SKRIPSI

PENDETEKSI GERAKAN KEPALA DENGAN GOOGLE CARDBOARD

EGA PRIANTO

NPM: 2013730047

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

UNDERGRADUATE THESIS

HEAD MOTION DETECTOR USING GOOGLE CARDBOARD

EGA PRIANTO

NPM: 2013730047

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

LEMBAR PENGESAHAN

PENDETEKSI GERAKAN KEPALA DENGAN GOOGLE CARDBOARD

EGA PRIANTO

NPM: 2013730047

Bandung, 20 «bulan» 2016

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Pascal Alfadian, M.Comp.	**«pembimbing pendamping/2»**
Ketua Tim Penguji	**Anggota Tim Penguji**
«penguji 1»	**«penguji 2»**

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

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

PENDETEKSI GERAKAN KEPALA DENGAN GOOGLE CARDBOARD

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

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

Dinyatakan di Bandung,
Tanggal 20 «bulan» 2016

Meterai

Ega Prianto
NPM: 2013730047

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

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

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

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

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

KATA PENGANTAR

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

Bandung, «bulan» 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Virtual Reality 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] *Virtual Reality* membuat pengalaman sensorik, di antaranya penglihatan, pendengaran, perabaan, dan penciuman secara buatan.[2] Gawai *Virtual Reality* terbaru sekarang yaitu dengan menggunakan *head-mounted display*, Google Cardboard salah satunya. *Head-mounted display* adalah menempatkan layar di kepala, sehingga pengguna hanya dapat melihat tampilan yang ditampilkan oleh layar.[3]

Google Cardboard[4] adalah gawai murah yang terbuat dari kardus untuk dapat merasakan pengalaman *virtual reality* dengan *smartphone* Android atau iOS. Kita dapat membuat Google Cardboard kita sendiri secara gratis dengan mengunduh templatenya di situs web Google Cardboard. [4]Template tersebut membantu dalam merakit kardus dengan dibentuk, dilipat dan digunting sedemikian rupa sehingga berbentuk kacamata. Bahan-bahan untuk merakit Google Cardboard hanyalah kardus, lem, dan lensa dengan spesifikasi tertentu.

Pada Gawai Google Cardboard cara pengguna memberikan *input* kepada program sangatlah terbatas. Cara tersebut hanyalah dengan gerakan kepala dan tombol magnet. Tombol magnet ini pun terkadang tidak berfungsi dengan baik, karena bergantung pada medan magnet yang di deteksi oleh *smartphone* yang digunakan. Cara lainnya agar dapat memberikan *input* kepada program adalah dengan menghubungkan *smartphone* yang digunakan dengan *bluetooth controller*.

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

Agar *Virtual Reality* 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 *Virtual Reality* akan tidak akurat atau lambat. *Magnetometer* digunakan untuk mengetahui arah pandang pengguna. *Accelerometer* digunakan untuk mengetahui arah gaya gravitasi.[6] *Gyroscope* digunak-

an 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 *Virtual Reality* dengan Google Cardboard dan *smartphone*, tetapi juga dapat berfungsi sebagai pendeteksi gerakan kepala.

1.2 Rumusan Masalah

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

1.3 Tujuan

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

1.4 Batasan Masalah

Penelitian ini dibuat berdasarkan batasan-batasan sebagai berikut:

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

1.5 Metode Penelitian

Berikut adalah metode penelitian yang digunakan dalam penelitian ini:

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

1.6 Sistematika Penulisan

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

1. Bab 1: Pendahuluan, yaitu membahas mengenai gambaran umum penelitian ini. Berisi tenang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.

2. Bab 2: Dasar Teori, yaitu membahas teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang Android SDK, Google VR SDK, Quaternion, dan algoritma *Head Motion Detection*.

3. Bab 3: Analisis, yaitu membahas mengenai analisa masalah. Berisi tentang analisis aplikasi-aplikasi sejenis, analisis grafik dari sensor-sensor pada *smartphone* ketika mengangguk dan menggeleng, analisis metode pendeteksi gerakan kepala.

4. Bab 4: Perancangan yaitu membahas mengenai perancangan

BAB 2

DASAR TEORI

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

2.1 Android SDK

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

2.1.1 Struktur File Android Studio Project

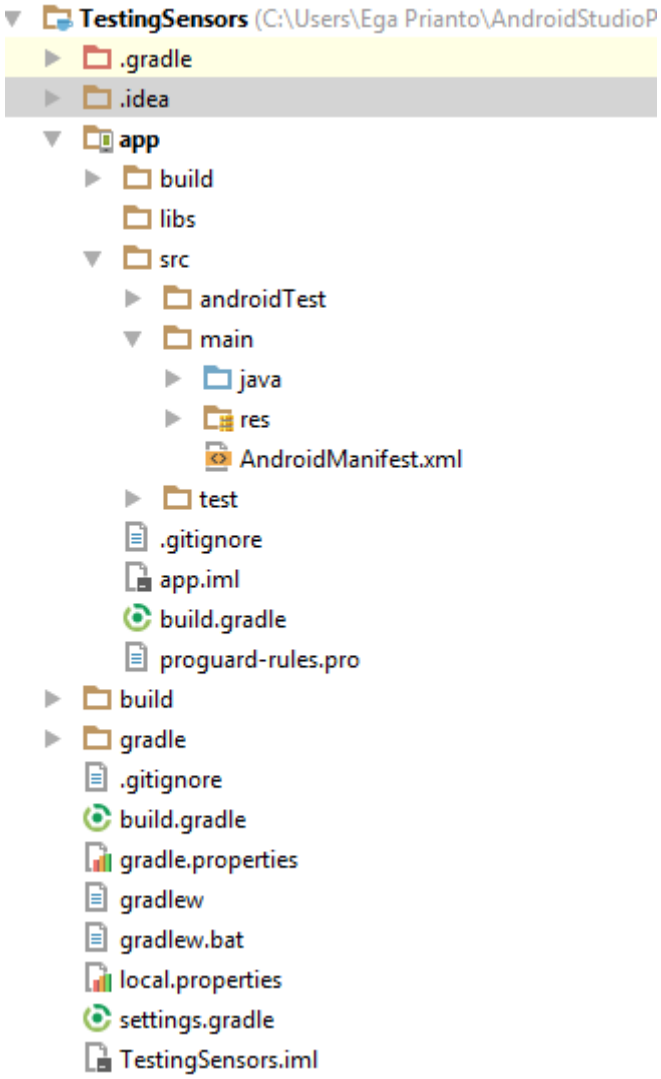
Pada saat **project** baru telah dibuat, Android Studio akan membuatkan folder-folder standar (Gambar 2.1). Berikut adalah sebagian penjelasan dari hal yang perlu di perhatikan pada struktur tersebut:

- Folder ***module*/build** mengandung file-file dari hasil pembangunan *project*.
- Folder ***module*/libs** mengandung *libraries* privat.
- Folder ***module*/src** mengandung semua kode dan file sumber untuk suatu modul yang terbagi menjadi *subdirectories* berikut:
  - Folder **androidTest**/ mengandung kode untuk mengetes yang berjalan di perangkat Android.
  - Folder **main**/ mengandung file-file kode dan file sumber inti dari suatu module yang terbagi menjadi *subdirectories* berikut:
    - File **AndroidManifest.xml** mendeskripsikan sifat dari aplikasi dan setiap komponennya.
    - Folder **java**/ mengandung kode java.
    - Folder **jni**/ mengandung kode yang menggunakan Java Native Interface (JNI).
    - Folder **gen**/ mengandung file java yang dihasilkan oleh Android Studio.
    - Folder **res**/ mengandung file-file sumber untuk aplikasi, seperti file drawable, layout, dan UI String.

    * Folder **assets**/ mengandung file yang akan di kompile menjadi file **.apk**. File-file yang biasa disimpan di folder ini adalah file audio, video, file html, gambar dan file bantu lainnya.
  - File **build.gradle**(module) mendefinisikan konfigurasi modul untuk proses *build*.

- File **build.gradle**(project) mendefinisikan konfigurasi proses **build** untuk *project* yang berlaku untuk semua modul.

Folder **App** pada Gambar 2.1 merupakan folder *module*.



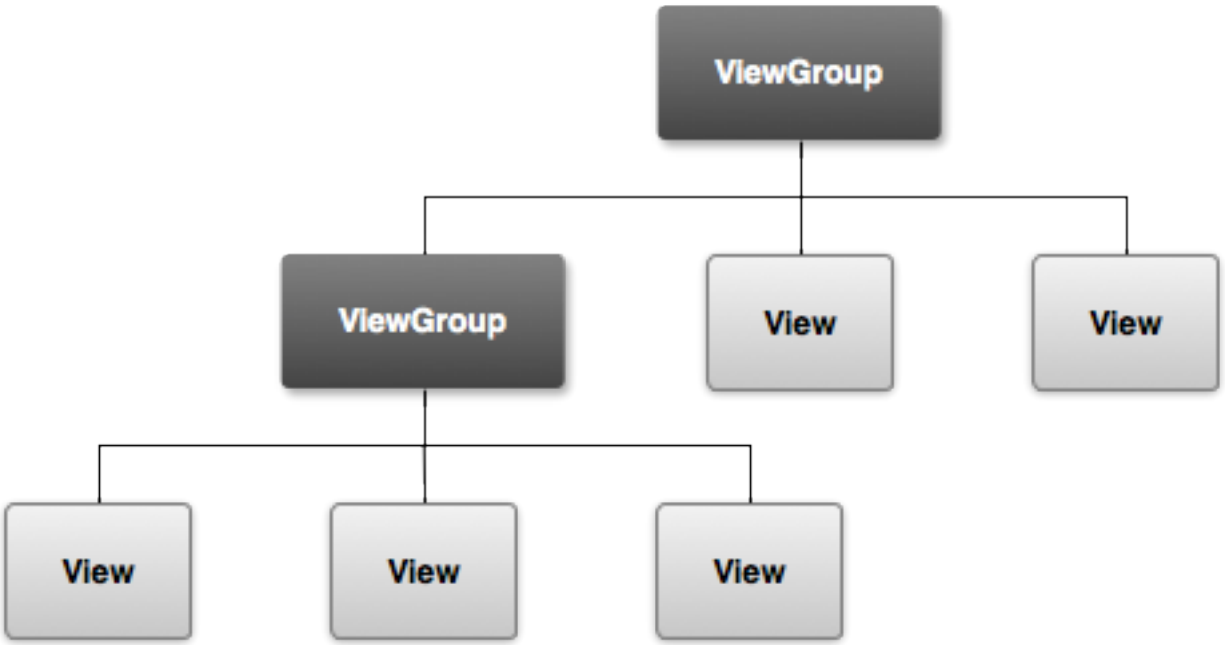
Gambar 2.1: Tampilan struktur folder pada *project* Android Studio

2.1.2 Membuat User Interface

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

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

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



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

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

Attribut-attribut Objek View

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

- **android:id** Attribut ini merupakan pengidentifikasi dari suatu view. Attribut ini dapat digunakan untuk menjadi referrensi object dari kode aplikasi seperti membaca dan memanipulasi objek tersebut (Akan dijelaskan lebih lanjut pada subbab 2.1.3). Tanda '@' dibutuhkan ketika mereferensi object dari suatu XML. Tanda '@' tersebut diikuti dengan tipe (id pada kasus ini), *slash*, dan nama (edit_message pada kode 2.2). Tanda tambah (+) sebelum tipe hanya dibutuhkan jika ingin mendefinisikan *resource ID* untuk pertama kalinya.

- **android:layout_width** dan **android:layout_height** Attribut ini digunakan untuk mendefinisikan panjang dan lebar dari suatu objek View. Daripada menggunakan besar spesifik untuk panjang dan lebarnya, lebih baik menggunakan "wrap_content" yang menspesifikasi viewnya hanya akan sebesar yang dibutuhkan untuk memuat konten-konten dari View. Ji-

ka menggunakan "match_parent" pada kasus kode 2.2 View akan memenuhi layar, karena besarnya akan mengikuti besar dari paretnya LinearLayout.

- **android:hint** Attribut ini merupakan *default string* untuk di tampilkan ketika objek View kosong. Daripada menggunakan *hard-coded string* sebagai *nilai* untuk ditampilkan, *value* "@string/edit_message" mereferensi ke sumber string pada file yang berbeda. Karena mereferensi ke sumber konkrit, maka tidak dibutuhkan tanda tambah (+). Nilai string ini akan di simpan pada file Strings.xml yang ditunjukkan pada Kode 2.1.

Listing 2.1: Contoh kode pada string.xml

```
<resources>
<string name="app_name">MyFirstAndroidApp</string>
<string name="edit_message">Ini adalah hint</string>
<string name="button_send">Send</string>
</resources>
```

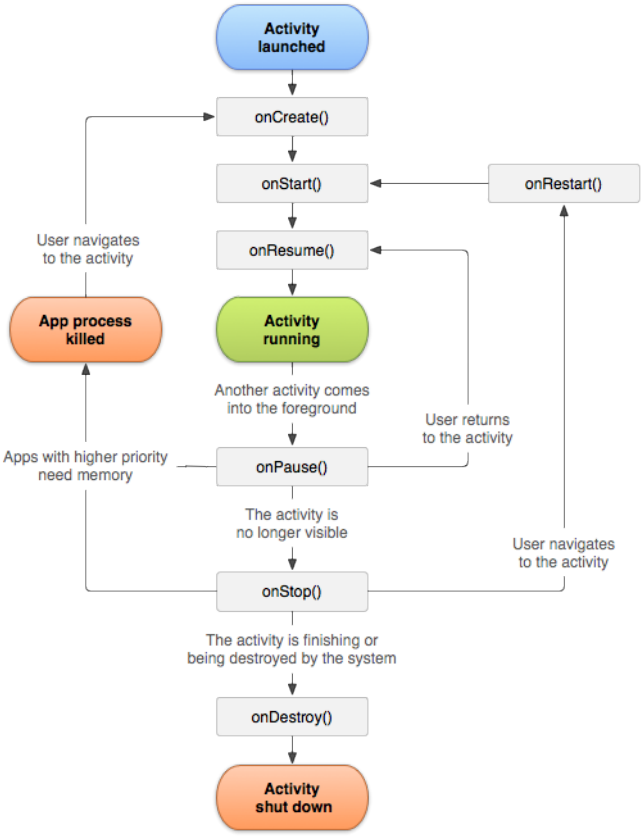
- **android:onClick** Attribut ini akan memberitahu *system* untuk memanggil method yang sesuai namanya (contoh pada kode 2.2 adalah **sendMessage()**) di Activity ketika pengguna melakukan klik pada *button* tersebut. Agar *system* dapat memanggil method yang tepat, method tersebut harus memenuhi kriteria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

Listing 2.2: Contoh kode file XML pada folder layout

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
              android:layout_weight="1"
              android:layout_width="0dp"
              android:layout_height="wrap_content"
              android:hint="@string/edit_message" />
    <Button
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="@string/button_send"
              android:onClick="sendMessage" />
</LinearLayout>
```

2.1.3 Activity

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



Gambar 2.3: *State diagram* siklus Activity

Activity Lifecycle

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

Activity didasari dari empat kondisi:

- Jika Activity berada di latar depan pada layar, Activity tersebut sedang aktif.
- Jika Activity sudah tidak terfokuskan tetapi masih dapat terlihat, Activity tersebut sedang berhenti sementara. Pada kondisi ini Activity tersebut masih berjalan, tapi bisa diberhentikan ketika system berada dalam situasi kekurangan memori.
- Jika suatu Activity benar-benar dihalangi oleh Activity lain, Activity tersebut telah berhenti. Activity tersebut akan tetap mengingat seluruh keadaan dan infomasi anggota tetapi, Activity tersebut tidak lagi terlihat oleh pengguna jadi tampilan jendelanya akan tersembunyi dan seringkali akan diberhentikan Activitynya ketika system membutuhkan memori.
- Jika suatu Activity sedang berhenti sementara atau berhenti total, sistem dapat membuang Activity dari memory dengan cara menanyakan kepada pengguna untuk memberhentikannya atau langsung diberhentikan oleh sistem. Jika Activity tersebut ditampilkan lagi kepada pengguna, Activity tersebut harus memulai dari awal dan kembali ke keadaan sebelumnya.

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

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

2.1.4 Android Sensor Framework

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

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

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

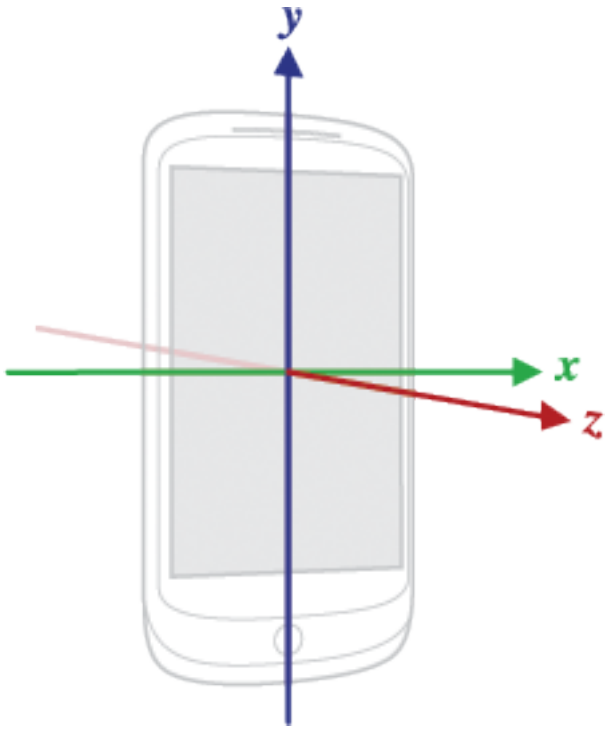
Sistem Koordinat Sensor

Pada umumnya, sensor framework menggunakan sistem tiga sumbu koordinat standar untuk mengekspresikan nilai data. Sebagian besar sensor sistem koordinat didefinisikan relatif terhadap layar

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.

perangkat bila perangkat dibuat dalam orientasi standar (lihat Gambar 2.4) Sensor-sensor yang



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

menggunakan sistem tiga sumbu seperti Gambar 2.4 adalah sebagai berikut :

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

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

Struktur Nilai yang Dikembalikan oleh Sensor

Nilai dari sensor akan didapatkan ketika ada perubahan nilai pada sensor. Setiap sensor memiliki ketelitian perubahan nilai yang berbeda-beda. Nilai ini akan didapatkan dengan tipe data array of float. Besar dan isi dari array tergantung pada sensor yang sedang di pantau.

TYPE_ACCELEROMETER

Semua nilai didefinisikan sebagai satuan m/s^2

- values[0]: Percepatan yang terjadi pada sumbu x dikali -1

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

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

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

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 di butuhkan untuk mengukur percepatan asli yang terjadi pada perangkat, sehingga kontribusi gravitasi harus di eliminasi. Hal ini bisa dilakukan dengan menerapkan *high-pass* filter. Sebaliknya, *low-pass* filter dapat digunakan untuk mendapatkan nilai gravitasi saja.

Listing 2.3: Implementasi *low-pass* filter

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

Low-pass filter dapat diimplementasikan pada kode 2.3

TYPE_MAGNETIC_FIELD

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

TYPE_GYROSCOPE

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

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

Biasanya keluaran dari gyroscope terintegrasi dari waktu ke waktu untuk menghitung rotasi yang menggambarkan perubahan sudut atas langkah waktu, misalnya pada kode 2.4

Listing 2.4: contoh implementasi gyroscope

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

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

TYPE_GRAVITY

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

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

TYPE_LINEAR_ACCELERATION

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

$$percepatan = gravitasi + percepatanlinear$$

TYPE_ORIENTATION
Semua nilai adalah sudut dalam derajat.

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

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

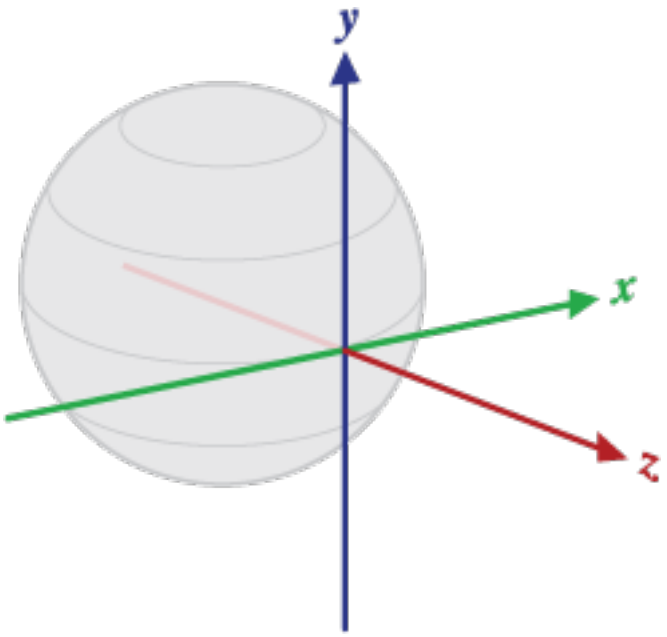
Catatan: Sensor ini sudah tidak digunakan lagi(deprecated), yang digunakan sekarang adalah sensor rotasi vector.

TYPE_ROTATION_VECTOR
Sensor ini merepresentasikan orientasi perangkat dengan kombinasi dari sumbu dan sudut. Perangkat akan di putar sebesar sudut θ mengelilingi sumbu x, y, z. Tiga elemen dari vektor rotasi adalah $(x\sin(\frac{\theta}{2}), y\sin(\frac{\theta}{2}), z\sin(\frac{\theta}{2}))$, sehingga besarnya vektor rotasi sama dengan $\sin(\frac{\theta}{2})$, dan arah vektor rotasi sama dengan sumbu rotasi. Tiga elemen dari vektor rotasi sama dengan tiga komponen terakhir pada unit quaternion$(\cos(\frac{\theta}{2}), x\sin(\frac{\theta}{2}), y\sin(\frac{\theta}{2}), z\sin(\frac{\theta}{2}))$ yang dijelaskan pada subbab 2.4. Elemen dari vektor rotasi tak memiliki satuan. Sistem koordinat yang digunakan sama dengan sistem koordinat yang digunakan pada sensor percepatan. Referensi koordinat didefinisikan sebagai basis orthonormal, yaitu:

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

2.2 Google VR SDK

Google VR SDK[8] digunakan untuk membantu dalam pembuatan aplikasi Virtual Reality pada *smartphone*. Google VR SDK memberikan beberapa fitur sebagai berikut :



Gambar 2.5: Sistem koordinat sensor rotasi vektor terhadap Bumi

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

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

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

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

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

2.2.1 API Audio

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

- Metode pertama dikenal sebagai *Sound Object rendering*. Metode ini memungkinkan pengguna membuat sumber suara virtual dalam ruang tiga dimensi.
- Metode kedua memungkinkan pengguna untuk memutar kembali rekaman *Ambisonic soundfield*. Rekaman *Ambisonic soundfield* adalah file audio *multi-channel* yang telah terspasialisasi.

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

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

2.2.2 API Base

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

- AndroidCompat
  Kelas ini merupakan kelas utilitas untuk menggunakan fitur VR. Fitur-fitur ini mungkin tidak tersedia pada semua versi android. Kelas ini memiliki method-method sebagai berikut:
  - setSustainedPerformanceMode(Activity activity, boolean enabled):
    Method ini digunakan untuk mengubah window android ke mode performa secara berkelanjutan.
  - public static void setVrModeEnabled (Activity activity, boolean enabled):
    Mengatur pengaturan yang tepat untuk "mode VR" pada suatu Activity. Method ini tidak digunakan karena hanya dapat digunakan pada Android N+.
  - public static boolean trySetVrModeEnabled (Activity activity, boolean enabled): Method ini kegunaanya sama dengan method **setVrModeEnabled (Activity activity, boolean enabled)**, namun mengembalikan boolean true jika berhasil dan sebaliknya.

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

- GvrActivity
  Kelas ini merupakan Activity dasar yang menyediakan integrasi yang mudah dengan headset Google VR. Kelas ini mengekspos kejadian untuk berinteraksi dengan headset Google VR dan menangani detil-detil yang biasa diperlukan saat membuat suatu Activity untuk perenderan VR. Activity ini membuat layar tetap menyala selama perangkat android bergerak. Jika tidak ada pergerakan dari perangkat android maka layar reguler (*wakeclock*) akan ditampilkan. Pada kelas ini terdapat method **onCardboardTrigger ()** untuk mendeteksi ketika Cardboard Trigger sedang ditarik dan dilepaskan (Magnet yang berada pada sisi Google Cardboard).

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

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

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

  - **public void onNewFrame(HeadTransform headTransform)**
    method ini terpanggil ketika Framebaru akan digambar. Method ini memungkinkan untuk membedakan antara perenderan pandangan mata dan frame-frame yang berbeda. Setiap operasi per-frame harus tidak spesifik pada satu tampilan saja.
  - **public abstract void onDrawEye (Eye eye)**
    Method ini meminta untuk menggambar suatu konten dari sudut pandang mata.
  - **public abstract void onFinishFrame (Viewport viewport)**
    Method ini dipanggil ketika suatu frame telah selesai. Dengan pemanggilan ini, konten frame telah di gambar dan jika koreksi distorsi diaktifkan, koreksi distrosi akan diterapkan. Setiap perendereran pada tahap ini relatif terhadap seluruh permukana, tidak terhadap satu pandangan mata tunggal.
  - **public abstract void onRendererShutdown ()**
    Method ini dipanggil ketika thread perender sedang menutup. Melepaskan sumber GL(Graphics Library) dan sedang melakukan penutupan operasi pada thread perender. Dipanggil hanya jika sebelumnya ada pemanggilan method onSurfaceCreated.

  - **public abstract void onSurfaceChanged (int width, int height)**
    Dipanggil ketika ada perubahan dimensi permukaan. Semua nilai adalah relatif ke ukuran yang dibutuhkan untuk merender sebuah mata.
  - **public abstract void onSurfaceCreated (EGLConfig config)**
    Method ini dipanggil ketika suatu permukaan dibangun atau dibangun ulang.

- HeadTransform
  Method ini mendeskripsikan transformasi kepala secara independen dari setiap parameter mata. Kelas ini digunakan di kelas GvrView.StereoRenderer sebagai parameter pada method **onNewFrame**. Method-method yang perlu diperhatikan pada kelas ini adalah:
  - **public void getHeadView (float[] headView, int offset)**
    Method ini digunakan untuk mendapatkan matriks transformasi dari camera virtual ke kepala. Kepala disini didefinisikan sebagai titik tengah diantara kedua mata. Matriks yang didapatkan akan disimpan pada parameter **headView**.
  - **public void getQuaternion (float[] quaternion, int offset)**
    Method ini digunakan untuk mendapatkan quaternion yang merepresentasikan rotasi kepala.

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

2.3 Head Motion Detection Menggunakan Camera Inframerah

[9]Mengangguk kepala dan menggeleng kepala merupakan gerakan non-verbal yang sering digunakan untuk berkomunikasi. Arah dari pergerakan kepala dapat diperoleh dari posisi pupil pada camera. Posisi pupil pada camera ini digunakan pada observasi diskrit Hidden Markov Model(HMM) berdasarkan pola alat analisa untuk mendeteksi kepala ketika sedang mengangguk atau menggeleng. Sistem ini akan dilatih dan diuji pada data rekaman dari sepuluh pengguna dengan variasi pencahayaan dan ekspresi muka. Sistem ini memiliki tingkat akurasi sebesar 78.46% berdasarkan pada data tes.

2.3.1 Probabilitas Kemungkinan dan Kebebasan

Terkadang kita memiliki pengetahuan tentang hasil dari suatu eksperimen dan secara alamiah mungkin mempengaruhi hasil pencobaan lainnya. Pengetahuan ini dapat di peroled dengan konsep probabilitas bersyarat. Kemungkinan dari suatu kejadian sebelum mempertimbahkan pengetahuan tambahan disebut juga *prior probability* dari suatu kejadian. Kemungkinan dari hasil penggunaan pengetahuan tambahan disebut dengan *posterior probability* dari suatu kejadian. Probabilitas bersyarat dari suatu kejadian A yang terjadi setelah kejadian B terjadi($P(A|B)$) dengan Peluang B

lebih besar dari 0 ($P(B) > 0$) adalah:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$atau,$$

$$P(A|B) = \frac{P(A, B)}{P(B)}$$

bahkan jika P(B) = 0, masih dapat diselesaikan dengan aturan perkalian:

$$P(A \cap B) = P(B)P(A|B) = P(A)P(B|A)$$

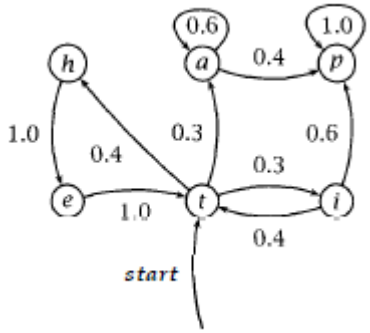
Generalisasi dari aturan ini untuk banyak kejadian adalah sebagai berikut, atau disebut juga sebagai *chain rule*:

$$P(A_1 \cap ... \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2)...P(A_n| \cap_{i=1}^{n-1} A_i)$$

Chain rule ini akan digunakan dalam Markov Model. Kejadian A dengan B bersifat independen. Hal ini menyebabkan jika $P(A \cap B) = P(A)P(B)$ kecuali $P(B) = 0$ akan menghasilkan persamaan $P(A) = P(A|B)$

2.3.2 Markov Model

[10] Markov model dapat digunakan ketika ingin memodelkan probabilitas dari suatu urutan linear



Gambar 2.6: Contoh Markov Model

pada suatu peristiwa. Contohnya, markov model yang digunakan dalam pemodelan urutan tutur perkataan dalam suatu dialog. Selain itu markov model juga digunakan untuk pengenal suara dan lain-lain. Deret Markov menggunakan Deret Markov dapat merepresentasikan oleh Markov model dengan suatu State Diagram seperti pada gambar 2.6. Transisi yang mungkin terjadi digambarkan dengan panah yang menghubungkan antar kondisi, dan pada lengkungan panahnya di berikan label probabilitas dari transisi tersebut. Jumlah dari seluruh probabilitas yang keluar dari suatu kondisi akan berjumlah 1. Dalam Markov Model, dapat diketahui bahwa kondisi-kondisi yang akan suatu mesin tempuh, sehingga urutan kondisi atau beberapa fungsi deterministik dapat dijadikan sebagai pengeluaran. Probabilitas dari suatu urutan kondisi (yaitu urutan dari variabel-variabel random) $X_i, ..., X_T$ dapat dengan mudah di kalkukasi dengan deret Markov. Deret Markov dapat

Kumpulan kondisi	$S = \{s_1, ..., s_N\}$
Keluaran alphabet	$\{k_1, ..., k_M\} = \{1, ..., M\}$
Probabilitas kondisi awal	$II = \{\pi_i\}, i \in S$
Probabilitas kondisi transisi	$\{a_{ij}\}, i, j \in S$
Probabilitas emisi simbol	$\{a_{ij}\}, i, j \in S$
Deretan kondisi	$x = (X,..., X_{T+1})X_t : s \to \{1, ..., M\}$
Deretan keluaran	$O = (o_1, ..., o_T)O_t \in K$

Tabel 2.2: Notasi yang digunakan dalam HMM

direpresentasikan dengan persamaan berikut:

$$\begin{aligned} P(X_1, ..., X_T) =& P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)...P(X_T|X_1, ..., X_{T-1}) \\ =& P(X_1)P(X_2|X_1)P(X_3|X_2)...P(X_T|X_1, ..., X_{T-1}) \\ =& \pi_{X_1}\Pi_{t=1}^{T-1} AX_tX_{t+1} \end{aligned}$$

Jadi, dengan Markov Model pada gambar 2.6, didapatkan:

$$\begin{aligned} P(t, i, p) =& P(X_1 = t)P(X_2 = i|X_1 = t)P(X_3 = p|X_2 = i) \\ =& 1.0 \times *0.3 \times *0.6 \\ =& 0.18 \end{aligned}$$

Hidden Markov Models

Pada HMM(Hidden Markov Models), tidak dapat diketahui urutan kondisi yand dilewati model, tetapi hanya beberapa fungsi probabilistik saja yang diketahui. HMM berguna ketika dapat memikirkan peristiwa secara probabilistik yang menghasilkan peristiwa muka.

Bentuk Umum HMM

HMM dispesifikasikan dengan (S, K, II, A, B), dimana S dengan K adalah kumpulan kondisi dan keluaran alphabet, dan II, A, dan B adalah probabilitas untuk kondisi awal, kondisi transisi, dan emisi simbul, secara berturut-turut. Notasi-notasi tersebut akan dijelaskan lebih lanjut pada tabel 2.2

Mencari Probabilitas dari Suatu Observasi Diberikan suatu urutan observasi $O = (o_1, ..., o_T)$ dan suatu model $\mu = (A, B, II)$, yang ingin dicari adalah bagaimana mengkalukasi $P(O|\mu)$ secara efisien. Proses ini sering disebut juga sebagai *decoding*. Untuk semua deret kondisi $X = (X_1, ..., X_{T+1})$,

$$\begin{aligned} P(O|X, \mu) =& \Pi_{t=1}^{T} P(o_t|X_t, X_t + 1, \mu) \\ =& b_{X_1X_2o_1}b_{X_2X_3o_2}...b_{X_TX_{T+1}o_T} \end{aligned}$$

dan,

$$P(X|\mu) = \Pi_{X_1}a_{X_1X_2}a_{X_2X_3}...a_{X_TX_{T+1}}$$

adapun,

$$P(O, X|\mu) = P(O|X, \mu)P(X|\mu)$$

olehkarena itu,

$$\begin{aligned} P(O|\mu) =& \Sigma_X P(O|X, \mu)P(X|\mu) \\ =& \Sigma_{X_1 \ldots X_{T+1}} \pi X_1 \Pi_{t=1}^{T} a_{X_t X_{T+1}} b_{X_t X_{t+1} o_T} \end{aligned}$$

2.4 Teori Quaternion

Pada Android SDK **SensorEvent.values** [11] tipe sensor **Sensor.TYPE_ROTATION_VECTOR**, yaitu tipe sensor yang mendeteksi vektor perputaran pada *smartphone*. Tipe sensor ini dijelaskan akan mengembalikan nilai-nilai dari komponen quaternion. Quaternion[12] adalah objek penggabungan dari suatu skalar dengan suatu vektor, sesuatu yang tidak dapat didefinisikan dalam aljabar linear biasa.

2.4.1 Struktur Ajabar

Struktur-Struktur aljabar yang digunakan adalah Bilangan Kompleks, Konjugasi Kompleks, *Quaternion Algebra*, dan Operasi-operasi pada *Quaternion*.

Bilangan Kompleks

Dalam matematika, bilangan kompleks adalah bilangan yang berbentuk

$$a + bi$$

[12]
dan a dengan b merupakan bilangan riil, dan i merupakan bilangan imajiner tertentu yang memiliki sifat $i^2 = -1$.

Berikut notasi-notasi bilangan kompleks :

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

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

$$a(c + id) = ac + iad$$

Bilangan kompleks dapat digunakan untuk rotasi dua dimensi. Dengan $a = \cos(\frac{\theta}{2})$, dan $b = \sin(\frac{\theta}{2})$, kemudian akan dikalikan dengan vektor yang ingin di putar, dengan sumbu putar adalah titik pusat.

Konjugasi Kompleks

[12]Berhubungan dengan bentuk umum bilangan kompleks,

$$z = a + ib$$

satu bilangan dikatakan konjugasi kompleks jika

$$\bar{z} = a - ib$$

Dari kedua persamaan diatas dapat dihasilkan :

$$z + \bar{z} = 2a$$

dan,

$$z\bar{(z)} = a^2 + b^2 = |z|^2$$

2.4.2 *Quaternion Algebra* dan Operasi-operasi pada Quaternion

[12]Pada buku "Quaternions and Rotation Sequences : A Primer with Applications to Orbits, Aerospace, and Virtual Reality" disebutkan :

> In 1843 Hamilton invented the so-called hyper-complex number of rank 4, to which he gave the name *quaternion*. Crucial to this invention was his celebrated rule
>
> $$i^2 = j^2 = k^2 = ijk = -1$$
>
> for dealing with the operations on the vector part of the quaternion .

[12]
Dari kutipan diatas bilangan *hyper-complex* tingkat empat yang dapat disebut juga *quaternion* memiliki satu aturan yang dicetuskan oleh Hamilton yaitu:

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

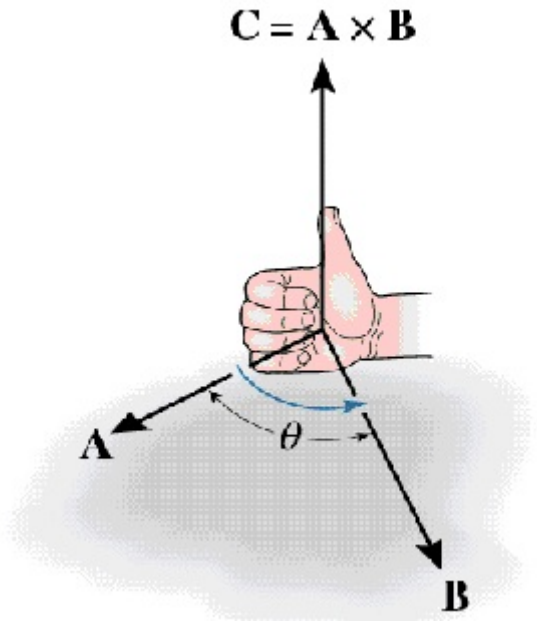
Untuk hasil dari perkalian dua *quaternion* memiliki aturan yang lebih rumit, sehingga memiliki aturan-aturan khusus. Berikut aturan-aturan khususnya :

$$\begin{aligned} ij &= k = -ji \\ jk &= i = -kj \\ ki &= j = -ik \end{aligned} \tag{2.1}$$

Perhatikan bahwa ketiga persamaan diatas mirip dengan cross product vektor sesuai dengan aturan tangan kanan (*right-hand rule*). Dapat di katakan pada Gambar 2.7, A berperan sebagai i, B berperan sebagai j, dan C berperan sebagai k. Sehingga terpenuhi sesuai dengan persaman-persamaan 2.1

Quaternion memiliki persamaan umum yang memiliki empat bilangan riil atau skalar. Persamaan tersebut adalah

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



Gambar 2.7: Right-hand rule dalam *cross product* vektor

[12] Sama seperti pada bilangan kompleks, *quaternion* juga memiliki konjugasi kompleksnya. Berikut adalah konjugasi kompleks *quaternion*

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

DAFTAR REFERENSI

[1] T. Parisi, *Learning virtual reality: developing immersive experiences and applications for desktop, web, and mobile.* O'Reilly Media, 1 ed., 2015.

[2] G. J. Kim, *Designing virtual reality systems: the structured approach.* Springer, 2005.

[3] J. Vince, *Introduction to virtual reality.* Springer, 2004.

[4] "Google cardboard." https://vr.google.com/cardboard/. [Online; diakses 10-September-2016].

[5] "Sensor types." https://source.android.com/devices/sensors/sensor-types.html. [Online; diakses 10-September-2016].

[6] G. Bleser and D. Stricker, "Advanced tracking through efficient image processing and visual–inertial sensor fusion," *Computers & Graphics*, vol. 33, no. 1, pp. 59–72, 2009.

[7] A. Developers, "What is android," 2011.

[8] "Google vr | google developers." https://developers.google.com/vr/. [Online; diakses 20-September-2016].

[9] A. Kapoor and R. W. Picard, "A real-time head nod and shake detector," in *Proceedings of the 2001 workshop on Perceptive user interfaces*, pp. 1–5, ACM, 2001.

[10] H. S. Christopher D. Manning, *Foundations of Statistical Natural Language Processing.* The MIT Press, 1 ed., 1999.

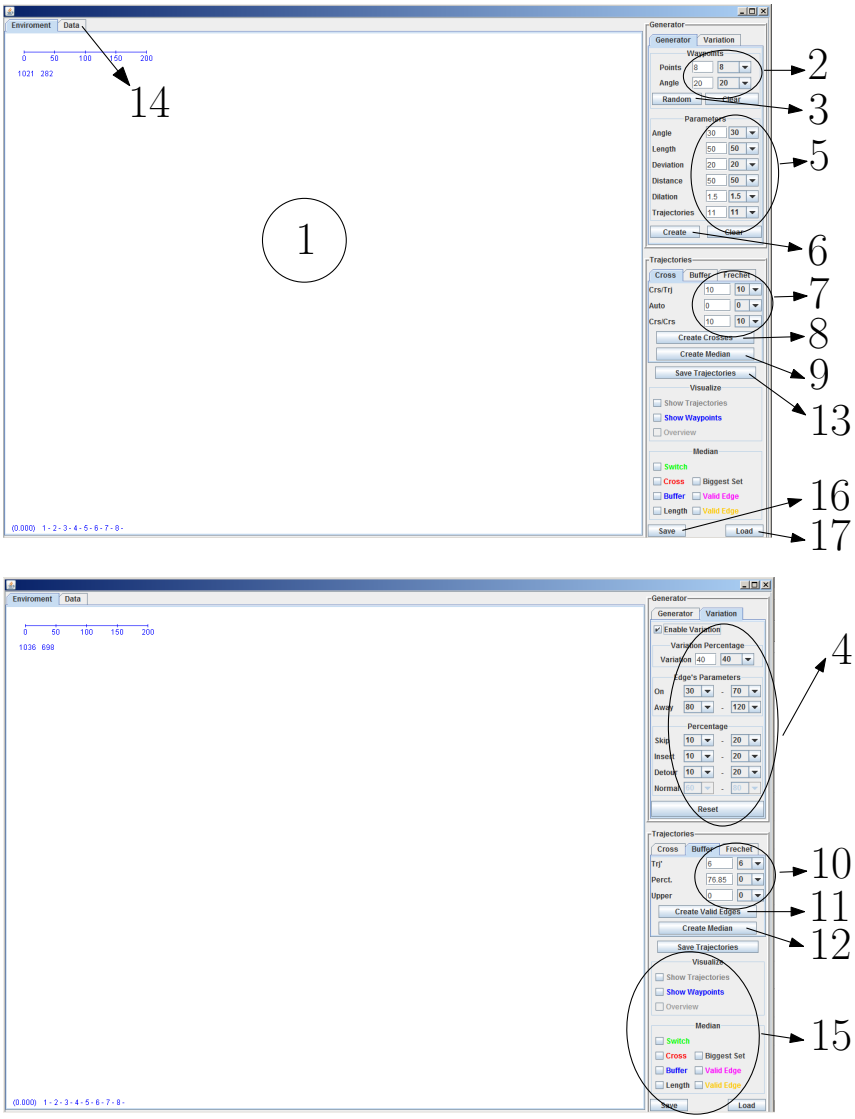
[11] "Android 7.0 nougat!." https://developer.android.com/. [Online; diakses 12-September-2016].

[12] J. B. Kuipers, *Quaternions and Rotation Sequences : A Primer with Applications to Orbits, Aerospace, and Virtual Reality.* Princeton University Press, 1998.

LAMPIRAN A

THE PROGRAM

The interface of the program is shown in Figure A.1:



Gambar A.1: Interface of the program

Step by step to compute the median trajectory using the program:

1. Create several waypoints. Click anywhere in the "Environment" area(1) or create them automatically by setting the parameters for waypoint(2) or clicking the button "Random"(3).

2. The "Variation" tab could be used to create variations by providing values needed to make them(4).

3. Create a set of trajectories by setting all parameters(5) and clicking the button "Create"(6).

4. Compute the median using the homotopic algorithm:

   - Define all parameters needed for the homotopic algorithm(7).
   - Create crosses by clicking the "Create Crosses" button(8).
   - Compute the median by clicking the "Compute Median" button(9).

5. Compute the median using the switching method and the buffer algorithm:

   - Define all parameters needed for the buffer algorithm(10).
   - Create valid edges by clicking the "Create Valid Edges"button(11).
   - Compute the median by clicking the "Compute Median"button(12).

6. Save the resulting median by clicking the "Save Trajectories" button(13). The result is saved in the computer memory and can be seen in "Data" tab(14)

7. The set of trajectories and its median trajectories will appear in the "Environment" area(1) and the user can change what to display by selecting various choices in "Visualize" and "Median" area(15).

8. To save all data to the disk, click the "Save"(16) button. A file dialog menu will appear.

9. To load data from the disk, click the "Load"(17) button.

LAMPIRAN B

THE SOURCE CODE

Listing B.1: MyFurSet.java

```
1
2  import java.util.ArrayList;
3  import java.util.Collections;
4  import java.util.HashSet;
5
6  /**
7   *
8   * @author Lionov
9   */
10
11 //class for set of vertices close to furthest edge
12 public class MyFurSet {
13     protected int id;                                    //id of the set
14     protected MyEdge FurthestEdge;                       //the furthest edge
15     protected HashSet<MyVertex> set;                     //set of vertices close to furthest edge
16     protected ArrayList<ArrayList<Integer>> ordered;     //list of all vertices in the set for each
           trajectory
17     protected ArrayList<Integer> closeID;                //store the ID of all vertices
18     protected ArrayList<Double> closeDist;               //store the distance of all vertices
19     protected int totaltrj;                              //total trajectories in the set
20
21     /**
22      * Constructor
23      * @param id : id of the set
24      * @param totaltrj : total number of trajectories in the set
25      * @param FurthestEdge : the furthest edge
26      */
27     public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
28         this.id = id;
29         this.totaltrj = totaltrj;
30         this.FurthestEdge = FurthestEdge;
31         set = new HashSet<MyVertex>();
32         ordered = new ArrayList<ArrayList<Integer>>();
33         for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
34         closeID = new ArrayList<Integer>(totaltrj);
35         closeDist = new ArrayList<Double>(totaltrj);
36         for (int i = 0;i <totaltrj;i++) {
37             closeID.add(-1);
38             closeDist.add(Double.MAX_VALUE);
39         }
40     }
41
42     /**
43      * set a vertex into the set
44      * @param v : vertex to be added to the set
45      */
46     public void add(MyVertex v) {
47         set.add(v);
48     }
49
50     /**
51      * check whether vertex v is a member of the set
52      * @param v : vertex to be checked
53      * @return true if v is a member of the set, false otherwise
54      */
55     public boolean contains(MyVertex v) {
56         return this.set.contains(v);
57     }
58 }
```