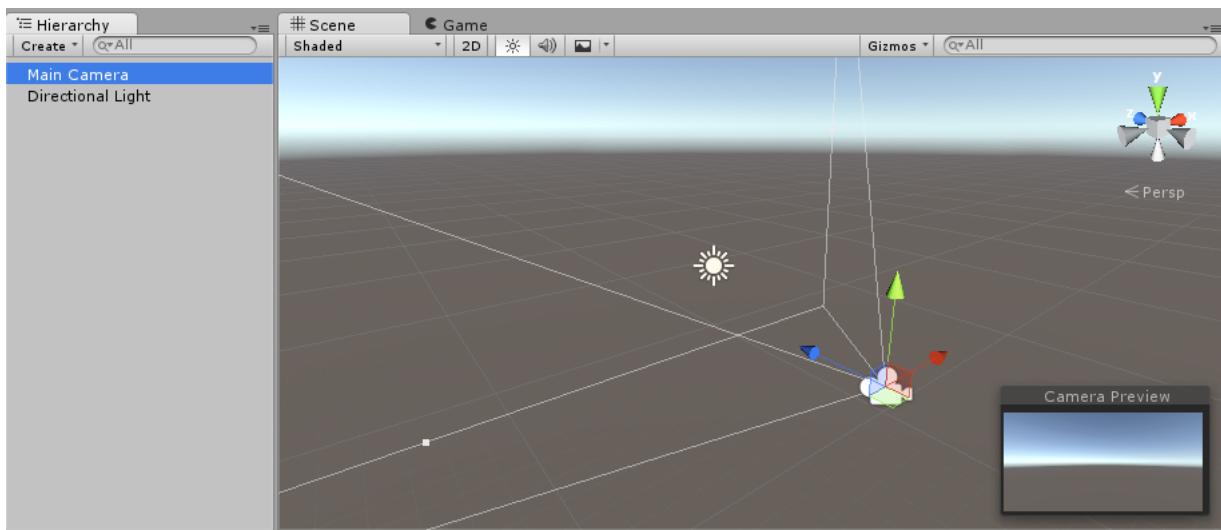
15 2.4.3 Scene

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

20 Pada saat pembuatan proyek pertama, unity akan membuatkan suatu *scene* baru. *Scene*
 21 tersebut merupakan *scene default*. *Scene* tersebut hanya akan diberikan objek-objek *default*
 22 saja seperti kamera dan cahaya. Kamera yang akan diberikan antara dua buah jenis yaitu
 23 *orthographic camera* atau *perspective camera*, proyek 2D akan diberikan *orthographic camera*
 24 dan proyek 3D akan diberikan *perspective camera*. Contoh dari *scene default* pada proyek
 25 3D ditunjukkan pada Gambar 2.10.

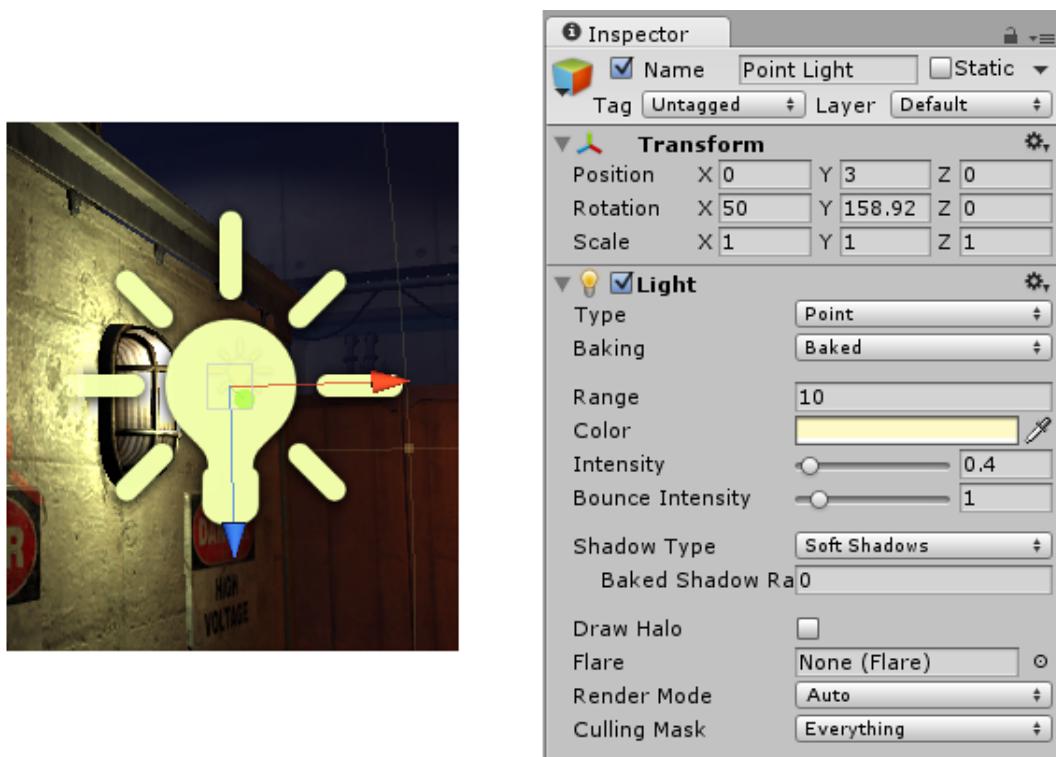
26 2.4.4 GameObject

27 GameObject pada unity yang merupakan representasi karakter, barang, dan pemandang-
 28 an. GameObject tidak dapat melakukan apa pun dengan sendirinya, tetapi GameObject



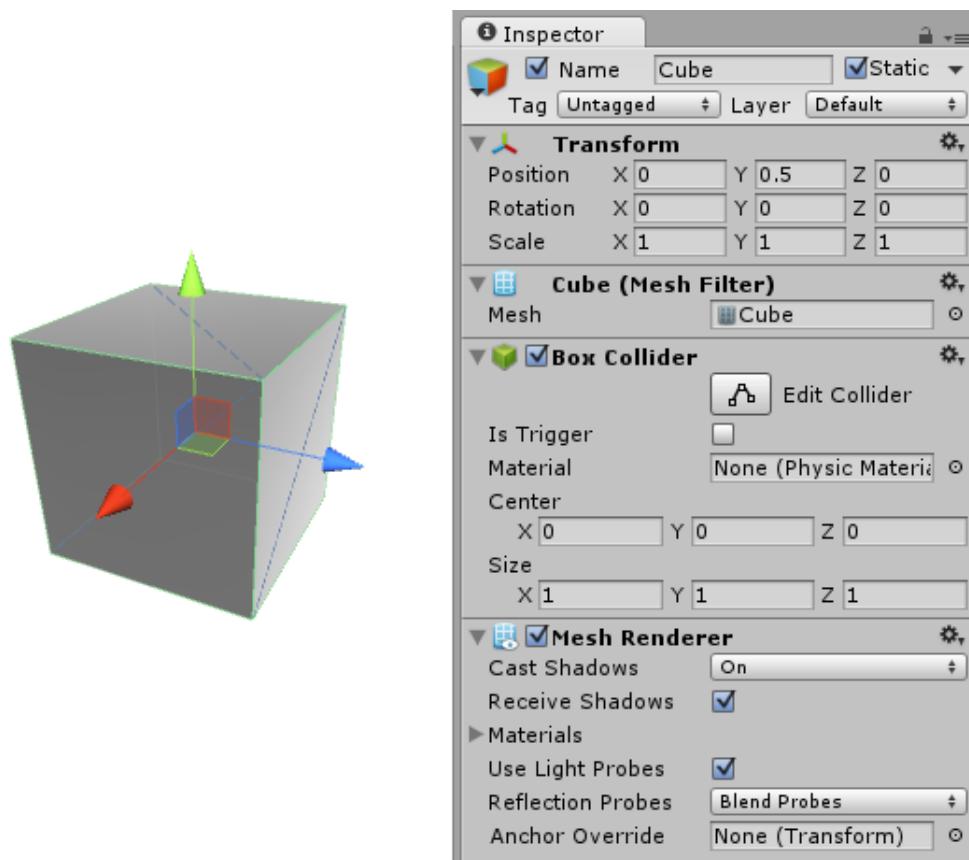
Gambar 2.10: Contoh *scene* kosong.

- 1 melakukan sesuatu sesuai dengan komponen yang ada pada *GameObject*. komponen meng-
- 2 implementasikan fungsi-fungsi nyata pada *GameObject*. Contohnya objek cahaya dapat
- 3 terbuat dengan menambahkan komponen Light kepada suatu *GameObject* (Gambar 2.11).



Gambar 2.11: Contoh penggunaan komponen pada *GameObject*.

- 4 Selain itu ada pula contoh *GameObject* Kubus. *GameObject* ini memiliki komponen Cube(Mesh Filter) dan Mesh Renderer. Kedua komponen tersebut berguna untuk menggambar
- 5 permukaan dari kubus tersebut. Selain itu ada juga komponen Box Collider yang digunakan
- 6 untuk merepresentasikan *GameObject* tersebut dalam hal-hal fisika seperti gravitasi, gaya,
- 7 dan lain sebagainya(Gambar 2.12).
- 8



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 dipahami 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.

- 1 ● Euler Angles dapat mengalami Gimbal Lock. Gimbal Lock adalah kejadian ketika dua
2 sumbu dari sudut perputaran pada Euler Angles berputar pada poros yang sama.
- 3 Kelebihan menggunakan representasi Quaternion.
- 4 ● Quaternion tidak mengalami Gimbal Lock.
- 5 Kekurangan menggunakan representasi Quaternion.
 - 6 ● suatu quaternion tidak dapat merepresentasikan rotasi yang melebihi 180 derajat pada
7 arah manapun.
 - 8 ● Representasi angka-angka pada Quaternion lebih susah untuk dimengerti.
- 9 Pada Unity, meski pun semua orientasi disimpan dengan menggunakan Quaternion, te-
10 tapi pada *Transform Inspector* (Gambar ??) Unity menampilkan nilai-nilai orientasi pada
11 Euler Angles. Hal ini bertujuan untuk mempermudah mengubah orientasi bagi pengguna,
12 karena nilai-nilainya lebih mudah untuk dimengerti. Representasi pada *Transform Inspector*
13 tersebut menyebabkan masukkan yang diberikan oleh pengguna terkadang berbeda dengan
14 data yang disimpan. Contohnya ketika berada pada sudut 365 derajat, maka data yang
15 akan disimpan akan menunjukkan perputaran sebesar 5 derajat saja.

16 2.5 Google VR SDK for Unity

17 Google juga menyediakan SDK untuk *Game Engine* Unity. Berbeda dengan Google Cardbo-
18 ard API, Google VR SDK for Unity tidak hanya berfungsi untuk membuat aplikasi untuk
19 Google Cardboard. Google VR SDK juga berfungsi untuk membuat aplikasi untuk Google
20 Daydream, yaitu Aplikasi VR dari Google sebagai pembaruan Google Cardboard. Google
21 daydream memiliki kelebihan memiliki *remote controller* yang dapat merekam gerakan ta-
22 ngan pengguna. Jadi jika menggunakan Google VR SDK for Unity, pembuat permainan
23 juga dapat membuat pengaturan untuk *remote controller* pada Google Daydream.

24 Untuk mendapatkan Google VR SDK for Unity, SDK tersebut dapat di download pa-
25 da halaman situs web Google VR Developers.[9] Pada situs tersebut akan diberikan assets
26 package dengan ekstensi file ".unitypackage". Assets package tersebut akan memasukkan
27 seluruh kebutuhan untuk membuat Google VR pada proyek permainan yang sedang diker-
28 jakan.

29 Untuk dapat me-*render* permainan dengan tampilan *stereo rendering* (tampilan layar
30 yang di bagi menjadi dua bagian sesuai dengan mata manusia), hierarki GameObject pa-
31 da scene yang sedang dikerjakan harus memiliki GvrViewerMain dengan komponen script
32 GvrViewer. GameObject ini dapat diperoleh dengan menggunakan *prefabs* yang telah di-
33 *import*. Selain untuk me-*render*, permainan object tersebut juga mengimplementasikan ori-
34 entasi pada *smartphone* secara langsung.

35 Seperti yang sudah dijelaskan Google Cardboard memberikan masukkan dengan me-
36 narik/menekan pelatuk pada Google Cardboard tersebut. Untuk mendapatkan masukkan
37 tersebut, dapat dilakukan dengan cara menggunakan *prefab* GvrEventSystem. Kemudian
38 menambahkan komponen Event Trigger pada GameObject yang akan merespons masukkan
39 tersebut. Respons dapat dilakukan dengan menambahkan Script baru pada GameObject
40 tersebut.

1

BAB 3

2

ANALISIS

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

6 **3.1 Analisis Aplikasi Sejenis**

7 Aplikasi sejenis pada perangkat *Virtual Reality* Google Cardboard hanyalah ada satu apli-
8 kasi saja. Aplikasi tersebut adalah aplikasi InMind VR. Aplikasi sejenis pada perangkat
9 *Virtual Reality* Oculust Rift adalah Trial of the Rift Drifter dengan Asunder yang di kem-
10 bangkan oleh AldinDynamics [13]. Aplikasi sejenis pada *Virtual Reality* Oculust Rift tidak
11 dapat dianalisis karena penulis tidak memiliki perangkat Oculust Rift.

12

13 **Analisis InMind VR**

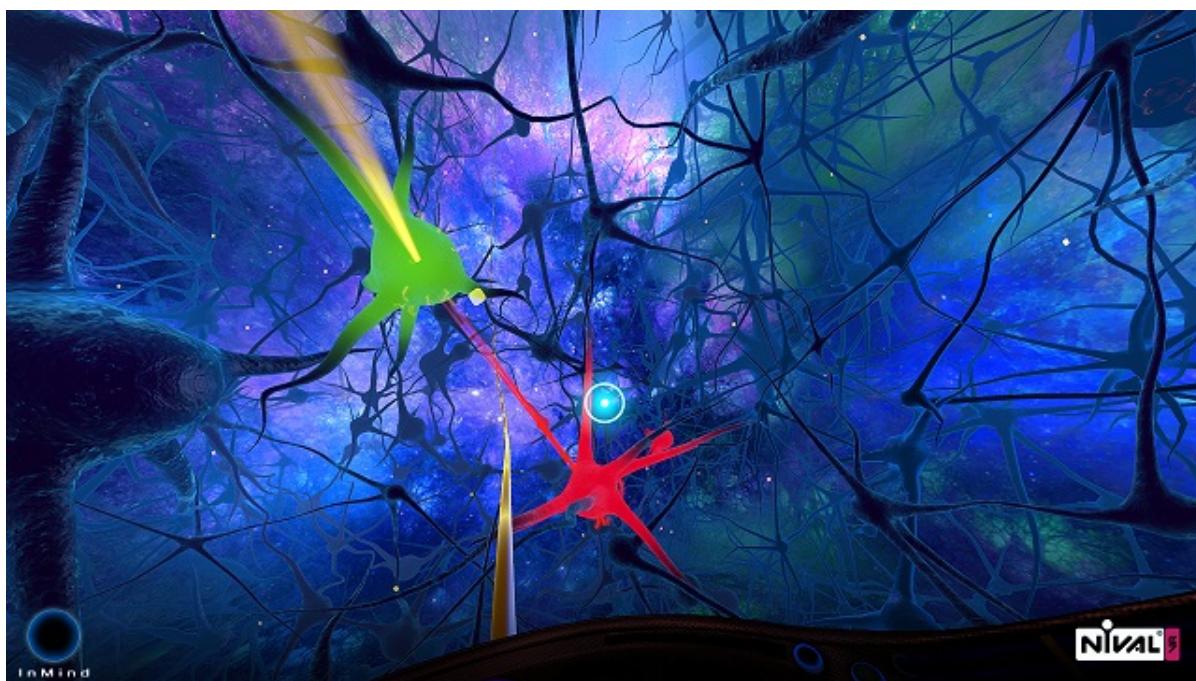
14

15 InMind VR merupakan permainan *Virtual Reality* yang seakan-akan pengguna berada
16 dalam sel-sel otak seorang manusia. Permainan ini dimainkan dengan mengarahkan arah
17 pandang kepala ke arah sel-sel neuron yang dapat menyebabkan gangguan mental (Gambar
18 3.1). Akan ada 50 sel neuron yang dapat menyebabkan gangguan mental. Sel-sel yang dapat
19 menyebabkan gangguan mental adalah sel-sel yang berwarna merah seperti pada Gambar
20 3.2.



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

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



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

3 Analisis dilakukan dengan cara mengetes aplikasi ini dengan mengangguk dan mengge-
4 leng yang spesifik. Mengangguk dilakukan dengan cara-cara berikut:

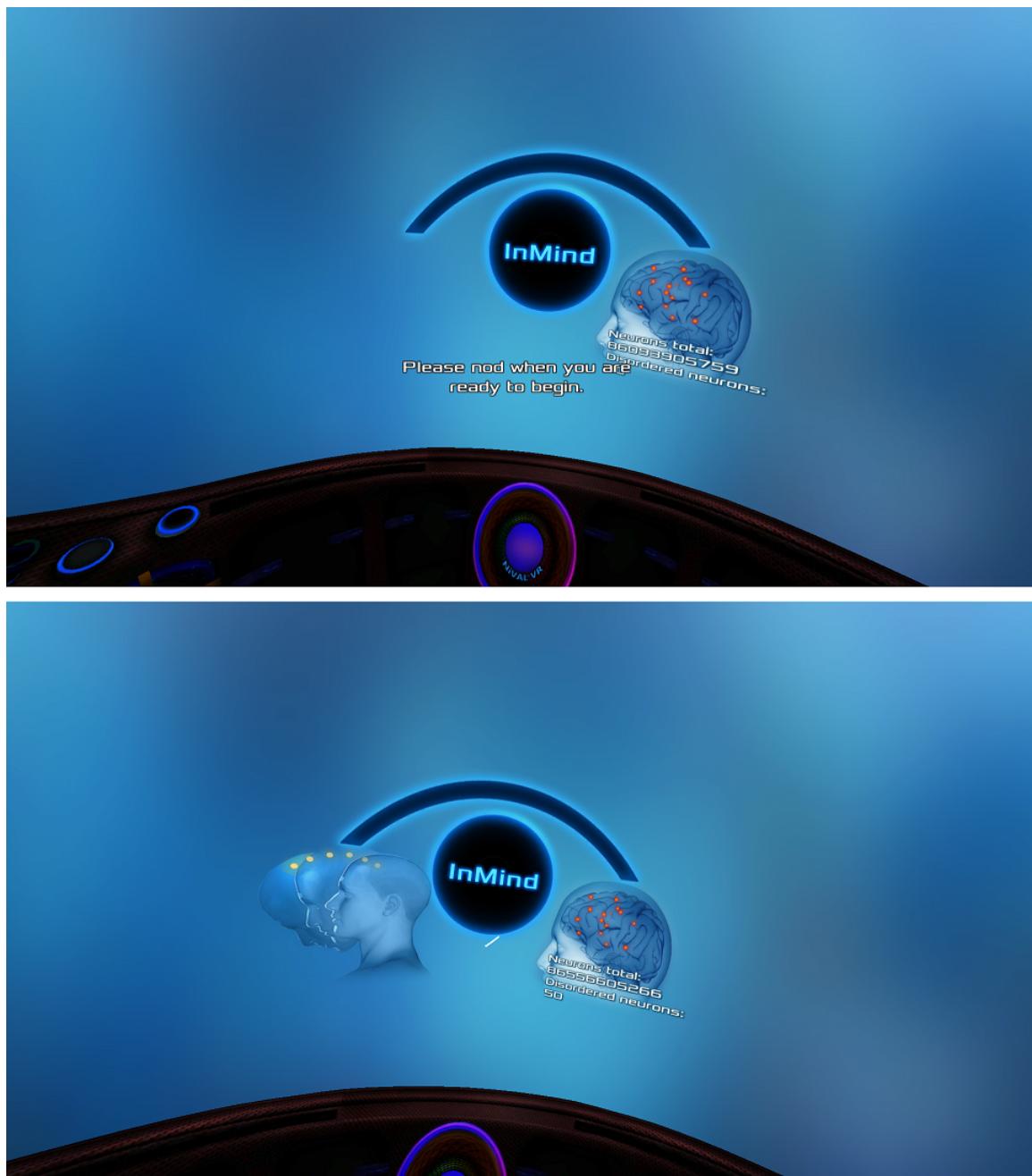
- 5 1. Mengangguk secara pelan.
6 2. Mengangguk yang diawali dengan gerakan ke atas.
7 3. Melihat kebawah saja tanpa membalikkan kepala ke posisi semula.
8 4. Tidak menggerakkan kepala sama sekali.

9 Dari keempat percobaan mengangguk yang dilakukan, hanya percobaan pertama dengan
10 percobaan ketiga yang terdeteksi mengangguk. Aplikasi ini dapat disimpulkan dengan ha-
11 nya menggerakkan kepala ke bawah saja sudah dapat dianggap mengangguk oleh aplikasi
12 ini. Pada percobaan kedua dan keempat terjadi keganjilan dari pendekstian anggukan.
13 Pada percobaan kedua dan keempat aplikasi tetap akan melanjutkan permainan setelah be-
14 berapa detik berlalu. Sel-sel yang dapat menyebabkan gangguan mental adalah sel-sel yang
15 berwarna merah seperti pada Gambar 3.2.

16 Ketika dilakukan percobaan menggeleng, tidak ada respon apapun dari aplikasi ini. Apli-
17 kasi ini akan tetap melanjutkan ke permainan setelah beberapa detik telah berlalu. Hal ter-
18 sebut menyimpulkan bahwa aplikasi ini tidak dapat mendekksi gerakan menggeleng. Oleh
19 karena itu hal yang dilakukan oleh aplikasi ini hanyalah melihat apakah pengguna sudah
20 melihat ke bawah atau belum saja.

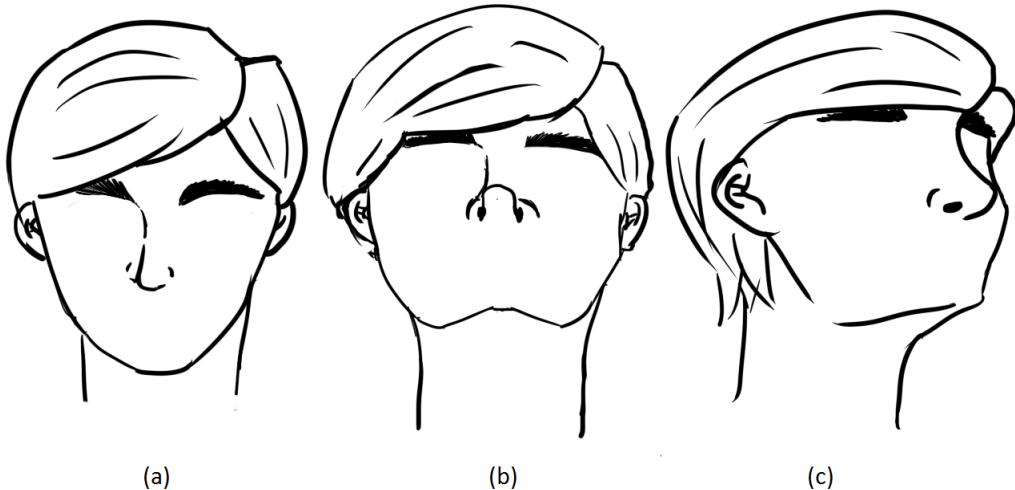
21 3.2 Perekaman Data Sensor

22 Pada analisis ini akan dilakukan perekaman mengangguk dan menggeleng dengan sensor-
23 sensor pada Android. Perekaman-perekaman ini akan dilakukan pada tiga kondisi muka
24 pengguna. Kondisi muka yang pertama adalah kondisi muka pengguna ketika menghadap
25 ke depan, digambarkan dengan Gambar 3.4 bagian (a). Kondisi muka yang kedua adalah



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

1 kondisi muka pengguna ketika menghadap ke atas sekitar 45° dari pandangan muka mengha-
 2 dap ke atas digambarkan dengan Gambar 3.4 bagian (b). Kondisi muka yang ketiga adalah
 3 kondisi muka ketika menghadap serong ke kiri atas digambarkan dengan Gambar 3.4 bagian
 4 (c). Anggukan yang dilakukan oleh pengguna hanya sebanyak satu kali mengangguk ke ba-
 5 wah saja. Sedangkan dalam menggeleng akan bergerak ke kiri terlebih dahulu dan ke kanan
 6 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.

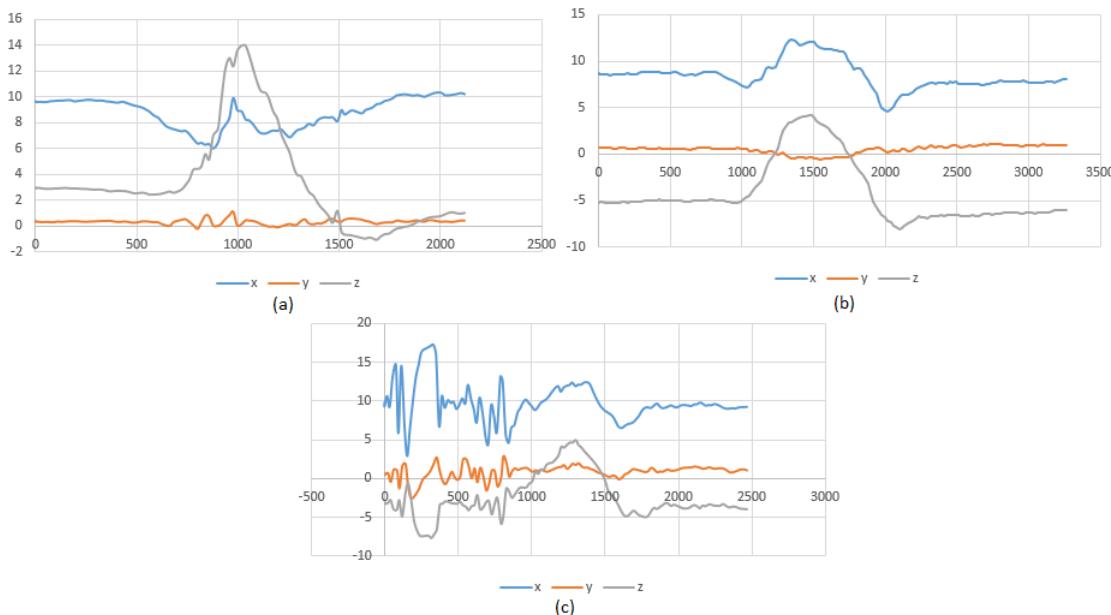
7 Grafik-grafik yang akan ditunjukkan pada bab ini akan memiliki beberapa karakteristik.
 8 Sumbu y pada grafik yang akan ditunjukkan akan merepresentasikan besar nilainya, Se-
 9 dangkan sumbu x akan merepresentasikan waktunya. Grafik yang ditunjukkan akan memi-
 10 liki beberapa nilai, tergantung dari jumlah nilai yang dikembalikan untuk setiap sensornya.
 11 Contohnya pada sensor *accelerometer* yang memiliki tiga jenis nilai, sehingga pada grafik
 12 akan terbentuk tiga buah garis nilai. Aplikasi akan merekam beberapa sensor secara lang-
 13 sung ketika pengguna mengangguk atau menggeleng, sehingga nilai waktunya akan sama
 14 untuk kondisi muka yang sama walaupun sensornya berbeda.

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

23 3.2.1 Perekaman Grafik Sensor *Accelerometer*

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

29 Gambar 3.5 merupakan grafik-grafik yang terbentuk dari nilai yang di terima oleh sensor
 30 *accelerometer* ketika pengguna sedang mengangguk. Pada Gambar 3.5 grafik (a) terlihat
 31 nilai z menaik dan nilai x menurun ketika sedang mengangguk. Tetapi nilai x kembali

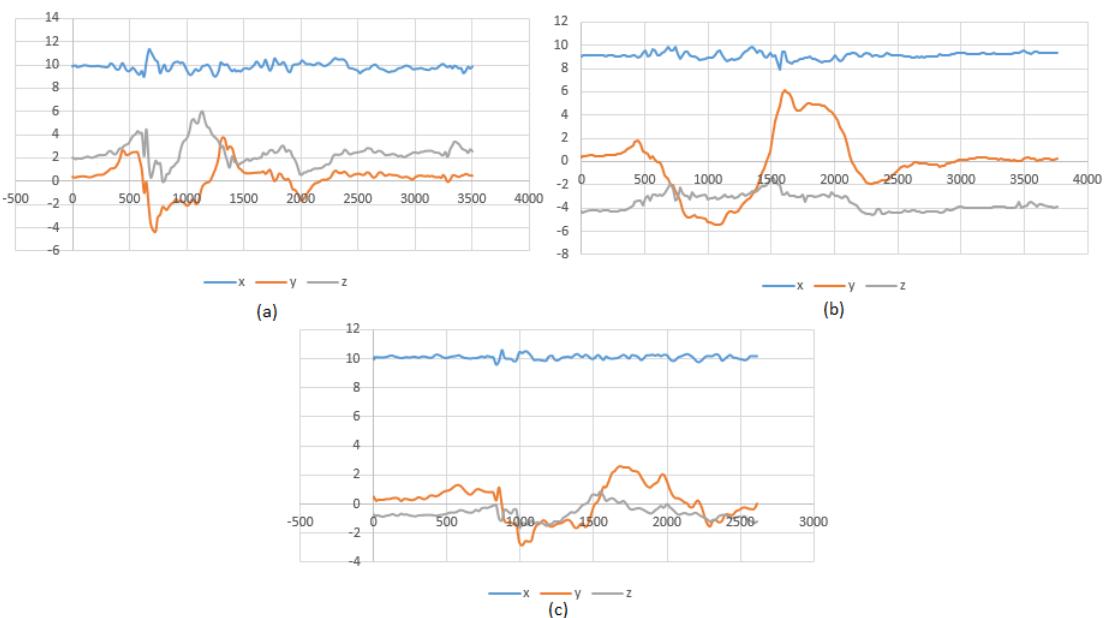


Gambar 3.5: Grafik nilai kuaternion dari sensor *accelerometer* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

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. Kedua nilai menaik ketika pengguna sedang 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 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.



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.

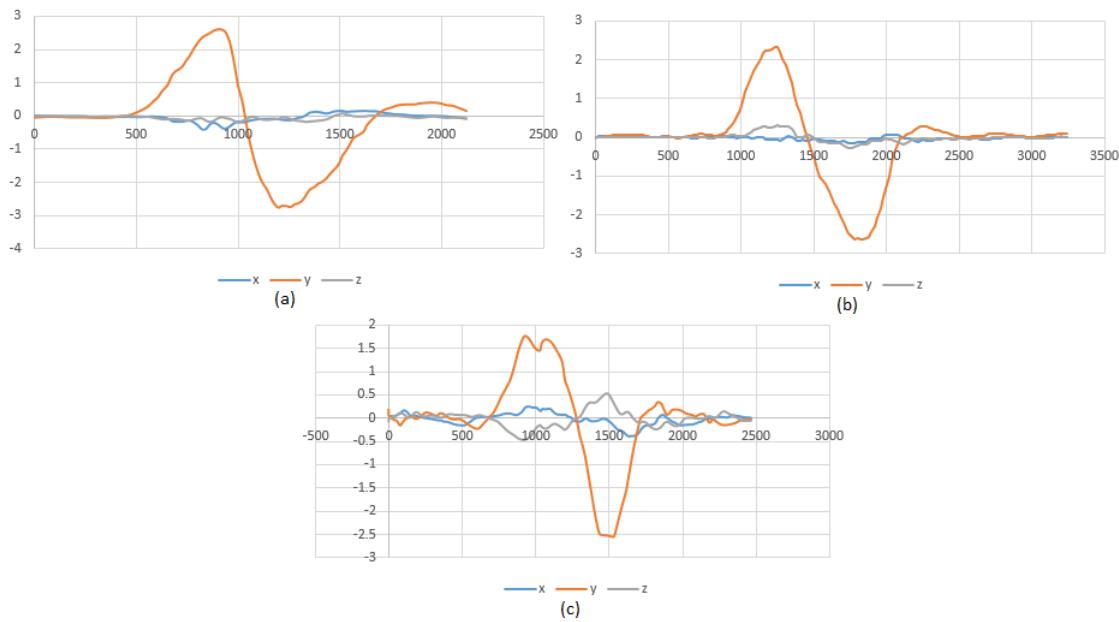
3.2.2 Perekaman Grafik Sensor *Gyroscope*

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.

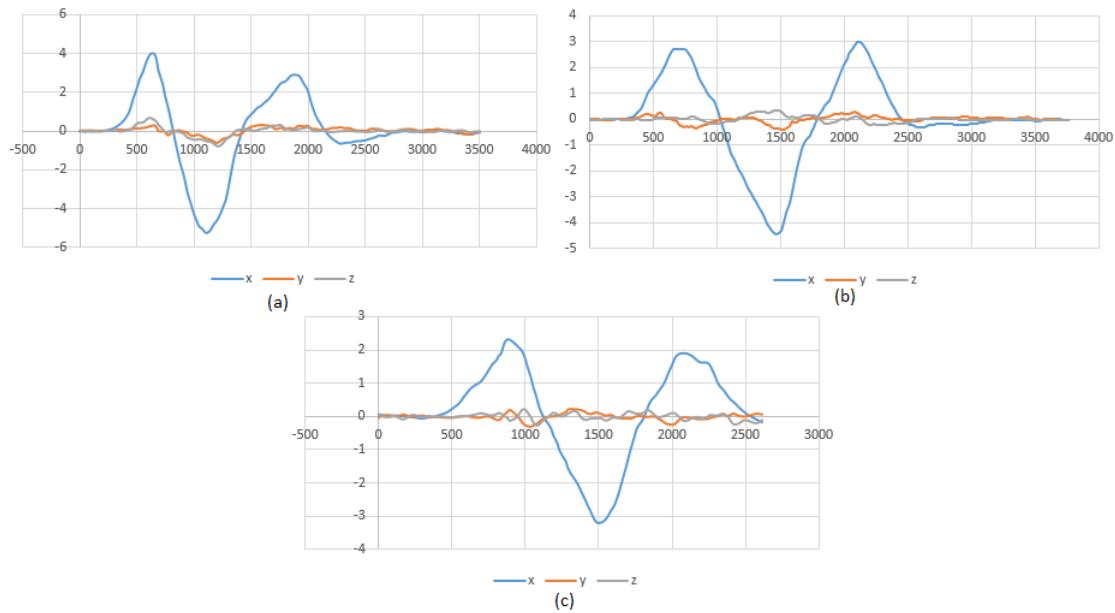
Gambar 3.7 menunjukkan nilai-nilai sensor *gyroscope* ketika pengguna sedang mengangguk. Grafik (a) membentuk sebuah bukit dan lembah pada nilai y. Bukit yang terjadi menunjukkan ketika pengguna menggerakkan kepalamanya ke bawah, dan ketika kepala pengguna kembali ke posisi semula percepatan angularnya 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 3.8 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.



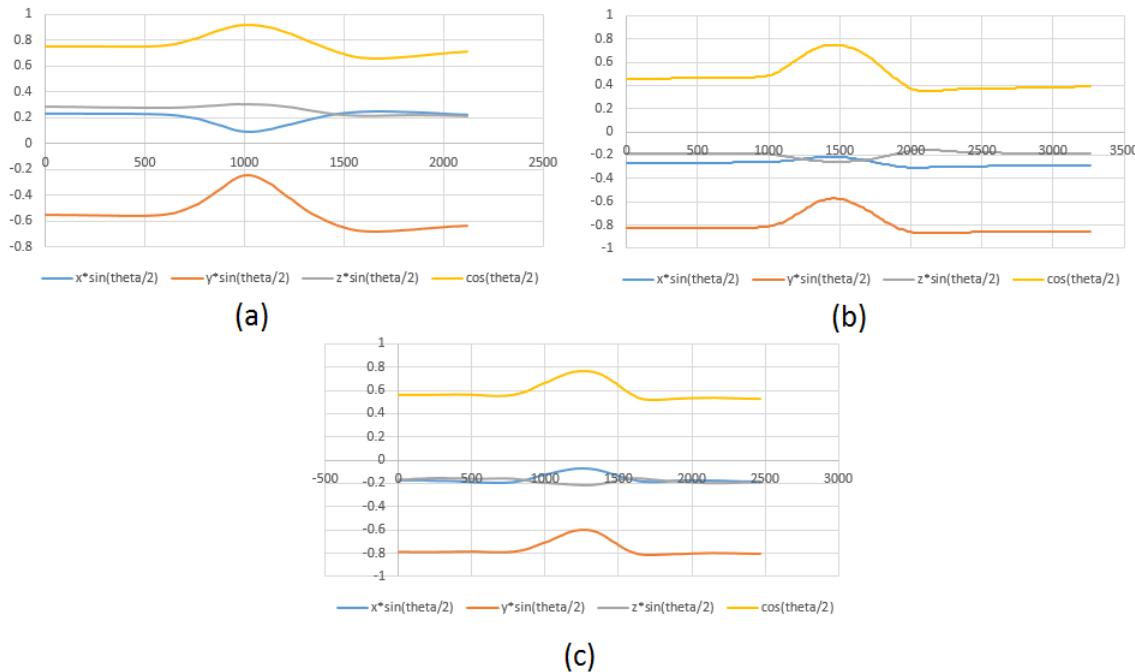
Gambar 3.7: Gambar grafik nilai sensor *gyroscope* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.



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.

3.2.3 Perekaman Grafik Sensor *Rotation Vector*

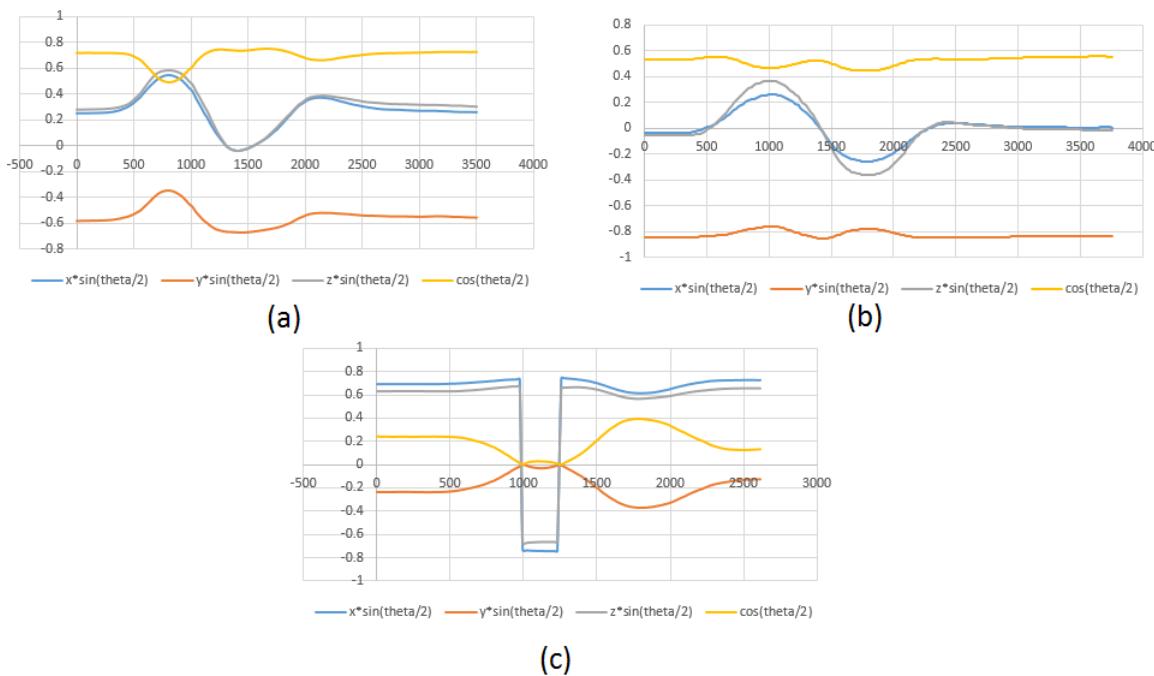
Perekaman menggunakan sensor *rotation vector* akan mendapatkan sebuah kuaternion yang merepresentasikan perputaran yang terjadi pada perangkat Android. Perputaran ini akan dideskripsikan dengan suatu vektor sebagai sumbu putarnya dan sudut perputarannya. Berbeda dengan sensor *gyroscope* yang merekam 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 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

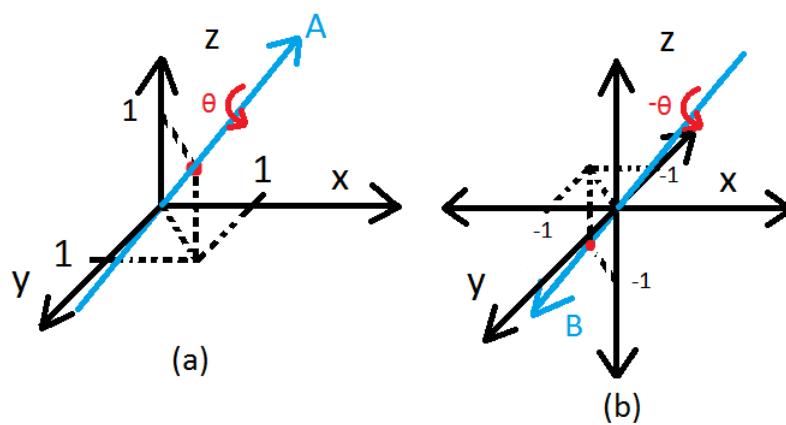


Gambar 3.10: Grafik nilai kuaternion dari sensor *rotation vector* ketika pengguna menggeleng dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

1 membentuk suatu bukit kecil ketika nilainya mencapai nilai 0 pada milidetik ke 1000. Garis
 2 jingga memiliki pola yang berlawanan dengan garis kuning. Pada saat garis kuning dengan
 3 garis jingga mencapai angka 0 perubahan drastis pun terjadi pada garis biru dengan abu-
 4 abu. Pada milidetik ke 1000 garis biru dengan abu mengalami perubahan nilai yang sangat
 5 drastis. Kedua nilai tersebut berubah dari nilai yang berkisar diantara 0.6 sampai 0.7 men-
 6 jadi berkisar diantara -0.7 sampai -0.8. Kasus ini tidak berarti sensor sedang mengalami
 7 kegagalan akurasi (*accuracy fail*), tetapi memang seperti itulah karakteristik dari perputar-
 8 an menggunakan kuaternion pada android. Perputaran yang terjadi pada batas mendekati
 9 sebelum terjadinya dengan setelah perubahan nilai yang drastis memiliki hasil perputaran
 10 yang sama. Hal ini disebabkan karena melakukan perputaran dengan vektor sebagai sum-
 11 bu dapat memiliki 2 buah nilai yang sejenis. Dua buah nilai tersebut akan menghasilkan
 12 perputaran yang sama ketika arah vektor dengan arah putarnya di balikkan seperti yang
 13 ditunjukkan pada Gambar 3.11. Sepertinya pada sistem android nilai $\cos(\theta/2)$ didesain
 14 agar tidak bernilai negatif, sehingga nilai-nilai yang lainnya akan mengalami perubahan nilai
 15 yang drastis ketika nilai $\cos(\theta/2)$ mencapai angka 0.

16 3.3 Analisis Data Sensor untuk Mendeteksi Gerakan Kepala

17 Dari ketiga hasil percobaan pada sensor-sensor pada bab 3.2 dapat disimpulkan bahwa sensor
 18 *gyroscope* adalah sensor yang terbaik untuk mendeteksi gerakan kepala. Data dari sensor
 19 *accelerometer* akan susah untuk digunakan dalam mendeteksi gerakan kepala karena ter-
 20 ganggu dengan aktivitas-aktivitas diluar gerakan pengguna yang juga ikut terekam oleh
 21 sensor *accelerometer*. Data dari sensor *rotation vector* juga akan lebih rumit dibandingkan
 22 dengan sensor *gyroscope*. Hal ini karena perputaran yang direkam oleh sensor *rotation vector*
 23 merekam kondisi putar pada suatu saat. Hasil rekaman ini akan mempersulit pada saat
 24 mendeteksikan gerakan kepala karena harus melakukan proses untuk menghitung kecepatan
 25 kepala bergerak, agar dapat membedakan gerakan mengangguk atau menggeleng atau se-
 26 kedar menoleh biasa. Dalam mendeteksi gerakan mengangguk dengan menggeleng banyak
 27 batas-batas yang perlu diperhatikan. Batas-batas tersebut untuk mengetahui apakah peng-



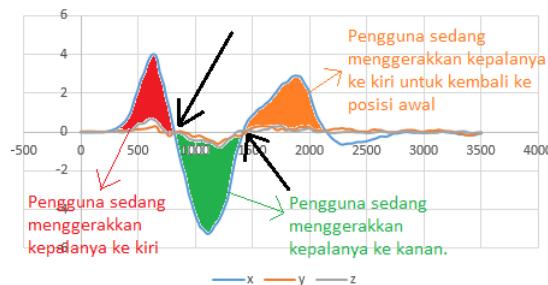
Gambar 3.11: Dua buah rotasi yang identik dengan nilai kuaternion yang berbeda.

guna 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.

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

13 3.4 Analisis Metode Pendeksi Gerakan Kepala

14 Seperti yang sudah dijelaskan pada bab 2.1.4, data akan didapatkan setiap ada perubahan
15 nilai pada sensor dengan rentang waktu tertentu, bergantung dengan konfigurasinya. Pen-
16 deteksian ini membutuhkan data yang *real-time*, sehingga akan lebih baik jika menggunakan
17 konfigurasi kecepatan pengambilan data setiap 20.000 mikrodetik. Penggunaan konfigurasi
18 dengan kecepatan 0.000 mikrodetik tidak terlalu baik, karena akan menggunakan kemam-
19 puan processor yang sangat tinggi.

20 3.4.1 Algoritma Mendeksi Gerakan Mengangguk

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

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

- 28 • **passLimitUp**, attribut ini akan mencatat apakah pengguna sudah melewati batas ke-
29 cepatan sudut ketika kepala pengguna bergerak ke atas.
- 30 • **passLimitDown**, attribut ini akan mencatat apakah pengguna sudah melewati batas
31 kecepatan sudut ketika kepala pengguna bergerak ke bawah.
- 32 • **currentlyLimitUp**, attribut ini akan menunjukkan bahwa pada saat ini gerakan peng-
33 guna sedang bergerak ke atas dan melebihi batas kecepatan sudutnya.
- 34 • **currentlyLimitDown**, attribut ini akan menunjukkan bahwa pada saat ini gerakan peng-
35 guna sedang bergerak ke bawah dan melebihi batas kecepatan sudutnya.
- 36 • **angSpeedLimit**, attribut ini akan menyimpan batas kecepatan sudut.
- 37 • **lastYAngSpeed**, attribut ini akan memiliki kecepatan sudut Y pada pemanggilan
38 method sebelumnya.
- 39 • **lastT**, attribut ini akan memiliki waktu dipanggilnya method sebelumnya.
- 40 • **calcArea**, attribut ini akan menyimpan besar luas bukit atau, lembah yang terjadi.
41 Luas bukit akan bernilai positif, dan nilai lembah akan bernilai negatif.
- 42 • **startTValley**, attribut ini akan menyimpan waktu mulai lembah yang memenuhi
43 syarat kecepatan sudut minimal.

- 1 ● `endTValley`, attribut ini akan menyimpan waktu akhir lembah yang memenuhi syarat
2 kecepatan sudut minimal.
- 3 ● `startTCurrMotion`, attribut ini akan menyimpan waktu mulai suatu gerakan(bukit
4 atau lembah) yang sedang berlangsung sekarang.
- 5 ● `endTCurrMotion`, attribut ini akan menyimpan waktu akhir suatu gerakan(bukit atau
6 lembah) yang sedang berlangsung sekarang.
- 7 ● `deviationLimit` adalah batas simpangan untuk melakukan gerakan mengangguk.
- 8 ● `deviationMotionDown`, attribut ini akan menyimpan simpangan yang terjadi ketika
9 pengguna menggeraka kepalanya ke bawah.
- 10 ● `deviationMotioUp`, attribut ini akan menyimpan simpangan yang terjadi ketika peng-
11 guna menggeraka kepalanya ke atas.

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

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

dengan,

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

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

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

Algorithm 1 Nod Detection Algoritma

```

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:       $startTValley \leftarrow startTCurrMotion$ 
16:       $endTValley \leftarrow endTCurrMotion$ 
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

```

3.4.2 Algoritma Mendeteksi Gerakan Menggeleng

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

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

- 9 • **currentlyLimitLeft**, attribut ini akan menunjukkan bahwa pada saat ini gerakan
10 pengguna sedang bergerak ke kiri dan melebihi batas kecepatan sudutnya.
- 11 • **currentlyLimitRight**, attribut ini akan menunjukkan bahwa pada saat ini gerakan
12 pengguna sedang bergerak ke kanan dan melebihi batas kecepatan sudutnya.
- 13 • **lastXAngSpeed**, attribut ini akan memiliki kecepatan sudut X pada pemanggilan
14 method sebelumnya.
- 15 • **deviationMotionLeft**, attribut ini akan menyimpan simpangan yang terjadi ketika
16 pengguna menggerakkan kepala ke kiri.
- 17 • **deviationMotioRight**, attribut ini akan menyimpan simpangan yang terjadi ketika
18 pengguna menggerakkan kepala ke kanan.

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

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

Algorithm 2 Shook Detection Algoritm

```

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 return true
31:  else if head moves left first AND time range between hill and valley no exceed the
limit. then return true
32:  else return false

```

1

BAB 4

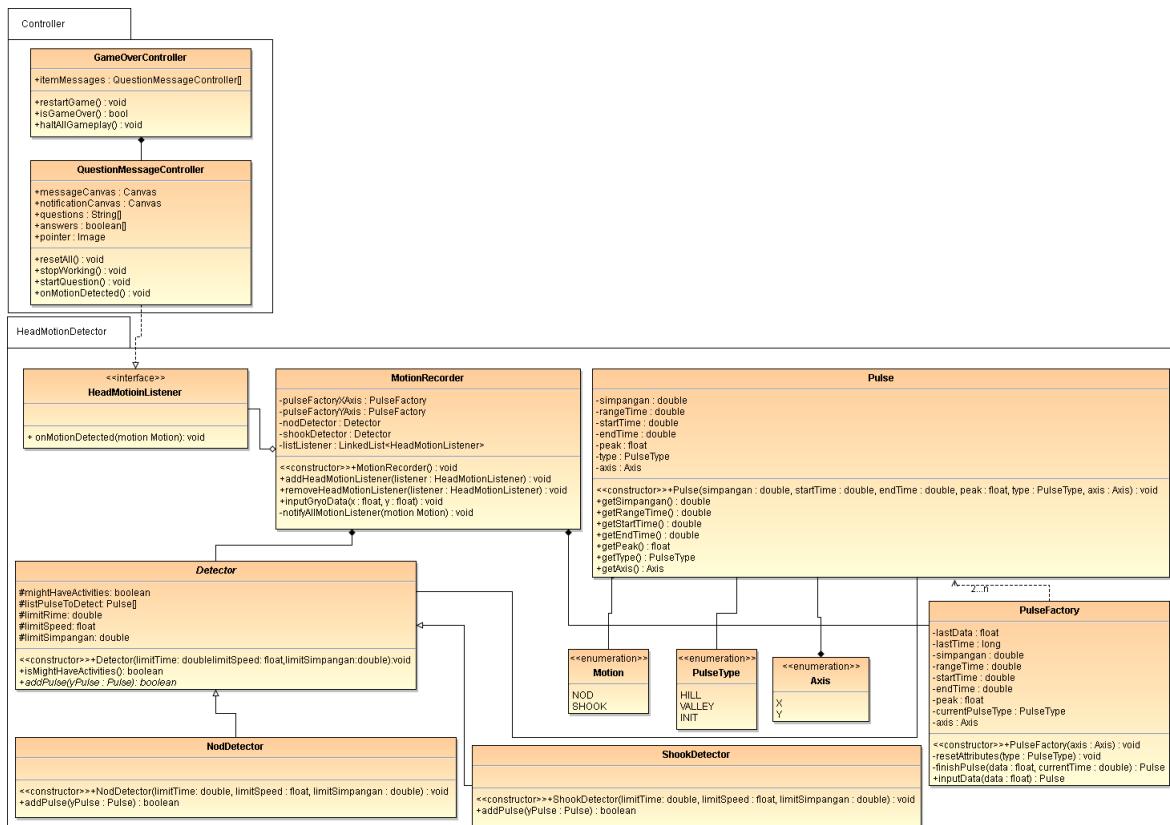
2

PERANCANGAN

- 3 Pada bab ini akan dijelaskan mengenai perancangan aplikasi yang akan dibangun meliputi
 4 perancangan kelas algoritma pendekripsi gerakan kepala, *GameObject*, *Gameplay*, dan per-
 5 ancangan antarmuka.

6 4.1 Perancangan Kelas Algoritma Pendekripsi Gerakan Kepala

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



Gambar 4.1: Diagram Kelas Algoritma Pendekripsi Gerakan Kepala.

13

- Package Controller

14

- Kelas GameOverController

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

3 * public QuestionMessageController[] itemMessages

4 Atribut ini digunakan untuk menyimpan seluruh QuestionMessageController
 5 yang ada pada permainan agar dapat diberhentikan dan di atur ulang.

6 Method-method pada kelas ini adalah sebagai berikut:

7 * public void restartGame()

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

10 * public boolean isGameOver()

11 Method ini berfungsi untuk mengecek apakah permainan sudah selesai atau
 12 belum. Method ini akan di panggil secara berulang-ulang setiap satu *frame*
 13 pada permainan.

14 * public void haltAllGameplay()

15 Method ini berfungsi untuk memberhentikan seluruh fungsi permainan. Me-
 16 thod ini akan dipanggil ketika permainan telah selesai.

17 – Kelas QuestionMessageController

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

22 * public Canvas messageCanvas

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

25 * public Canvas notificationCanvas

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

28 * public String[] question

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

31 * public boolean[] answers

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

34 * public Image pointer

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

37 Method-method pada kelas ini adalah sebagai berikut:

38 * public void resetAll()

39 Method ini berfungsi untuk mengembalikan kondisi permainan menjadi kon-
 40 disi awal permainan.

41 * public void stopWorking()

42 Method ini berfungsi untuk memberhentikan semua fungsi pada suatu objek
 43 QuestionMessageController.

44 * public void startQuestion()

45 Method ini berfungsi untuk memulai atau memunculkan suatu pertanyaan
 46 pada tampilan.

47 * public void onMotionDetected(Motion motion)

48 Method ini akan dipanggil jika terdeteksi suatu gerakan kepala. Parameter
 49 motion akan diisi dengan spesifikasi gerakan yang terdeteksi.

- 50 • Package HeadMotionDetector

```

1   – interface HeadMotionListener
2     Interface ini akan di implements oleh kelas-kelas yang akan menggunakan hasil
3     dari pendekripsi gerakan kepala. Method-method abstrak pada interface ini
4     adalah sebagai berikut:
5       * public void onMotionDetected(Motion motion)
6         Method ini akan dipanggil oleh jika terdeteksi suatu gerakan kepala. Para-
7         meter motion akan diisi dengan spesifikasi gerakan yang terdeteksi.
8   – Kelas MotionRecorder
9     Kelas ini digunakan untuk merekam, menentukan, dan mendekripsi gerakan kepala.
10    Atribut-atribut pada kelas ini adalah sebagai berikut:
11      * private PulseFactory pulseFactoryXAxis
12        Atribut ini digunakan untuk menyimpan PulseFactory yang merekam data
13        gyroscope pada sumbu X.
14      * private PulseFactory pulseFactoryYAxis
15        Atribut ini digunakan untuk menyimpan PulseFactory yang merekam data
16        gyroscope pada sumbu Y.
17      * private Detector nodDetector
18        Atribut ini digunakan untuk menyimpan Pendekripsi gerakan mengangguk.
19      * private Detector shookDetector
20        Atribut ini digunakan untuk menyimpan Pendekripsi gerakan menggeleng.
21      * private LinkedList<HeadMotionListener> listListener
22        Atribut ini digunakan untuk menyimpan kelas-kelas yang menjadi listener
23        pendekripsi gerakan kepala.

24  Method-method pada kelas ini adalah sebagai berikut:
25    * public MotionRecorder()
26      Constructor ini digunakan untuk menginisialisasi atribut-atribut yang akan
27      di gunakan.
28    * public void addHeadMotionListener(HeadMotionListener listener)
29      Method ini digunakan untuk menambahkan listener baru pada atribut list-
30      Listener
31    * public void removeHeadMotionListener(HeadMotionListener listener)
32      Method ini digunakan untuk menghapus suatu listener pada atribut listList-
33      ener.
34    * public void inputGyroData(float x, float y)
35      Method ini digunakan untuk memasukkan data gyroscope yang akan digu-
36      nakan untuk mendekripsi gerakan mengangguk dengan menggeleng.
37    * public void notifyAllMotionListener(Motion motion)
38      Method ini akan memberitahu seluruh listener ketika terdeteksi suatu gerak-
39      an. Jenis gerakan yang diberitahukan adalah jenis gerakan pada parameter
40      tersebut.

41  – Kelas PulseFactory
42    Kelas ini berguna untuk mendekripsi Pulse yang terjadi pada grafik data gyro-
43    scope. Beberapa atribut yang dimiliki oleh kelas ini akan menjadi kriteria dari
44    suatu Pulse yang akan terdeteksi. Atribut-atribut tersebut adalah:
45    * private float lastData
46      Atribut ini menyimpan data sebelum data yang baru dimasukkan
47    * private float lastTime
48      Atribut ini menyimpan waktu pada pemasukkan data sebelum daya yang
49      baru dimasukkan.
50    * private double simpangan
51      Atribut ini menyimpan simpangan terjauh pada Pulse terakhir yang terde-
52      meteksi.

```

```

1   * private double rangeTime
2     Atribut ini menyimpan rentang waktu yang terjadi pada suatu Pulse
3   * private double startTime
4     Atribut ini menyimpan waktu mulai dari suatu Pulse yang terjadi.
5   * private double endTime
6     Atribut ini menyimpan waktu akhir dari suatu Pulse yang terjadi.
7   * private float peak
8     Atribut ini menyimpan puncak nilai tertinggi atau terendah dari suatu Pulse
9     bukit atau lembah.
10  * private PulseType
11    Atribut ini menyimpan tipe Pulse antara bukit atau lembah.
12  * private Axis axis
13    Atribut ini mendefinisikan sumbu yang di rekam datanya.

```

Method-method pada kelas ini adalah:

```

15  * public PulseFactory(Axis axis)
16    Constructor ini berfungsi untuk mendefinisikan sumbu pada gyroscope yang
17    akan dideteksi.
18  * private void resetAttributes()
19    Method ini berfungsi untuk mengembalikan nilai-nilai atribut kembali ke nilai
20    asal.
21  * private Pulse finishPulse(float data, double currentTime)
22    Method ini akan dipanggil ketika grafik melewati angka nol. Pada pemang-
23    gilan ini Pulse baru akan terdeteksi dan resetAttributes() akan dipanggil
24    untuk pendekripsi Pulse selanjutnya.
25  * public Pulse inputData(float data)
26    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:

```

31  * public Detector(float limitSpeed, double limitSimpangan)
32    Constructor ini digunakan untuk meninisialisasi batasan-batasan yang dide-
33    finisikan pada atribut-atribut kelas ini.
34  * protected boolean mightHaveActivities()
35    Atribut ini menunjukkan apakah suatu detector memiliki kemungkinan se-
36    dang mendekripsi gerakan atau tidak.
37  * protected Pulse[] listPulseToDetect()
38    Atribut ini menyimpan beberapa Pulse untuk di periksa karakteristiknya,
39    apakah sesuai dengan algoritma pendekripsi gerakan kepala.
40  * protected float limitSpeed()
41    Atribut ini mendefinisikan batas kecepatan angular yang sesuai dengan pen-
42    deteksi gerakan kepala.
43  * protected double limitSimpangan()
44    Atribut ini mendefinisikan batas Simpangan yang sesuai dengan pendekripsi
45    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

1 serupa dengan atribut-atribut pada kelas PulseDetector. Method-method pada
2 kelas ini hanyalah Getter dengan Constructor saja.

3 4.2 Perancangan Hierarki GameObject pada Unity

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

10 permainan ini hanya akan dibangun dengan menggunakan dua buah scene saja. Scene
11 pertama digunakan hanya untuk mengecek apakah perangkat memiliki sensor gyroscope atau
12 tidak. Scene kedua digunakan untuk implementasi keseluruhan permainan. Permainan ini ti-
13 dak memerlukan Scene yang banyak karena permainan ini hanya memiliki satu ruangan saja.
14 Sebagian besar pada perancangan ini hanyalah mengelompokkan GameObject-GameObject
15 pada Scene. Perancangan hierarki scene pertama hanyalah satu buah GameObject Gyro-
16 scopeChecker saja. Gambar 4.2 merupakan hasil dari perancangan hierarki Scene kedua
17 permainan ini.

18 Pada Perancangan di atas ada beberapa GameObject utama, diantaranya adalah Player,
19 GameplayCanvas, Room, dan Items. GameObject Player akan digunakan sebagai camera
20 pada dunia permainan tersebut. Camera pada Player ini akan bergerak mengikuti orientasi
21 pada *smartphone*. Pada GameObject ini terdapat GameObject Main Camera yang ber-
22 fungsi untuk menjadi kamera pada permainan dan menentukan titik pandang pengguna.
23 Main Camera memiliki dua buah *Child* Game Object. Kedua GameObject tersebut adalah
24 GvrReticlePointer dan GameOverCanvas. GvrReticlePointer berguna untuk menampilkan
25 titik pandang pengguna. GvrReticlePointer ini akan menjadi lingkaran ketika objek yang
26 di pandangnya dapat memberikan respons dari masukkan tombol pada Google Cardbo-
27 ard. GameOverCanvas berguna untuk menampilkan pesan "Game Over!" ketika permainan
28 berakhir.

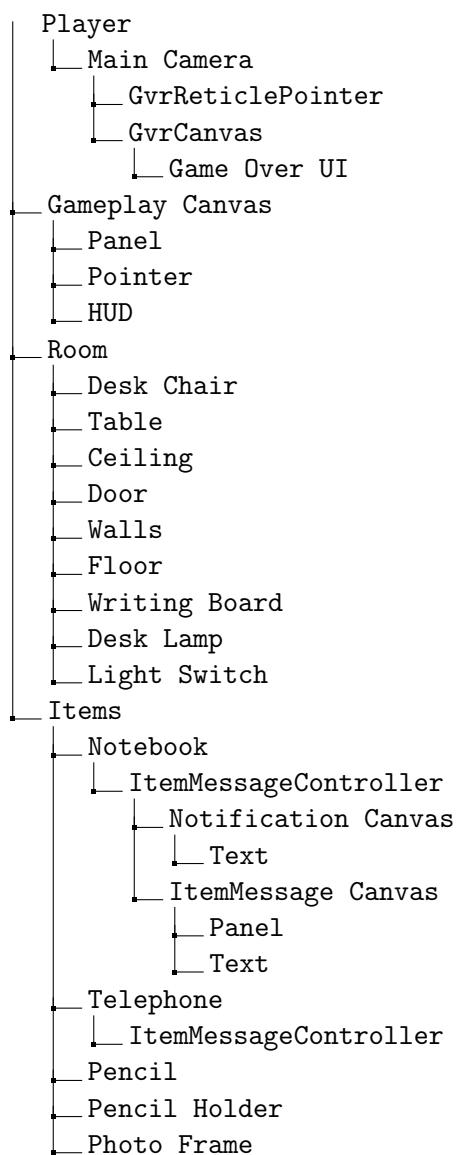
29 GameObject GameplayCanvas digunakan untuk menunjukkan kondisi permainan seka-
30 rang. GameObject ini memiliki dua buah child yaitu Panel, Pointer, dan HUD. Panel ber-
31 fungsi sebagai bidang putih pada UI. Pointer berfungsi sebagai menunjuk kondisi permainan
32 pada saat ini. HUD adalah bentuk kurang lebih 1/6 lingkaran yang membatasi Pointer.

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

36 GameObject Item digunakan untuk menyimpan model-model barang yang ada pada ru-
37 angan. Sebagian GameObject pada Item akan memiliki component yang akan diberikan
38 script untuk mendeteksi gerakan kepala. GameObject yang akan diberikan script terse-
39 but adalah GameObject yang memiliki *Child* GameObject ItemMessageController, yaitu
40 Notebook dengan Telephone. Selain kedua GameObject tersebut GameObject lainnya me-
41 rupakan model-model dekorasi pada meja.

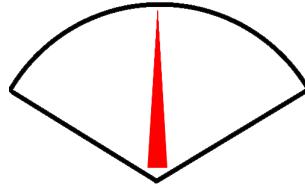
42 4.3 Perancangan *Gameplay* dan Perancangan Antarmuka

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



Gambar 4.2: Perancangan hierarki Game Object

1 Takaran kesenangan karyawan dengan keuntungan perusahaan di gambarkan dengan meteran seperti pada gambar 4.3. Jika jarum pada meteran tersebut mengarah ke kiri, berarti
 2 karyawan perusahaan sedang memiliki tingkat kesenangan yang tinggi, namun perusahaan
 3 tidak mendapatkan keuntungan yang baik. Begitu pula sebaliknya jika jarum mengarah ke
 4 kanan.
 5



Gambar 4.3: Meteran pada permainan.

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

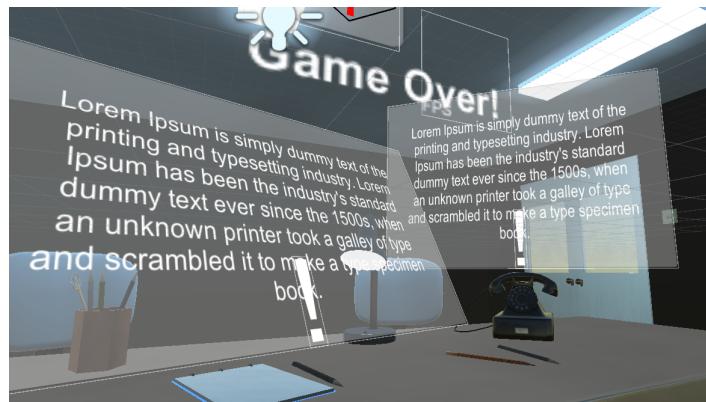
11 Antarmuka yang ditampilkan pada permainan ini merupakan pemandangan seseorang
 12 pimpinan perusahaan (Pemain) pada meja kerjanya. Ruangan tersebut akan sangat seder-
 13 hana, yaitu ruangan dengan bentuk balok biasa dan diisi dengan beberapa model untuk
 14 dekorasi. Pada meja kerja akan terdapat beberapa alat seperti, pensi, buku catatan, tele-
 15 pon, lampu, dan lain sebagainya. Alat-alat yang akan ditambahkan *script C#* untuk dapat
 16 mendeteksi anggukan dan gelangan adalah telepon dan buku catatan. Ruangan tersebut
 17 digambarkan pada gambar 4.4



Gambar 4.4: Desain ruangan untuk permainan yang akan dibuat.

18 Untuk menampilkan pertanyaan, pemain harus menunggu suatu alat menampilkan noti-
 19 fikasi yang ditunjukkan dengan memunculkan karakter tanda seru (!) di atas alat tersebut.
 20 Ketika notifikasi tersebut muncul, pemain dapat menghadapkan pandangannya ke alat terse-
 21 but dan menarik/menekan tombol pada Google Cardboard untuk menampilkan pertanyaan
 22 yang akan diberikan. Pertanyaan yang diberikan akan diberikan dengan memunculkan suatu
 23 *pop-up* di atas alat tersebut yang ditulis pertanyaan yang ditanyakan (Notifikasi dan *pop-up*
 24 ditunjukkan pada gambar 4.5). Jumlah pertanyaan yang dapat dimunculkan hanyalah satu
 25 pertanyaan saja, tidak dapat memunculkan dua buat pertanyaan secara bersamaan. Setelah
 26 pertanyaan dijawab dengan anggukan atau gelangan *pop-up* tersebut akan menghilang, dan

- 1 meteran akan bergerak tergantung dari jawaban pemain.



Gambar 4.5: Desain User Interface

BAB 5

IMPLEMENTASI DAN PENGUJIAN

³ Bab ini terdiri atas implementasi, pengujian, dan masalah yang dihadapi. Pada bagian
⁴ implementasi dijelaskan mengenai lingkungan implementasi dan hasil dari implementasi.
⁵ Pada bagian pengujian berisi hasil dari pengujian. Pada bagian masalah yang dihadapi
⁶ akan dijelaskan masalah-masalah menarik yang dihadapi pada saat implementasi.

5.1 Implementasi

⁸ 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

11 Berikut adalah spesifikasi laptop yang digunakan untuk implementasi:

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

Berikut adalah spesifikasi perangkat *smartphone* android yang digunakan untuk implementasi:

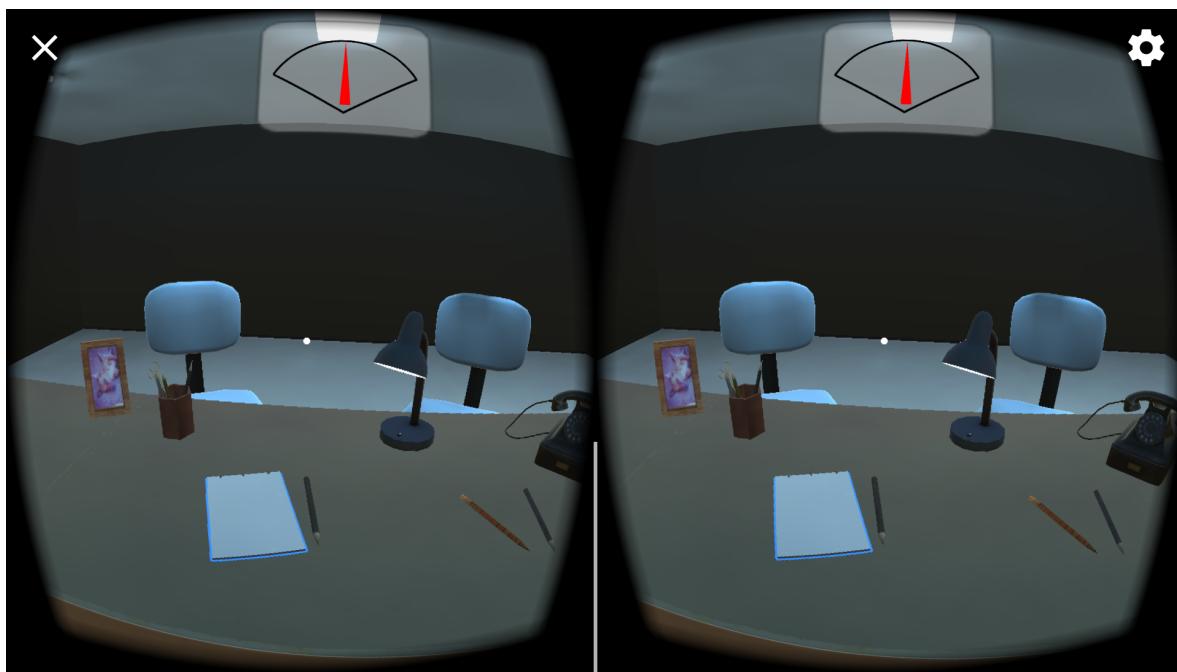
- 20 1. Processor : Qualcomm Snapdragon 625 Octa-core 2.0 GHz Cortex-A53
21 2. Sistem Operasi : Android OS 6.0 (Marshmallow)
22 3. Sensor-sensor : Accelerometer, gyro, proximity, compass, barometer

23 5.1.2 Hasil Implementasi

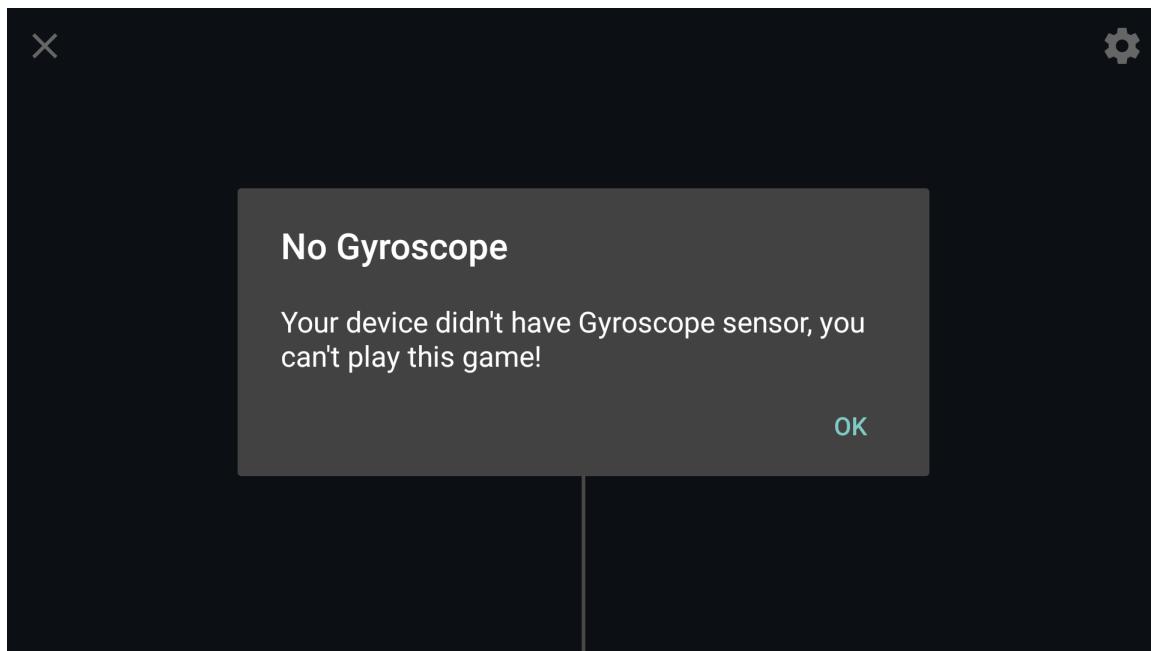
24 Hasil Implementasi ini adalah aplikasi permainan VR berbasis Android yang menggunakan
25 Game Engine Unity. Aplikasi ini dapat di *install* dengan menggunakan *Package Installer*
26 yang berekstensi file ".apk" pada Android.

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



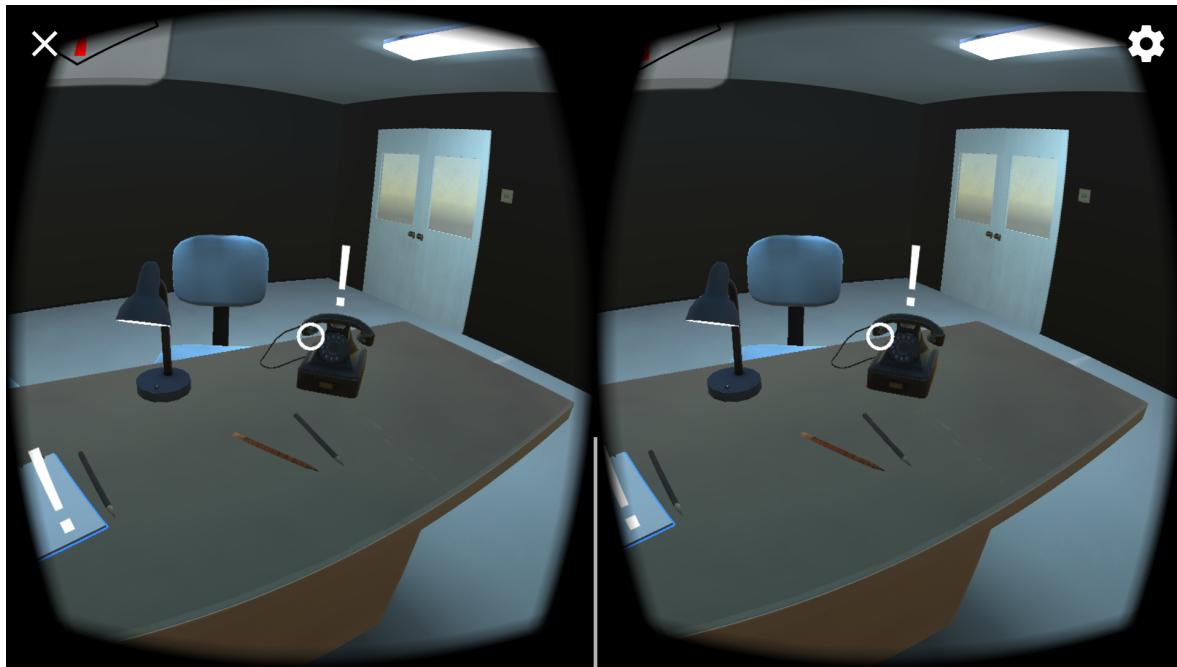
Gambar 5.1: Tampilan ketika aplikasi baru dimulai.



Gambar 5.2: Tampilan ketika perangkat tidak memiliki sensor Gyroscope.

1 2. **Aplikasi saat muncul notifikasi.**

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



Gambar 5.3: Tampilan ketika notifikasi muncul.

4 3. **Aplikasi saat memunculkan pertanyaan.**

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

7 4. **Aplikasi setelah memberikan respons mengangguk atau menggeleng.**

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

11 5. **Aplikasi ketika pemain kalah dalam permainan tersebut.** Ketika pengguna
12 kalah, maka akan muncul tulisan "Game Over!" di depan pandangan pengguna (Gam-
13 bar 5.7). Tulisan ini akan mengikuti pergerakan kepala pengguna. Jika pengguna
14 menarik/menekan tombol pada Google Cardboard, permainan akan diulang menjadi
15 kondisi awal.

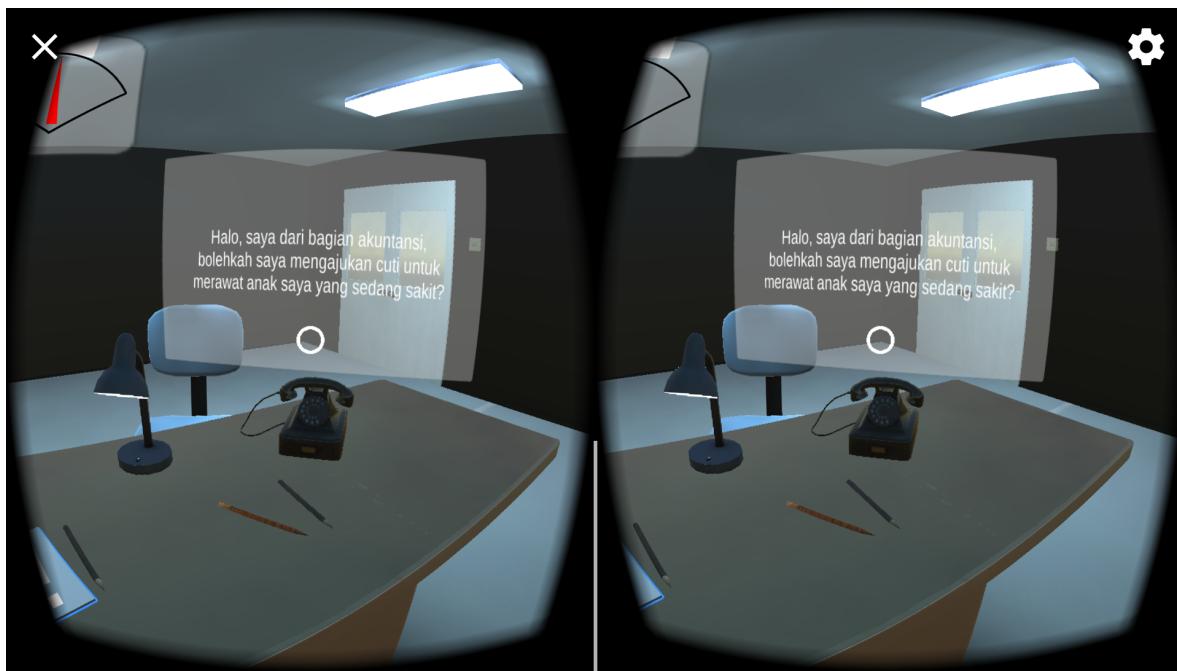
16 **5.2 Pengujian**

17 Pengujian dilakukan dengan menggunakan dua buah metode yaitu pengujian fungsional dan
18 pengujian eksperimental. Pengujian fungsional bertujuan untuk menguji fungsi-fungsi yang
19 disediakan pada aplikasi. Pengujian Eksperimental bertujuan untuk menguji pengalaman
20 pengguna dalam menggunakan aplikasi ini.

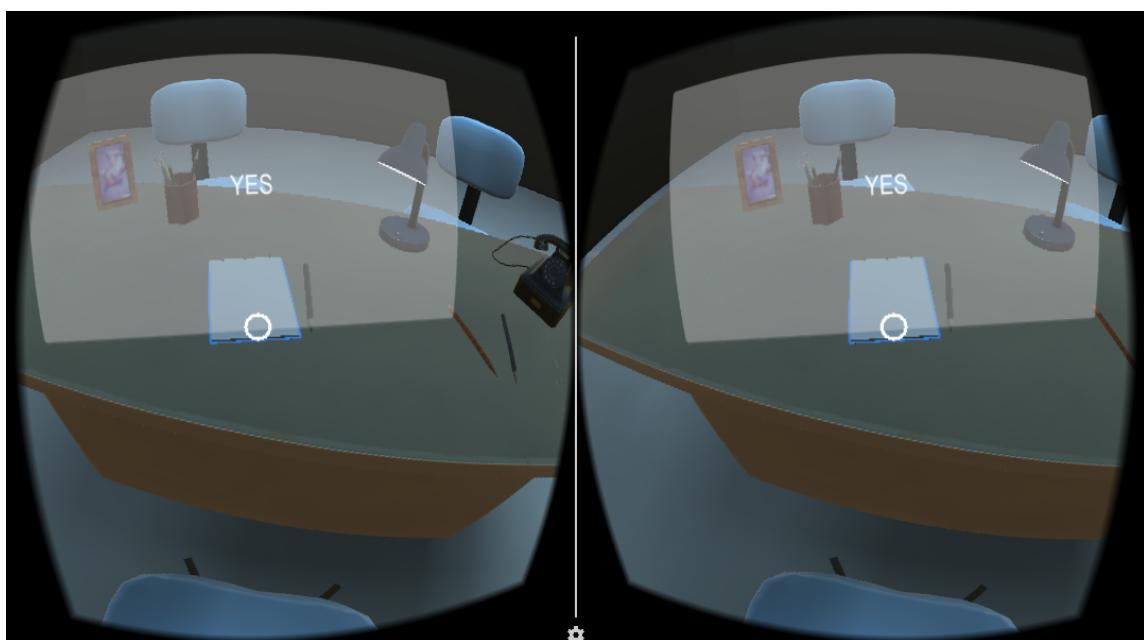
21 **5.2.1 Pengujian Fungsional**

22 Pengujian Fungsional ini dilakukan dengan mencoba menjalankan aplikasi pada perangkat-
23 perangkat yang berbeda spesifikasi-nya. Tabel 5.1 merupakan daftar perangkat-perangkat
24 *smartphone* Android yang di gunakan saat Implementasi

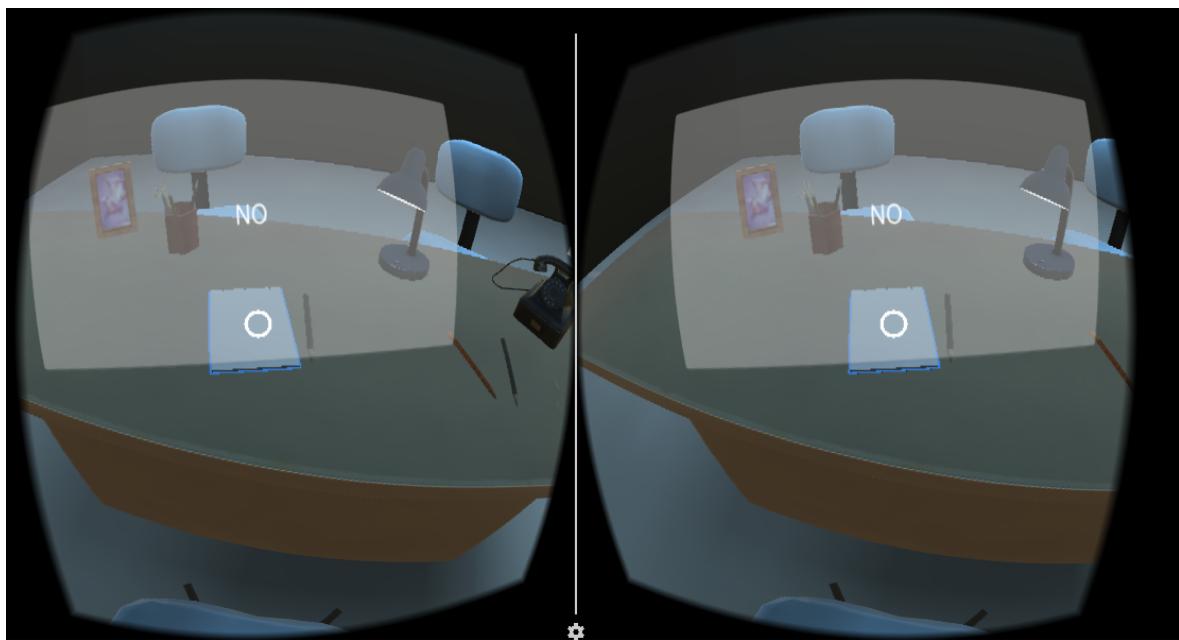
25 Pada *smartphone* 1 terjadi kegagalan pada saat instalasi pada perangkat. Perangkat ini
26 mengeluarkan *error* pada Google VR API for Unity, tetapi perangkat ini dapat berjalan



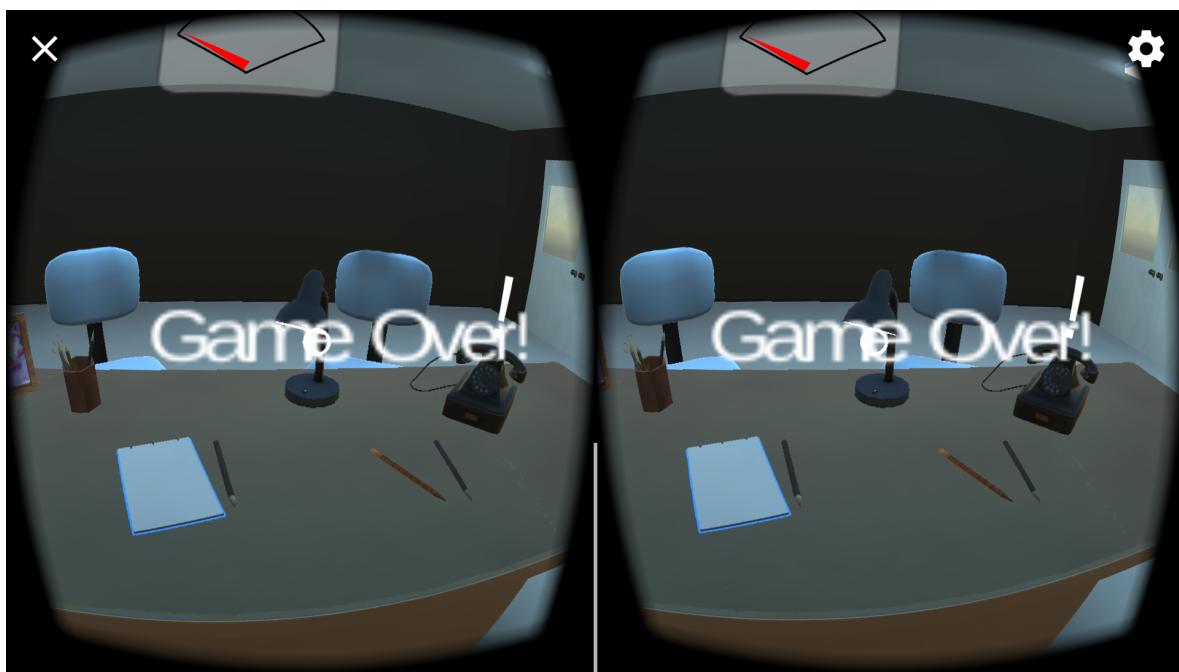
Gambar 5.4: Tampilan ketika *pop-up* pertanyaan muncul.



Gambar 5.5: Tampilan setelah pengguna mengangguk.



Gambar 5.6: Tampilan setelah pengguna menggeleng.



Gambar 5.7: Tampilan setelah permainan usai.

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

1 dengan baik ketika menjalankan aplikasi untuk menganalisis algoritma pendekripsi gerakan
 2 kepala. Aplikasi-aplikasi Google VR lainnya yang tidak menggunakan Unity dalam mem-
 3 buatnya dapat berjalan dengan baik. Hal ini disebabkan karena pada Google VR SDK for
 4 Unity membutuhkan Sistem Operasi minimum pada versi 4.4 (KitKat), sehingga perangkat
 5 ini tidak dapat menjalankan aplikasi ini. Aplikasi-aplikasi Google Cardboard yang dibu-
 6 at dengan menggunakan Google Cardboard SDK akan dapat berjalan dengan baik karena
 7 Google Cardboard SDK membutuhkan Sistem Operasi minimum pada versi 4.1 (Jelly Bean).

8 Pada *smartphone* 2 aplikasi dapat berjalan dengan baik.

9 Pada *smartphone* 3 aplikasi dapat berjalan dengan baik.

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

15 Pada *smartphone* 5 aplikasi dapat berjalan dengan baik, tetapi pergerakan kamera pada
 16 aplikasi terkadang tidak sesuai dengan pergerakan. Hal ini dikarenakan pada perangkat ini
 17 terdeteksi memiliki sensor Gryoscope, namun sensor gyroscope yang terbentuk pada perang-
 18 kat ini merupakan sensor yang menggunakan *software* untuk memenuhi fungsi gyroskopinya.
 19 Sensor Gyroscope yang terbentuk oleh *software* akan lebih tidak akurat dibandingkan de-
 20 ngan sensor Gyroscope yang terbentuk dari hardware. Oleh sebab itu pada perangkat ini
 21 aplikasi ini tidak berjalan begitu baik.

22 Pada *smartphone* 6 aplikasi dapat berjalan dengan baik.

23 5.2.2 Pengujian Eksperimental

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

27 Berikut adalah pengguna-pengguna pada pengujian aplikasi ini:

- 28 1. Ricky Setiawan (Gambar 5.8) menggunakan perangkat *smartphone* Android nomor 4

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



Gambar 5.8: Pengujian oleh Ricky Setiawan.

- 3 2. Harseto Pandityo (Gambar 5.9) menggunakan perangkat *smartphone* nomor 3
4 "Permainannya bagus dan menarik, khususnya pada fungsi mengangguk dengan meng-
5 geleng. Terkadang ada masalah ketika mengangguk tidak terdeteksi, tetapi harus
6 mengangguk dengan lebih jauh lagi simpangan-nya agar dapat terdeteksi. Tidak ada
7 masalah pada saat menggeleng."



Gambar 5.9: Pengujian oleh Harseto Pandityo.

- 8 3. Herfan Heryandi (Gambar 5.10) menggunakan perangkat *smartphone* nomor 5
9 "Untuk geleng sedikit kurang sensitif, tetapi untuk mengangguk sudah sensitif."
10 4. Kevin Antonius (Gambar 5.11) menggunakan perangkat *smartphone* 3
11
12 "Udah bagus, terkadang ketika mengangguk tidak terdeteksi. Mungkin karena ang-
13 gukan saya terlalu kecil simpangannya."



Gambar 5.10: Pengujian oleh Herfan Heryandi.



Gambar 5.11: Pengujian oleh Kevin Antonius.

- 1 5. Antonius Kurnia (Gambar 5.12) menggunakan perangkat *smartphone* 3
- 2
- 3 "Sensor untuk deteksinya sudah bagus, sudah dapat mendeteksi anggukan dan geleng-
- 4 an dengan baik. Hanya saja terkadang susah membaca tulisannya, karena gerakan
- 5 anggukan dan gelengannya"



Gambar 5.12: Pengujian oleh Antonius Kurnia.

6 5.3 Masalah yang Dihadapi pada Saat Implementasi

7 Berikut adalah beberapa masalah yang dihadapi pada saat implementasi:

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

¹ **BAB 6**

² **KESIMPULAN DAN SARAN**

³ **6.1 Kesimpulan**

⁴ Berdasarkan hasil penelitian yang dilakukan, diperoleh kesimpulan-kesimpulan sebagai ber-
⁵ ikut:

- ⁶ 1. Dalam pembuatan Grafik dapat diselesaikan menggunakan Android API pada Android
⁷ Studio. API ini digunakan untuk merekam seluruh data sensor-sensor pada perangkat
⁸ Android ketika pengguna mengangguk dan menggeleng. Data kemudian disimpan
⁹ pada file ber-format ".csv" dan dibuatkan grafiknya menggunakan Aplikasi Microsoft
¹⁰ Excel
- ¹¹ 2. Untuk dapat mendeteksi gerakan kepala khususnya mengangguk dan menggeleng, ha-
¹² nya diperlukan sensor gyroscope saja. Sensor-sensor lainnya tidak diperlukan untuk
¹³ membantu mendeteksian gerakan kepala. Selain dapat menggunakan gyroscope, ber-
¹⁴ dasarkan pengujian fungsional sensor gabungan accelerometer dan magnetometer da-
¹⁵ pat juga digunakan sebagai pengganti sensor gyroscope (sensor gyroscope software)
¹⁶ pada perangkat-perangkat tertentu.
- ¹⁷ 3. Aplikasi telah dapat membaca gerakan kepala dengan baik. Hal ini ditunjukkan dengan
¹⁸ tidak adanya kesalahan masukkan pada pengujian eksperimental.
- ¹⁹ 4. Berdasarkan hasil pengujian eksperimental, algoritma pada aplikasi ini dapat berjalan
²⁰ dengan baik. Kelemahan pada algoritma ini berada pada simpangan anggukan dengan
²¹ gelengan yang dilakukan, karena setiap orang mengangguk dan menggeleng dengan
²² besar simpangan yang berbeda-beda.

²³ **6.2 Saran**

²⁴ Berdasarkan hasil penelitian yang dilakukan, berikut adalah beberapa saran untuk pengem-
²⁵ bangan:

- ²⁶ 1. Mengimplementasikan fungsi kalibrasi anggukan dan gelenggan, sehingga sesuai de-
²⁷ ngan kriteria pengguna.
- ²⁸ 2. Mengimplementasikan fungsi pendeteksian gerakan kepala menggunakan sensor-sensor
²⁹ selain gyroscope, jika perangkat tidak memiliki sensor gyroscope. Sensor-sensor ter-
³⁰ sebut seperti accelerometer, compass, dan lain-lain. Jika hasilnya kurang memuaskan
³¹ dengan menggunakan satu buah sensor saja, implementasi dapat menggunakan me-
³² tode Sensor Fusion. Sensor Fusion adalah metode untuk menggabungkan nilai-nilai
³³ yang didapatkan pada setiap sensor untuk mencapai tujuan tertentu.
- ³⁴ 3. Membuat *library* untuk algoritma pendeteksi gerakan kepala, sehingga dapat diman-
³⁵ faatkan dan digunakan oleh pengembang lain.

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] "Android, the wold's most popular mobile platform | Android Developers." <https://developer.android.com/about/index.html>. [Online; diakses diakses 10-May-2016].
- [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].

1

LAMPIRAN A

2

KODE PROGRAM APLIKASI UNTUK ANALISIS

Listing A.1: MainActivity.java

```

3 package com.example.egaprianto.testingsensors;
4
5 import android.content.Intent;
6 import android.hardware.Sensor;
7 import android.hardware.SensorManager;
8 import android.support.v7.app.AppCompatActivity;
9 import android.os.Bundle;
10 import android.view.View;
11 import android.widget.TextView;
12
13 public class MainActivity extends AppCompatActivity {
14     TextView mTextView;
15     private SensorManager mSensorManager;
16     private Sensor mSensor;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22     }
23
24     @Override
25     public void onDestroy() {
26         super.onDestroy();
27         android.os.Debug.stopMethodTracing();
28     }
29
30     public void onClickLightSensorButton(View view){
31         Intent intent = new Intent(this, SensorAccelerometerActivity.class);
32         startActivity(intent);
33     }
34
35     public void onRotVecSensorButton(View view){
36         Intent intent = new Intent(this, QuaternionTheta.class);
37         startActivity(intent);
38     }
39
40     public void recordAllRawsensorData(View view){
41         Intent intent = new Intent(this, RecordAllRawsensorData.class);
42         startActivity(intent);
43     }
44
45     public void onGyroSensorClicked(View view){
46         Intent intent = new Intent(this, SensorGyroscopeActivity.class);
47         startActivity(intent);
48     }
49     public void onMagSensorClicked(View view){
50         Intent intent = new Intent(this, SensorGeomagneticRotationActivity.class);
51         startActivity(intent);
52     }
53 }
```

Listing A.2: RecordAllRawsensorData.java

```

54 package com.example.egaprianto.testingsensors;
55
56 import android.Manifest;
57 import android.content.Context;
58 import android.content.pm.PackageManager;
59 import android.hardware.Sensor;
60 import android.hardware.SensorEvent;
61 import android.hardware.SensorEventListener;
62 import android.hardware.SensorManager;
63 import android.media.Ringtone;
64 import android.media.RingtoneManager;
65 import android.net.Uri;
66 import android.os.Build;
67 import android.os.Bundle;
68 import android.os.CountDownTimer;
69 import android.os.Environment;
70 import android.support.v4.app.ActivityCompat;
71 import android.support.v7.app.AppCompatActivity;
72 import android.text.format.DateFormat;
73 import android.util.Log;
74 import android.view.View;
```

```

1 import android.view.WindowManager;
2 import android.widget.TextView;
3
4
5 import java.io.BufferedWriter;
6 import java.io.File;
7 import java.io.FileWriter;
8 import java.io.IOException;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 public class RecordAllRawSensorData extends AppCompatActivity implements SensorEventListener {
13     private static final String FILENAME = "sensorRecord-";
14     private SensorManager mSensorManager;
15     private ArrayList<Sensor> sensorArrayList;
16     private TextView textViewCountDown;
17     private boolean isCapturing;
18     private File file;
19     private ArrayList<double[]> acceleroData;
20     private ArrayList<double[]> gyroData;
21     private long startCaptureTime;
22     private long runningTime;
23     private ArrayList<double[]> rotData;
24     private ArrayList<double[]> rotVecData;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         this.sensorArrayList = new ArrayList<Sensor>();
30         setContentView(R.layout.activity_record_all_rawsensor_data);
31         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
32         textViewCountDown = (TextView) findViewById(R.id.textViewCountDown);
33         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
34             this.sensorArrayList.add(sensorGetter(Sensor.TYPE_ACCELEROMETER));
35             this.sensorArrayList.add(sensorGetter(Sensor.TYPE_GYROSCOPE));
36             this.sensorArrayList.add(sensorGetter(Sensor.TYPE_ROTATION_VECTOR));
37         }
38
39         int REQUEST_EXTERNAL_STORAGE = 1;
40         String[] PERMISSIONS_STORAGE = {
41             Manifest.permission.READ_EXTERNAL_STORAGE,
42             Manifest.permission.WRITE_EXTERNAL_STORAGE
43         };
44         int permission = ActivityCompat.checkSelfPermission(this, Manifest.permission.
45             WRITE_EXTERNAL_STORAGE);
46         if (permission != PackageManager.PERMISSION_GRANTED) {
47             ActivityCompat.requestPermissions(
48                 this,
49                 PERMISSIONS_STORAGE,
50                 REQUEST_EXTERNAL_STORAGE
51             );
52         }
53         isCapturing = false;
54         for (Sensor sensor :
55             sensorArrayList) {
56             mSensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_GAME);
57         }
58     }
59
60     @Override
61     protected void onResume() {
62         super.onResume();
63         for (Sensor sensor :
64             sensorArrayList) {
65             mSensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_GAME);
66         }
67     }
68
69     @Override
70     protected void onPause() {
71         super.onPause();
72         mSensorManager.unregisterListener(this);
73     }
74
75     @Override
76     public void onSensorChanged(SensorEvent event) {
77         Sensor sensor = event.sensor;
78         if (isCapturing) {
79             float[] copyData = new float[4];
80             System.arraycopy(event.values, 0, copyData, 1, 3);
81             copyData[0] = System.currentTimeMillis() - startCaptureTime;
82             if (sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
83                 this.acceleroData.add(convertFloatsToDoubles(copyData));
84             } else if (sensor.getType() == Sensor.TYPE_GYROSCOPE) {
85                 this.gyroData.add(convertFloatsToDoubles(copyData));
86             } else if (sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {
87                 copyData = new float[6];
88                 System.arraycopy(event.values, 0, copyData, 1, event.values.length);
89                 copyData[0] = System.currentTimeMillis() - startCaptureTime;
90                 this.rotData.add(convertFloatsToDoubles(copyData));
91                 double theta = (Math.acos(event.values[3])) * 2;
92                 double x = event.values[0] / Math.sin(theta / 2);
93                 double y = event.values[1] / Math.sin(theta / 2);
94                 double z = event.values[2] / Math.sin(theta / 2);
95                 double[] vecData = new double[5];
96                 vecData[0] = copyData[0];
97                 vecData[1] = x;
98                 vecData[2] = y;
99                 vecData[3] = z;
100                vecData[4] = theta;
101                this.rotVecData.add(vecData);
102            }
103        }
104        runningTime = (System.currentTimeMillis() - startCaptureTime);
105    }
106}
```

```

1         this.textViewCountDown.setText("Time : " + runningTime / 1000 + " sec");
2     }
3 }
4
5 @Override
6 public void onAccuracyChanged(Sensor sensor, int accuracy) {
7 }
8
9
10 public void onStartButtonClicked(View view) throws InterruptedException {
11     Uri notification = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
12     Ringtone r = RingtoneManager.getRingtone(getApplicationContext(), notification);
13     acceleroData = new ArrayList<double[]>();
14     gyroData = new ArrayList<double[]>();
15     rotData = new ArrayList<double[]>();
16     rotVecData = new ArrayList<double[]>();
17     String time = DateFormat.format("MM-dd-yyyy-h|m|m|ss-aa", System.currentTimeMillis());
18     time.toString();
19     File root = new File(Environment.getExternalStoragePublicDirectory(Environment.
20             DIRECTORY_DOCUMENTS) + File.separator + "DataSensors");
21     root.mkdirs();
22     file = new File(root, FILENAME + time + ".csv");
23     try {
24         file.createNewFile();
25     } catch (IOException e) {
26         e.printStackTrace();
27     }
28     new CountDownTimer(10000, 1000) {
29
30         public void onTick(long millisUntilFinished) {
31             textViewCountDown.setText("Count Down = " + millisUntilFinished / 1000);
32         }
33
34         public void onFinish() {
35             textViewCountDown.setText("Capturing Data !");
36             Uri notification = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION
37                     );
38             Ringtone r = RingtoneManager.getRingtone(getApplicationContext(), notification);
39             r.play();
40             isCapturing = true;
41             getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
42             startCaptureTime = System.currentTimeMillis();
43         }
44     }.start();
45 }
46
47
48 private Sensor sensorGetter(int input) {
49
50     List<Sensor> sensorList = mSensorManager.getSensorList(input);
51     Sensor sensor = null;
52     for (int i = 0; i < sensorList.size(); i++) {
53         sensor = sensorList.get(i);
54     }
55     return sensor;
56 }
57
58 private void writeData(BufferedWriter bw, ArrayList<double[]> data, String title, String
59             columnTitle) throws IOException {
60     bw.write(title);
61     bw.newLine();
62     bw.write(columnTitle);
63     bw.newLine();
64     for (int i = 0; i < data.size(); i++) {
65         for (int j = 0; j < data.get(i).length; j++) {
66             bw.write(data.get(i)[j] + ",");
67         }
68         bw.newLine();
69     }
69     bw.newLine();
70 }
71
72
73 private double[] convertFloatsToDoubles(float[] input)
74 {
75     if (input == null)
76     {
77         return null;
78     }
79     double[] output = new double[input.length];
80     for (int i = 0; i < input.length; i++)
81     {
82         output[i] = input[i];
83     }
84     return output;
85 }
86
87 public void onStopButtonClicked(View view) {
88     isCapturing = false;
89     getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
90     try {
91         BufferedWriter bw = new BufferedWriter(new FileWriter(this.file));
92         writeData(bw, acceleroData, "Accelerometer", "time,x,y,z");
93         writeData(bw, gyroData, "Gyroscope", "time,x,y,z");
94         writeData(bw, rotData, "Rotation_Vector", "time,x*sin(theta/2),y*sin(theta/2),z*sin(
95             theta/2),cos(theta/2),akurasi(dalam_radians)(-1 jika tidak ada)");
96         writeData(bw, rotVecData, "Rotation_Vector_by_Vector_and_Angle", "time,x,y,z,theta");
97         bw.newLine();
98         bw.write("Running Time : " + runningTime + " ms");
99         bw.flush();
100        bw.close();
101        file.createNewFile();
102        textViewCountDown.setText("Data Captured! at " + file.getAbsolutePath());
103        Log.d("File", "File is located at " + file.getAbsolutePath());

```

```

1     } catch (IOException e) {
2         e.printStackTrace();
3     }
4 }
5 }
6 }
```

Listing A.3: SensorAccelerometerActivity.java

```

7 package com.example.egaprianto.testingsensors;
8
9 import android.app.Activity;
10 import android.content.Context;
11 import android.hardware.Sensor;
12 import android.hardware.SensorEvent;
13 import android.hardware.SensorEventListener;
14 import android.hardware.SensorManager;
15 import android.os.Build;
16 import android.os.Bundle;
17 import android.widget.TextView;
18
19 import java.util.List;
20
21 public class SensorAccelerometerActivity extends Activity implements SensorEventListener {
22     private SensorManager mSensorManager;
23     private Sensor mAccelSensor;
24     private TextView textViewXData;
25     private TextView textViewYData;
26     private TextView textViewZData;
27     private TextView textViewDebug;
28     private TextView textViewAccuracy;
29
30     @Override
31     public final void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_accelero_sensor);
34         textViewXData = (TextView) findViewById(R.id.textViewXData);
35         textViewYData = (TextView) findViewById(R.id.textViewYData);
36         textViewZData = (TextView) findViewById(R.id.textViewZData);
37         textViewDebug = (TextView) findViewById(R.id.textViewDebug);
38         textViewAccuracy = (TextView) findViewById(R.id.textViewAccuracy);
39         mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
40         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
41             mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
42             List<Sensor> accelerometerSensors = mSensorManager.getSensorList(Sensor.
43                 TYPE_ACCELEROMETER);
44             mAccelSensor = null;
45             if (accelerometerSensors != null) {
46                 for (int i = 0; i < accelerometerSensors.size(); i++) {
47                     mAccelSensor = accelerometerSensors.get(i);
48                 }
49             }
50         }
51     }
52
53     @Override
54     public final void onAccuracyChanged(Sensor sensor, int accuracy) {
55     }
56
57     @Override
58     public final void onSensorChanged(SensorEvent event) {
59         float x = event.values[0];
60         float y = event.values[1];
61         float z = event.values[2];
62         textViewXData.setText("x=" + x);
63         textViewYData.setText("y=" + y);
64         textViewZData.setText("z=" + z);
65     }
66
67     @Override
68     protected void onResume() {
69         super.onResume();
70         mSensorManager.registerListener(this, mAccelSensor, SensorManagerSENSOR_DELAY_NORMAL);
71     }
72
73     @Override
74     protected void onPause() {
75         super.onPause();
76         mSensorManager.unregisterListener(this);
77     }
78
79 }
80 }
```

Listing A.4: SensorGyroscopeActivity.java

```

81 package com.example.egaprianto.testingsensors;
82
83 import android.app.Activity;
84 import android.content.Context;
85 import android.hardware.Sensor;
86 import android.hardware.SensorEvent;
87 import android.hardware.SensorEventListener;
88 import android.hardware.SensorManager;
89 import android.os.Build;
90 import android.os.Bundle;
91 import android.widget.TextView;
92
93 import java.util.List;
94
95 public class SensorGyroscopeActivity extends Activity implements SensorEventListener {
```

```

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

```


1

LAMPIRAN B

2

KODE PROGRAM APLIKASI GAME

Listing B.1: GyroscopeChecker.cs

```

3  izfusing System.Collections;
4  using System.Collections.Generic;
5  using TheNextFlow.UnityPlugins;
6  using UnityEngine;
7
8  public class GyroscopeChecker : MonoBehaviour {
9
10     // Use this for initialization
11     private void Awake()
12     {
13         //Debug.Log("init");
14
15         if (!SystemInfo.supportsGyroscope)
16         {
17             MobileNativePopups.OpenAlertDialog(
18                 "No_Gyroscope", "Your_device_didn't_have_Gyroscope_sensor,_you_can't_play_this_
19                 game!", "OK", null);
20             Application.Quit();
21         }
22     }
23 }
24 }
```

Listing B.2: GameplayStatistic.cs

```

25 izf
26 public class GameplayStatistic {
27
28     public static bool isAnswering = false;
29 }
```

Listing B.3: GameOverController.cs

```

30 izfusing System;
31 using UnityEngine;
32 using UnityEngine.UI;
33
34 public class GameOverController : MonoBehaviour, IGvrGazeResponder {
35
36     public QuestionMessageController[] itemMessages;
37     public Image pointer;
38     public Animator animator;
39
40     public void OnGazeEnter()
41     {
42         //Do nothing
43     }
44
45     public void OnGazeExit()
46     {
47         //Do nothing
48     }
49
50     public void OnGazeTrigger()
51     {
52         //print("trigger");
53         restartGame();
54     }
55     public void restartGame()
56     {
57         //print("restart");
58         foreach(QuestionMessageController itemMessage in itemMessages)
59         {
60             itemMessage.resetAll();
61         }
62         animator.SetBool("poppedOut", false);
63         pointer.transform.Rotate(new Vector3(0,0,-1*pointer.transform.rotation.eulerAngles.z));
64     }
65     // Use this for initialization
66     void Start () {
67
68     }
69
70     // Update is called once per frame
```

```

1 void Update ()
2 {
3     //print(Math.Abs(pointer.transform.rotation.eulerAngles.z));
4     //RectTransform gameOverTransform= this.GetComponent<RectTransform>();
5     //gameOverTransform.rotation.eulerAngles.Set(gameOverTransform.rotation.eulerAngles.x,
6         gameOverTransform.rotation.eulerAngles.y,0);
7     if (isGameOver())
8     {
9         animator.SetBool("poppedOut", true);
10        haltAllGameplay();
11    }
12 }
13 void LateUpdate()
14 {
15     GvrViewer.Instance.UpdateState();
16     if (GvrViewer.Instance.BackButtonPressed)
17     {
18         Application.Quit();
19     }
20 }
21
22 public bool isGameOver()
23 {
24     return Math.Abs(pointer.transform.rotation.eulerAngles.z) >= 59 && Math.Abs(pointer.
25     transform.rotation.eulerAngles.z) <= 309;
26 }
27 public void haltAllGameplay()
28 {
29     foreach (QuestionMessageController itemMessage in itemMessages)
30     {
31         itemMessage.stopWorking();
32     }
33 }
34 }

```

Listing B.4: QuestionMessageController.cs

```

35 // Izf
36 using System;
37 using TheNextFlow.UnityPlugins;
38 using UnityEngine;
39 using UnityEngine.UI;
40
41 public class QuestionMessageController : MonoBehaviour, IGvrGazeResponder, HeadMotionListener
42 {
43     public Canvas messageCanvas;
44     public Canvas notificationCanvas;
45     public String[] questions;
46     public bool[] answers;
47     public Image pointer;
48     private Animator animatorMessage;
49     private Animator animatorNotification;
50     private Text messageText;
51     private bool startCounting;
52     private bool isDetecting;
53     private bool clickable;
54     private float timeAvailable;
55     private MotionRecorder mMotionRecorder;
56     private int selectedIndexQuestions;
57
58     public void OnGazeEnter()
59     {
60     }
61
62     public void OnGazeExit()
63     {
64     }
65
66     public void OnGazeTrigger()
67     {
68         if (clickable && !GameplayStatistic.isAnswering)
69         {
70             Input.gyro.enabled = true;
71             Input.gyro.updateInterval = 0.00002F;
72             mMotionRecorder = new MotionRecorder();
73             mMotionRecorder.addHeadMotionListener(this);
74             isDetecting = true;
75             selectedIndexQuestions = (int)(UnityEngine.Random.Range(0.0f, 0.9999f) * questions.
76             Length);
77             messageText.text = questions[selectedIndexQuestions];
78             animatorMessage.SetBool("poppedOut", !animatorMessage.GetBool("poppedOut"));
79             animatorNotification.SetBool("poppedOut", !animatorNotification.GetBool("poppedOut"));
80             clickable = false;
81             GameplayStatistic.isAnswering = true;
82         }
83     }
84
85     // Use this for initialization
86     void Start() {
87         timeAvailable = UnityEngine.Random.Range(5.0f, 10.0f);
88         clickable = false;
89         animatorMessage = messageCanvas.GetComponentInChildren<Animator>();
90         animatorNotification = notificationCanvas.GetComponentInChildren<Animator>();
91         messageText = messageCanvas.GetComponentInChildren<Text>();
92         startCounting = true;
93     }
94
95     // Update is called once per frame
96     void Update() {
97         if (startCounting)
98         {
99             timeAvailable -= Time.deltaTime;

```

```

1      }
2      if (timeAvailable < 0)
3      {
4          startQuestion();
5      }
6      if (isDetecting)
7      {
8          this.mMotionRecorder.inputGryoData(Input.gyro.rotationRate.y, Input.gyro.rotationRate.
9              x);
10         if (Input.GetKeyDown("d")) debugOnNodMotionDetected();
11         if (Input.GetKeyDown("f")) debugOnShookMotionDetected();
12     }
13 }
14
15 public void resetAll()
16 {
17     print("resetting _object");
18     animatorMessage.SetBool("poppedOut", false);
19     animatorNotification.SetBool("poppedOut", false);
20     startCounting = true;
21     isDetecting = false;
22     clickable = false;
23     timeAvailable = UnityEngine.Random.Range(5.0f, 10.0f);
24     mMotionRecorder = null;
25 }
26
27 public void stopWorking()
28 {
29     isDetecting = false;
30     clickable = false;
31 }
32 public void debugOnNodMotionDetected()
33 {
34     this.onMotionDetected(Motion.NOD);
35 }
36
37 public void debugOnShookMotionDetected()
38 {
39     this.onMotionDetected(Motion.SHOOK);
40 }
41
42 public void startQuestion()
43 {
44     startCounting = false;
45     animatorNotification.SetBool("poppedOut", !animatorNotification.GetBool("poppedOut"));
46     clickable = true;
47     timeAvailable = UnityEngine.Random.Range(5.0f, 10.0f);
48 }
49
50 public void onMotionDetected(Motion motion)
51 {
52     if (isDetecting)
53     {
54         if (motion == Motion.NOD)
55         {
56             messageText.text = "YES";
57             if (answers[selectedIndexQuestions])
58             {
59                 pointer.transform.Rotate(new Vector3(0, 0, 10));
60             }
61             else
62             {
63                 pointer.transform.Rotate(new Vector3(0, 0, -10));
64             }
65         }
66         else
67         {
68             messageText.text = "NO";
69             if (!answers[selectedIndexQuestions])
70             {
71                 pointer.transform.Rotate(new Vector3(0, 0, 10));
72             }
73             else
74             {
75                 pointer.transform.Rotate(new Vector3(0, 0, -10));
76             }
77         }
78         clickable = false;
79         startCounting = true;
80         GameplayStatistic.isAnswering = false;
81         animatorMessage.SetBool("poppedOut", false);
82         Input.gyro.enabled = false;
83         isDetecting = false;
84         mMotionRecorder.removeHeadMotionListener(this);
85         mMotionRecorder = null;
86     }
87 }
88 }
```

Listing B.5: Axis.cs

```

89 | if public enum Axis
90 | {
91 |     X,
92 |     Y
93 | }
```

Listing B.6: Detector.cs

```

94 | if public abstract class Detector {
95 | }
```

```

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

```

Listing B.7: HeadMotionListener.cs

```

24 interface HeadMotionListener
25 {
26     void onMotionDetected(Motion motion);
27 }

```

Listing B.8: Motion.cs

```

28 enum Motion
29 {
30     NOD,
31     SHOOK
32 }

```

Listing B.9: MotionRecorder.cs

```

33 using System.Collections.Generic;
34 using UnityEngine;
35
36
37 public class MotionRecorder {
38
39     private PulseFactory pulseFactoryXAxis;
40     private PulseFactory pulseFactoryYAxis;
41     private Detector nodDetector;
42     private Detector shookDetector;
43
44     private LinkedList<HeadMotionListener> listListener;
45
46     public MotionRecorder()
47     {
48         listListener = new LinkedList<HeadMotionListener>();
49         pulseFactoryXAxis = new PulseFactory(Axis.X);
50         pulseFactoryYAxis = new PulseFactory(Axis.Y);
51         nodDetector = new NodDetector(700, 2, 0.25);
52         shookDetector = new ShookDetector(700, 2, 0.25);
53     }
54
55     public void addHeadMotionListener(HeadMotionListener listener)
56     {
57         listListener.AddFirst(listener);
58     }
59
60     public void removeHeadMotionListener(HeadMotionListener listener)
61     {
62         listListener.Remove(listener);
63     }
64
65     public void inputGryoData(float x, float y)
66     {
67         Pulse xPulse = pulseFactoryXAxis.inputData(x);
68         Pulse yPulse = pulseFactoryYAxis.inputData(y);
69         if (xPulse != null)
70         {
71             Debug.Log("PULSE");
72             bool shookDetected = shookDetector.addPulse(xPulse);
73             if (shookDetected && !nodDetector
74                 .isMightHaveActivites())
75                 // terdeteksi gelangan tanpa ada pergerakan pada sumbu y
76                 notifyAllMotionListener(Motion.SHOOK);
77                 //Log.d("MotionDetectedPulsesX", shookDetector.toString());
78         }
79         if (yPulse != null)
80         {
81             bool nodDetected = nodDetector.addPulse(yPulse);
82             if (nodDetected && !shookDetector
83                 .isMightHaveActivites())
84                 // terdeteksi gelangan tanpa ada pergerakan pada sumbu x
85                 //Log.d("MotionDetectedPulsesY", shookDetector.toString());
86         }
87     }
88 }

```

```

1 |     public void notifyAllMotionListener(Motion motion)
2 |     {
3 |         foreach (HeadMotionListener motionListener in listListener)
4 |         {
5 |             motionListener.onMotionDetected(motion);
6 |         }
7 |     }
8 |

```

Listing B.10: NodDetector.cs

```

9 | if using System;
10 | public class NodDetector : Detector
11 | {
12 |     /*
13 |     private double limitTime; // in milliseconds
14 |     private float limitSpeed = 2;
15 |     private double limitSimpangan = 0.25;
16 |     */
17 |     public NodDetector(double limitTime, float limitSpeed, double limitSimpangan) : base(limitTime,
18 |         limitSpeed, limitSimpangan)
19 |     {
20 |
21 |         this.mightHaveActivites = false;
22 |         this.listPulseToDetect = new Pulse[2];
23 |         this.listPulseToDetect[0] = PulseFactory.STANDARD_PULSE; // firstPulse
24 |         this.listPulseToDetect[1] = PulseFactory.STANDARD_PULSE; // lastPulse
25 |     }
26 |
27 |     public override bool addPulse(Pulse yPulse)
28 |     {
29 |         bool result = false;
30 |         this.listPulseToDetect[0] = yPulse;
31 |         if (Math.Abs(this.listPulseToDetect[0].getPeak()) > limitSpeed)
32 |         {
33 |             mightHaveActivites = true;
34 |             bool isPassLimitSimpangan = Math.Abs(this.listPulseToDetect[0].getSimpangan()) >
35 |                 limitSimpangan
36 |                 && Math.Abs(listPulseToDetect[1].getSimpangan()) > limitSimpangan;
37 |             bool isPassLimitTimeRange =
38 |                 listPulseToDetect[0].getRangeTime() < limitTime && listPulseToDetect[1].
39 |                     getRangeTime() < limitTime;
40 |             bool isDifferentTypePulse = this.listPulseToDetect[0].getType() != listPulseToDetect
41 |                 [1].getType();
42 |             result = isPassLimitSimpangan && isPassLimitTimeRange && isDifferentTypePulse;
43 |         }
44 |         else
45 |         {
46 |             if (Math.Abs(listPulseToDetect[1].getPeak()) < limitSpeed)
47 |             {
48 |                 mightHaveActivites = false;
49 |             }
50 |         }
51 |         this.listPulseToDetect[1] = this.listPulseToDetect[0];
52 |         return result;
53 |     }
54 | }
55 |

```

Listing B.11: Pulse.cs

```

56 | if using System;
57 |
58 | public class Pulse
59 | {
60 |     private double simpangan;
61 |     private double rangeTime;
62 |     private double startTime;
63 |     private double endTime;
64 |     private float peak;
65 |     private PulseType type;
66 |     private Axis axis;
67 |
68 |     public Pulse(double simpangan, double startTime, double endTime, float peak, PulseType type,
69 |                 Axis axis)
70 |     {
71 |         this.simpangan = simpangan;
72 |         this.startTime = startTime;
73 |         this.endTime = endTime;
74 |         this.rangeTime = endTime - startTime;
75 |         this.peak = peak;
76 |         this.type = type;
77 |         this.axis = axis;
78 |     }
79 |
80 |     public double getSimpangan()
81 |     {
82 |         return simpangan;
83 |     }
84 |
85 |     public double getRangeTime()
86 |     {
87 |         return rangeTime;
88 |     }
89 |
90 |     public double getStartTime()
91 |     {
92 |         return startTime;
93 |     }
94 |
95 |     public double getEndTime()

```

```

1     {
2         return endTime;
3     }
4
5     public float getPeak()
6     {
7         return peak;
8     }
9
10    public PulseType getType()
11    {
12        return type;
13    }
14
15    public Axis getAxis()
16    {
17        return axis;
18    }
19 }
```

Listing B.12: PulseFactory.cs

```

20  using System;
21
22  public class PulseFactory {
23
24      private long lastTime;
25      private float lastData;
26      private double simpangan;
27      private long rangeTime;
28      private double startTime;
29      private double endTime;
30      private float peak;
31      private PulseType currentPulseType;
32      private Axis axis;
33
34      public static Pulse STANDARD_PULSE = new Pulse(0, 0, 0, 0, PulseType.HILL, Axis.X);
35
36      public PulseFactory(Axis axis)
37      {
38          this.axis = axis;
39      }
40
41      private void resetAttributes(PulseType type)
42      {
43          this.simpangan = 0;
44          this.peak = 0;
45          this.currentPulseType = type;
46      }
47
48      private Pulse finishPulse(float data, double currentTime)
49      {
50          this.endTime = ((-lastData / (data - lastData)) * (currentTime - lastTime)) + lastTime;
51          double areaBeforeIntersect = ((endTime - lastTime) / 1000) * lastData / 2;
52          this.simpangan += areaBeforeIntersect;
53          Pulse newPulse = new Pulse(this.simpangan, this.startTime, this.endTime, this.peak,
54                                      this.currentPulseType,
55                                      this.axis);
56          this.startTime = endTime;
57          return newPulse;
58      }
59
60      public Pulse inputData(float data)
61      {
62          Pulse result = null;
63          long currentTime = DateTime.Now.Millisecond;
64
65          if (currentPulseType == PulseType.HILL)
66          {
67              if (peak < data)
68              {
69                  peak = data;
70              }
71              if (data < 0)
72              {//if values intersect data=0
73                  result = finishPulse(data, currentTime);
74                  resetAttributes(PulseType.VALLEY);
75              }
76          }
77          else
78          {
79              this.simpangan += ((lastData + data) / 1000) * (currentTime - lastTime) / 2;
80          }
81          else if (currentPulseType == PulseType.VALLEY)
82          {
83              if (peak > data)
84              {
85                  peak = data;
86              }
87              if (data > 0)
88              {//if values intersect data=0
89                  result = finishPulse(data, currentTime);
90                  resetAttributes(PulseType.HILL);
91              }
92          }
93          else
94          {
95              this.simpangan += ((lastData + data) / 1000) * (currentTime - lastTime) / 2;
96          }
97
98          lastData = data;
99          lastTime = currentTime;

```

```

1 |     return result;
2 | }
3 |

```

Listing B.13: PulseType.cs

```

4 | if public enum PulseType
5 | {
6 |     HILL,
7 |     VALLEY,
8 |     INIT
9 | }

```

Listing B.14: ShookDetector.cs

```

10 | if using System;
11 |
12 | public class ShookDetector : Detector
13 | {
14 |     /*
15 |     private double limitTime; // in millisecons
16 |     private float limitSpeed = 2;
17 |     private double limitSimpangan = 0.25;
18 |     */
19 |     public ShookDetector(double limitTime, float limitSpeed, double limitSimpangan) : base(
20 |         limitTime, limitSpeed, limitSimpangan)
21 |     {
22 |         mightHaveActivites = false;
23 |         listPulseToDetect = new Pulse[3];
24 |         listPulseToDetect[0] = PulseFactory.STANDARD_PULSE;
25 |         listPulseToDetect[1] = PulseFactory.STANDARD_PULSE;
26 |         listPulseToDetect[2] = PulseFactory.STANDARD_PULSE;
27 |     }
28 |
29 |     public override bool addPulse(Pulse xPulse)
30 |     {
31 |         bool result = false;
32 |         listPulseToDetect[2] = listPulseToDetect[1];
33 |         listPulseToDetect[1] = listPulseToDetect[0];
34 |         listPulseToDetect[0] = xPulse; //currentPulse
35 |         if (Math.Abs(listPulseToDetect[0].getPeak()) > limitSpeed)
36 |         {
37 |             mightHaveActivites = true;
38 |             bool isPassLimitSimpangan = Math.Abs(listPulseToDetect[0].getSimpangan()) >
39 |                 limitSimpangan
40 |                 && Math.Abs(listPulseToDetect[1].getSimpangan()) > limitSimpangan
41 |                 && Math.Abs(listPulseToDetect[2].getSimpangan()) > limitSimpangan;
42 |             bool isPassLimitTimeBetween =
43 |                 (listPulseToDetect[1].getStartTime() - listPulseToDetect[2].getEndTime() == 0) &&
44 |                 (
45 |                     listPulseToDetect[0].getStartTime() - listPulseToDetect[1].getEndTime() == 0);
46 |             bool isPassLimitTimeRange =
47 |                 listPulseToDetect[0].getRangeTime() < limitTime && listPulseToDetect[1].
48 |                     getRangeTime() < limitTime
49 |                     && listPulseToDetect[2].getRangeTime() < limitTime;
50 |             bool isDifferentTypePulse =
51 |                 (listPulseToDetect[1].getType() != listPulseToDetect[2].getType()) && (
52 |                     listPulseToDetect[0].getType() != listPulseToDetect[1]
53 |                     .getType());
54 |             result = isPassLimitSimpangan && isPassLimitTimeBetween && isDifferentTypePulse
55 |                 && isPassLimitTimeRange;
56 |         }
57 |         else
58 |         {
59 |             if (Math.Abs(listPulseToDetect[2].getPeak()) < limitSpeed
60 |                 && Math.Abs(listPulseToDetect[1].getPeak()) < limitSpeed)
61 |             {
62 |                 mightHaveActivites = false;
63 |             }
64 |         }
65 |     return result;
66 | }
67 |
68 | }

```