Deteksi Gerakan Kepala dengan Google Cardboard

Ega Prianto–2013730047

1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian**
Pembimbing pendamping: -
Kode Topik : **PAS4102**
Topik ini sudah dikerjakan selama : **1** semester
Pengambilan pertama kali topik ini pada : Semester **41 - Ganjil 16/17**
Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**
Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencan kerja/laporan perkembangan terkahir :

1. Melakukan studi literatur tentang Android SDK, Google VR SDK, Quaternion.
   status : Ada sejak rencana kerja skripsi. Algoritma *Head Motion Detection* tidak ditemukan literaturnya, algoritma dibuat pada bagian analisis metode pendeteksi gerakan kepala
   hasil :

   - **Android SDK**

   Dalam membuat aplikasi pada perangkat Android, dibutuhkan Android SDK. Android SDK(*software development kit*) adalah kumpulan *source code, development tools, emulator,*[1] 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[1], sekaligus dengan Android SDKnya.

   Struktur File Android Studio Project

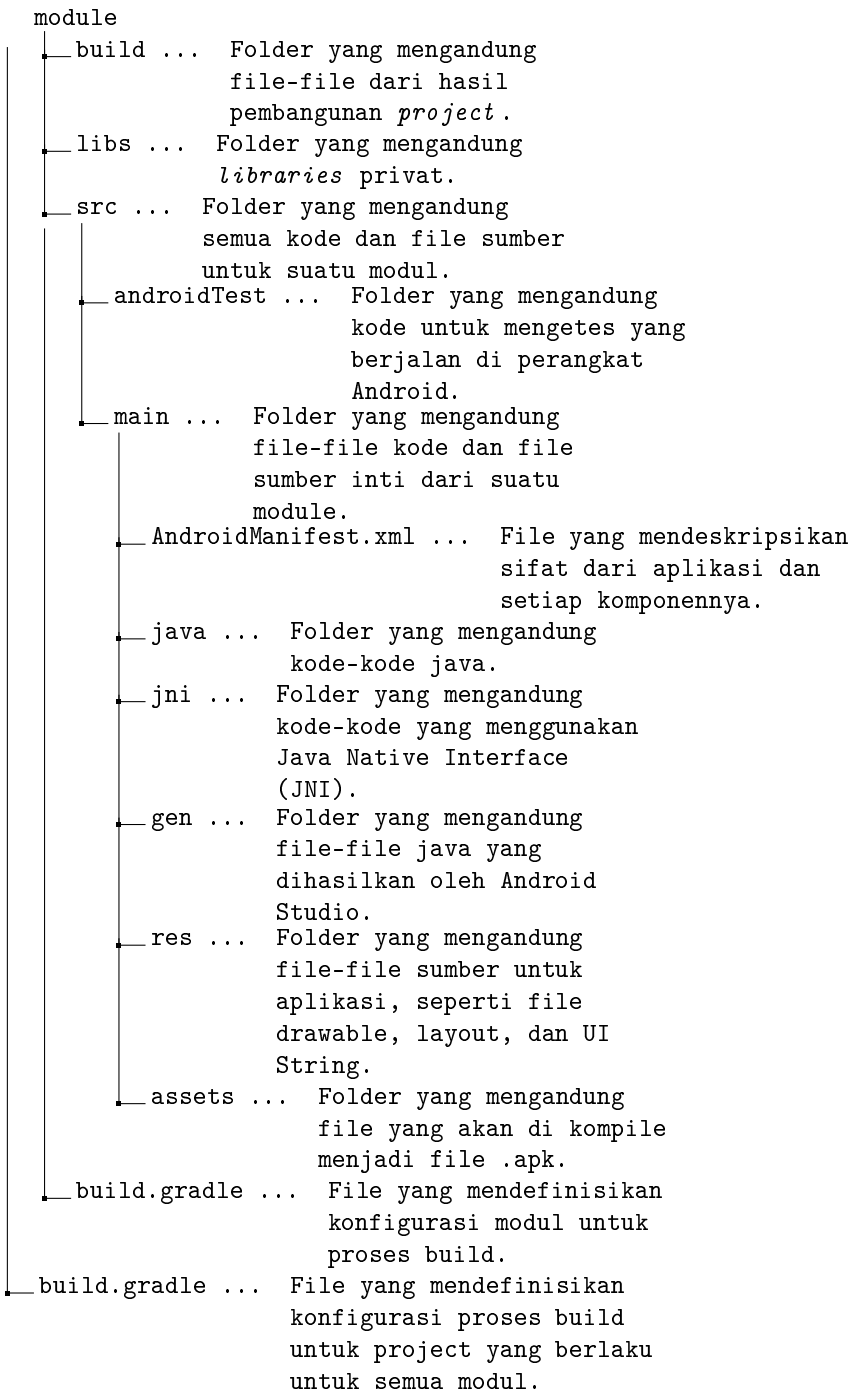
   Pada saat project baru telah dibuat, Android Studio akan membuatkan folder-folder standar seperti pada Gambar 1. Gambar 2 merupakan *screenshot* dari project pada IDE Android Studio. Folder **App** pada Gambar 2 merupakan folder *module*.

   Membuat Interface Android

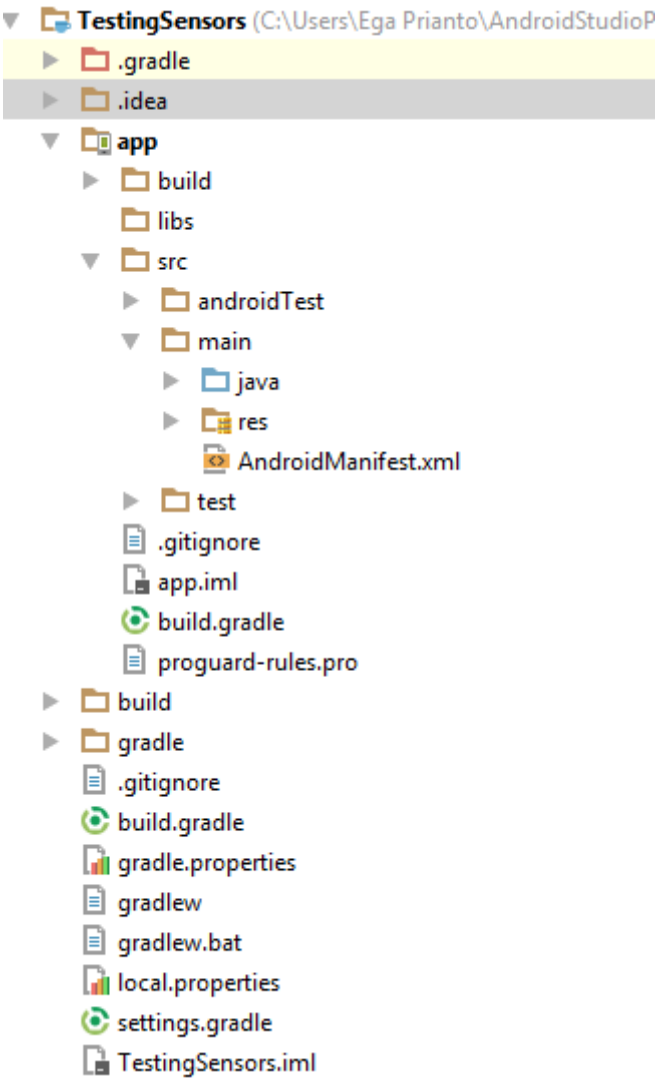
   [2] Pada bagian ini akan dijelaskan bagaimana membuat layout di XML termasuk *text field* dan *button*.

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

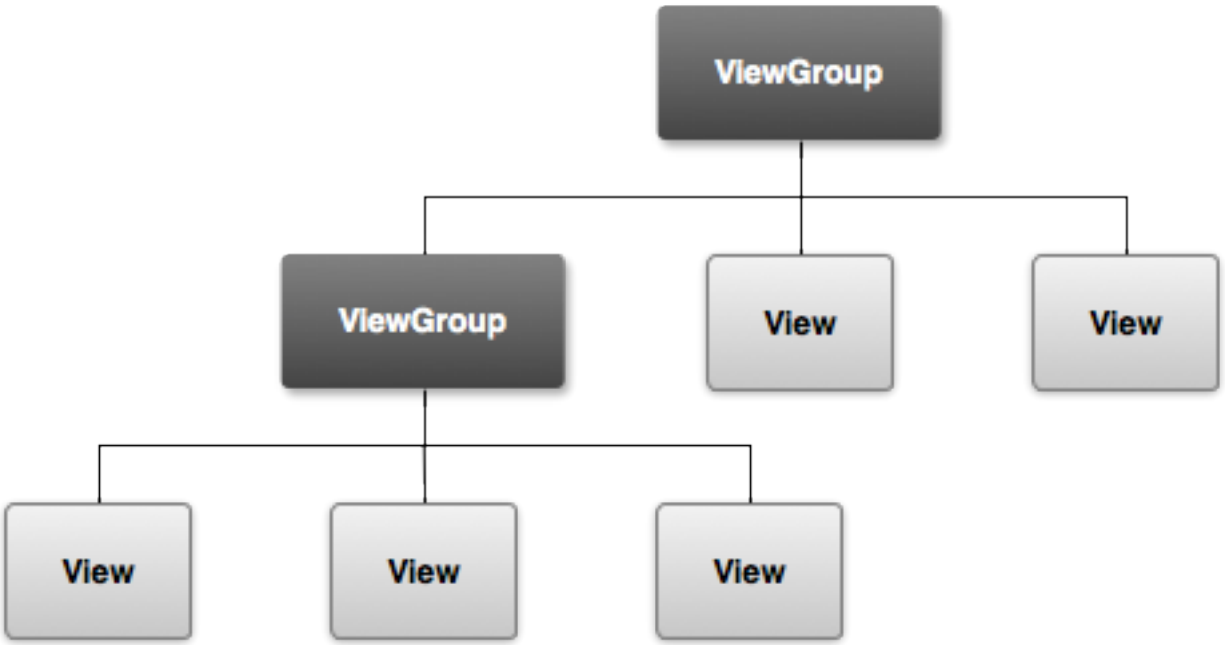
   GUI untuk aplikasi Android dibuat dengan hierarki dari objek View dan ViewGroup (Gambar 3). Objek-objek dari View biasanya adalah *UI(User Interface) Widgets* seperti *button* atau *text field*. Objek-objek dari ViewGroup tidak terlihat oleh *view containers* yang mendefinisikan bagaimana *child views* ditata seperti *grid* atau *vertical list*.



Gambar 1: Tampilan struktur folder pada *project* Android Studio



Gambar 2: Tampilan struktur folder pada *project* Android Studio



Gambar 3: 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 bagian 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 bagian Activity). Tanda '@' dibutuhkan ketika mereferensi object dari suatu XML. Tanda '@' tersebut diikuti dengan tipe (id pada kasus ini), *slash*, dan nama (edit_message pada Listing 2). Tanda tambah (+) sebelum tipe hanya dibutuhkan jika ingin mendefinisikan *resource ID* untuk pertama kalinya.
- **android:layout_width** dan **android:layout_height** Attribut ini digunakan untuk mendefinisikan panjang dan lebar dari suatu objek View. Daripada menggunakan besar spesifik untuk panjang dan lebarnya, lebih baik menggunakan "wrap_content" yang menspesifikasi viewnya hanya akan sebesar yang dibutuhkan untuk memuat konten-konten dari View. Jika menggunakan "match_parent" pada kasus Listing 2 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 Listing 1.

Listing 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 Listing 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: Contoh kode file XML pada folder layout

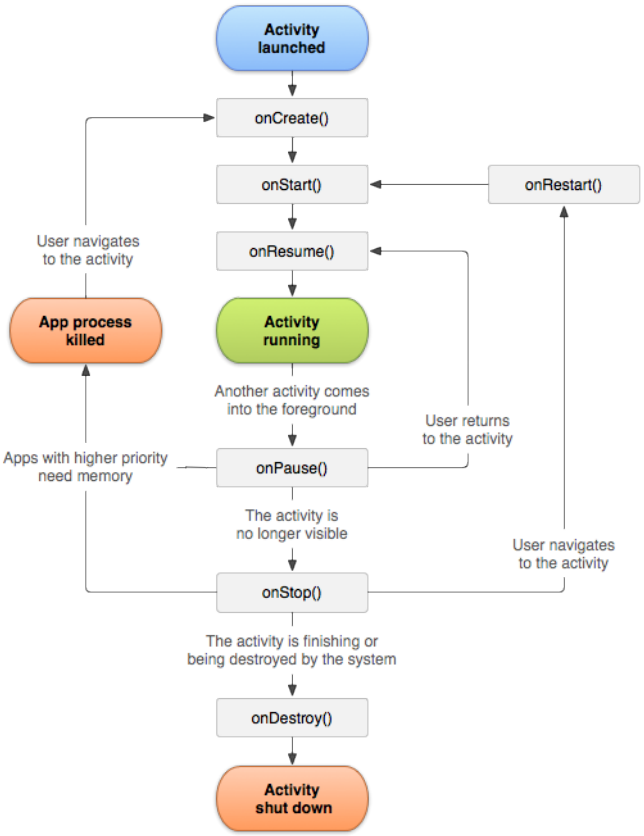
```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <EditText android:id="@+id/edit_message"
                        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>
```

Activity

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

Activity Lifecycle

Aktivitas dalam sistem android di atur sebagai *activity stack*. Ketika ada Activity baru yang dimulai, Activity tersebut ditempatkan di paling atas pada *stack* dan menjadi Activity aktif. Activity sebe-



Gambar 4: *State diagram* siklus Activity

lumnya 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 4 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.

Android Sensor Framework

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

- **Sensor gerak** Sensor-sensor ini mengukur akselerasi dan rotasi pada tiga sumbu. Kategori sensor ini meliputi *accelerometers*, sensor gravitasi, *gyroscope*, dan *rotation vector*.
- **Sensor keadaan lingkungan** Sensor-sensor ini mengukur berbagai keadaan lingkungan seperti suhu udara, tekanan, pencahayaan, dan kelembaban. 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

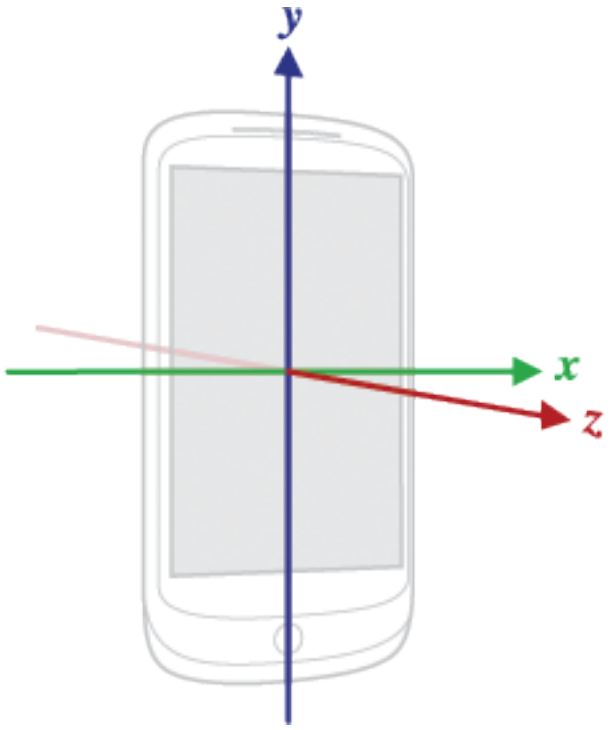
Pada umumnya, sensor framework menggunakan sistem tiga sumbu koordinat standar untuk mengekspresikan nilai data. Sebagian besar sensor sistem koordinat didefinisikan relatif terhadap layar perangkat bila perangkat dibuat dalam orientasi standar (lihat Gambar 5) Sensor-sensor yang menggunakan sistem tiga sumbu seperti Gambar 5 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*

Tabel 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.



Gambar 5: Sistem koordinat (relatif dengan perangkatnya) yang digunakan oleh Sensor API

standar orientasinya adalah *landscape*. Sistem koordinat sensor selalu di dasarkan pada orientasi dasar dari suatu perangkat android.

Mengidentifikasi Sensor dan Kapabilitas Sensor

Android Sensor Framework menyediakan beberapa method yang dapat membuat developer mudah untuk menentukan sensor mana yang akan digunakan. APInya juga dapat menyediakan method yang memungkinkan penggunanya menentukan kapabilitas masing-masing sensor, seperti jangkauan maksimum, resolusi, dan kebutuhan dayanya. Untuk mengidentifikasi sensor-sensor yang ada pada perangkat, hal pertama yang perlu dilakukan adalah mendapatkan referensi sensor tersebut. Untuk mendapatkan referensi tersebut, dapat dilakukan dengan membuat instansiasi dari kelas SensorManager dan memanggul method getSystemService() dan memasukkan isi Parameternya dengan SENSOR_SERVICE. Contohnya pada Listing 3.

Listing 3: Contoh inisialisasi kelas SensorManager

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

Kemudian untuk mendapatkan daftar dari setiap sensor pada suatu perangkat dapat dilakukan dengan cara memanggil method getSensorList() dan menggunakan konstanta TYPE_ALL pada kelas Sensor. Contohnya pada Listing 4

Listing 4: Contoh untuk mendapatkan daftar dari setiap sensor yang ada

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

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

Untuk menentukan jenis tertentu dari sensor yang ada pada perangkat dapat didapatkan dengan method getDefaultSensor() dengan dimasukkan dengan konstanta yang berada pada kelas Sensor. Jika perangkatnya memiliki lebih dari satu sensor dari tipe sensor yang diberikan, salah satu dari sensor tersebut akan dianggap sebagai sensor dasar. Jika sensor dasanya tidak ada untuk sensor tersebut, perangkat tersebut berarti tidak memiliki sensor dengan jenis yang diberikan. Listing 5 adalah contoh untuk mengecek apakah perangkat yang digunakan memiliki sensor dengan jenis yang diberikan.

Listing 5: Contoh untuk mengecek apakah perangkat yang digunakan memiliki sensor dengan jenis yang diberikan

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

Jika ingin membatasi versi atau vendor dari sensor yang akan digunakan, dapat menggunakan method getVendor() dan getVersion(). Seperti pada Listing 6 yang mengharuskan sensor gravitasi bervendor "Google Inc." dan memiliki versi 3. Jika sensor tersebut tidak tersedia pada perangkat, sensor accelerometerlah yang digunakan.

Listing 6: Contoh untuk mengecek apakah perangkat yang digunakan memiliki sensor dengan jenis yang diberikan

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

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

Memonitor Nilai Sensor

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

- **Perubahan akurasi sensor.**
  Dalam kasus ini sistem memanggil method onAccuracyChanged(Sensor sensor, int accuracy).

Parameter sensor akan diberikan objek Sensor yang telah berubah akurasinya, dan parameter accuracy adalah nilai akurasi sensor yang baru.

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

Pada Listing 7 akan ditunjukkan bagaimana menggunakan method onSensorChanged(SensorEvent event) untuk memonitor data dari sensor cahaya. Pada Listing 7 akan menampilkan data mentah dari sensor ke TextView yang telah didefinisikan pada file main.xml sebagai sensor_data.

Listing 7: Contoh memonitor data mentah pada sensor cahaya

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

```
32
33    @Override
34    protected void onPause() {
35      super.onPause();
36      mSensorManager.unregisterListener(this);
37    }
38  }
```

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

Struktur Nilai yang Dikembalikan oleh Sensor

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

- **TYPE_ACCELEROMETER**
  Semua nilai didefinisikan sebagai satuan m/s^2
  - values[0]: Percepatan yang terjadi pada sumbu x dikali -1
  - values[1]: Percepatan yang terjadi pada sumbu y dikali -1
  - values[2]: Percepatan yang terjadi pada sumbu z dikali -1

  Sensor ini mengukur percepatan(Ad) yang diterapkan pada perangkat. Sensor tersebut dapat mengukur percepatan dengan mengukur gaya(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 $g = 9.81 m/s^2$. Demikian pula ketika perangkat android sedang dalam keadaan jatuh bebas. Perangkat akan mempercepat menuju ke tanah pada percepatan $9.81 m/s^2$, 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 8: Implementasi *low-pass* filter

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

Low-pass filter dapat diimplementasikan pada Listing 8

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

- **TYPE_GYROSCOPE**
  Sensor ini mengukur rata-rata perputaran pada perangkat yang berputar di sumbu X,Y, dan Z dalam satuan radians/second. Sistem koordinat yang digunakan sama dengan sistem koordinat pada sensor percepatan(Accelerometer). Jika perangkat berputar berlawanan arah jarum jam pada sumbu tertentu, maka rotasi yang terjadi akan bernilai positif. Perhatikan bahwa standar perputaran ini adalah definisi matematika standar pada rotasi positif.
  - values[0]: Percepatan angular pada sumbu X.
  - values[1]: Percepatan angular pada sumbu Y.
  - values[2]: Percepatan angular pada sumbu Z.

  Biasanya keluaran dari gyroscope terintegrasi dari waktu ke waktu untuk menghitung rotasi yang menggambarkan perubahan sudut atas langkah waktu, misalnya pada Listing 9

Listing 9: contoh implementasi gyroscope

```
1  private static final float NS2S = 1.0f / 1000000000.0f;
2  private final float[] deltaRotationVector = new float[4]();
3  private float timestamp;
```

```
4
5  public void onSensorChanged(SensorEvent event) {
6  // Pada tahapan ini delta rotasi akan dikalikan dengan rotasi saat ini
7  // setelah 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                 float thetaOverTwo = omegaMagnitude * dT / 2.0f;
26                 float sinThetaOverTwo = sin(thetaOverTwo);
27                 float cosThetaOverTwo = cos(thetaOverTwo);
28                 deltaRotationVector[0] = sinThetaOverTwo * axisX;
29                 deltaRotationVector[1] = sinThetaOverTwo * axisY;
30                 deltaRotationVector[2] = sinThetaOverTwo * axisZ;
31                 deltaRotationVector[3] = cosThetaOverTwo;
32         }
33 }
34 timestamp = event.timestamp;
35 float[] deltaRotationMatrix = new float[9];
36 SensorManager.getRotationMatrixFromVector(deltaRotationMatrix,
       deltaRotationVector);
37 // User code should concatenate the delta rotation we computed with
       the current rotation
38 // in order to get the updated rotation.
39 // rotationCurrent = rotationCurrent * deltaRotationMatrix;
40 }
```

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:

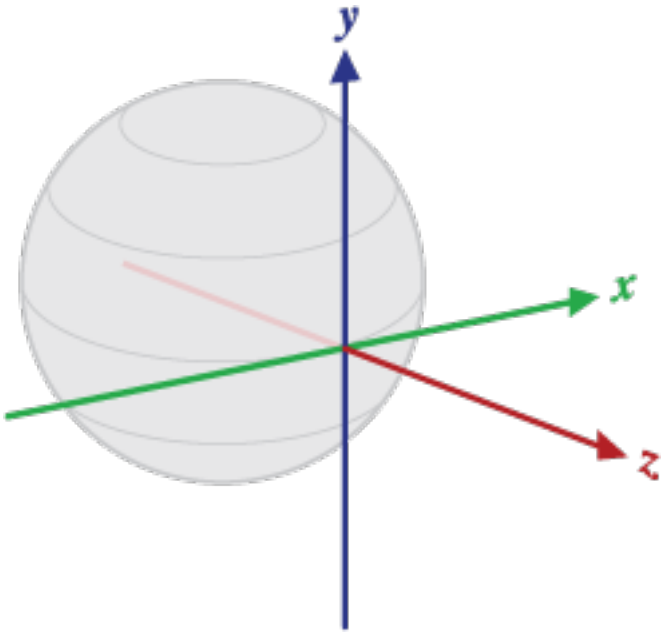
$$percepatan = gravitasi + percepatanlinear$$

- **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 kuaternion$(\cos(\frac{\theta}{2}), x\sin(\frac{\theta}{2}), y\sin(\frac{\theta}{2}), z\sin(\frac{\theta}{2}))$ yang dijelaskan pada bagian Teori Kuaternion. 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 6
  - values[0]: $x\sin(\frac{\theta}{2})$
  - values[1]: $y\sin(\frac{\theta}{2})$
  - values[2]: $z\sin(\frac{\theta}{2})$
  - values[3]: $\cos(\frac{\theta}{2})$



Gambar 6: Sistem koordinat sensor rotasi vektor terhadap Bumi

- values[4]: Perkiraan akurasi (dalam radians) (-1 jika tidak tersedia)

- **Google VR SDK**

Google VR SDK[3] digunakan untuk membantu dalam pembuatan aplikasi Virtual Reality pada *smartphone*. Google VR SDK memberikan beberapa fitur sebagai berikut :

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

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

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

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

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

API Audio [3] 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).

API Base [3] 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 kuaternion yang merepresentasikan rotasi kepala.

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

- **Teori Kuaternion**

Pada Android SDK **SensorEvent.values** [2] tipe sensor **Sensor.TYPE_ROTATION_VECTOR**, yaitu tipe sensor yang mendeteksi vektor perputaran pada *smartphone*. Seperti yang sudah di jelaskan pada bagian sebelumnya, tipe sensor ini akan mengembalikan nilai-nilai dari komponen kuaternion. Kuaternion[4] adalah objek penggabungan dari suatu skalar dengan suatu vektor, sesuatu yang tidak dapat didefinisikan dalam aljabar linear biasa. Kuaternion ditemukan oleh William Rowan Hamilton dengan memperpanjang notasi dari bilangan kompleks menjadi Kuaternion.

Struktur Ajabar

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

Bilangan Kompleks

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

$$a + bi \tag{1}$$

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

- Penjumlahan

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

- Perkalian

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

- Pengurangan

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

- Pembagian

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

Pada operasi penjumlahan dan perkalian untuk kedua bilangan kompleks tersebut memiliki hukum assosiatif dan komutatif. Notasi 2 menunjukkan bagaimana kedua hukum tersebut berlaku pada pen-

jumlahan.

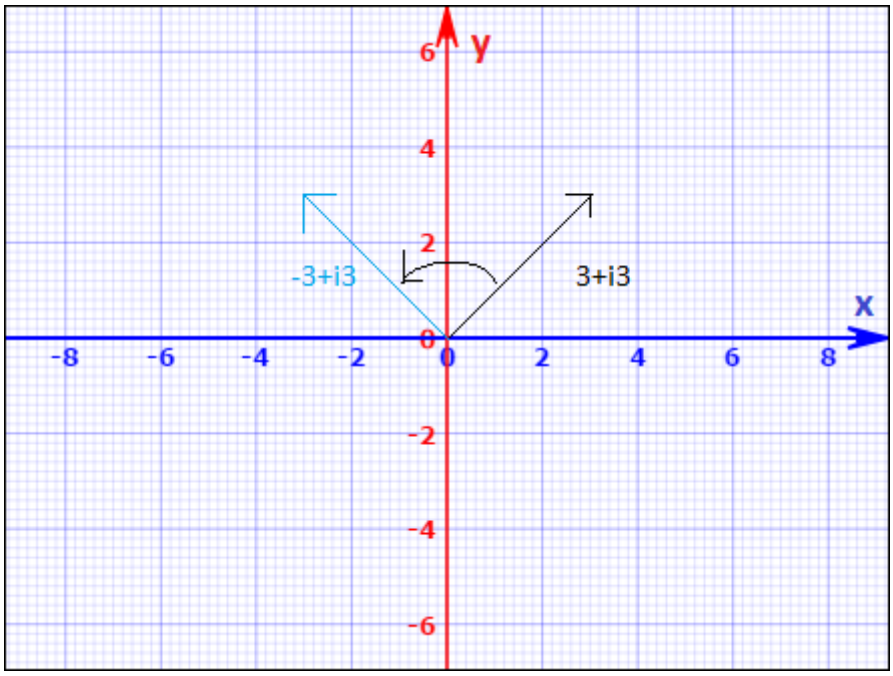
$$\begin{aligned}(a+ib)+(c+id)=(a+c)+i(b+d)=\\(c+id)+(a+ib)=(c+a)+i(d+b)\end{aligned} \tag{2}$$

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

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

$$\begin{aligned}v' =&i(3+i3)\\=&i3+i^2 3\\=&i3+(-1)3\\=&-3+i3\end{aligned} \tag{3}$$

Dengan bilangan pada bilangan riil diasumsikan sebagai nilai pada sumbu x dan bilangan yang di-



Gambar 7: Contoh perputaran dengan bilangan kompleks

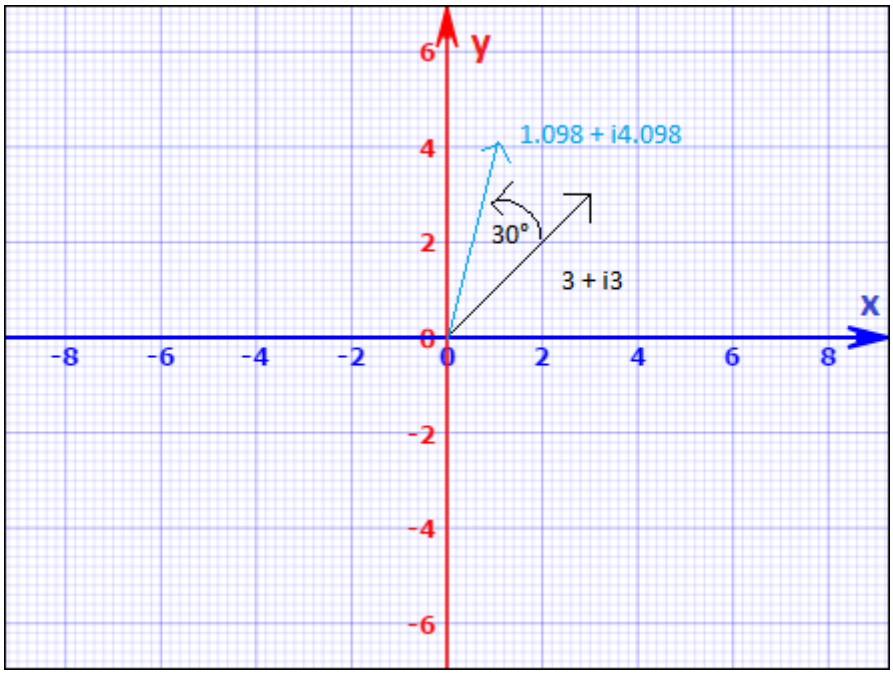
kalikan dengan bilangan i diasumsikan pada sumbu y($x + iy$) Seperti yang ditunjukkan pada Gambar 7. Oleh karena itu rumus perputaran menggunakan bilangan kompleks dapat di rumuskan sebagai berikut:

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

dengan θ adalah besar sudut perputaran. Jika vektor $v = 3 + i3$ diputar 30° berlawanan arah jarum jam menggunakan konsep diatas, akan menghasilkan vektor berikut:

$$\begin{aligned} v' =& (3+i3)(\cos 30^\circ + i \sin 30^\circ) \\ =& (3+i3)(\frac{\sqrt{3}}{2} + i\frac{1}{2}) \\ =& \frac{3\sqrt{3}-3}{2} + i\frac{3\sqrt{3}+3}{2} \\ =& 1.098 + i4.098 \end{aligned} \tag{4}$$

Persamaan 4 dapat divisualisasikan pada Gambar 8.



Gambar 8: Contoh perputaran tiga puluh derajat dengan bilangan kompleks

2. Merancang dan Membuat aplikasi untuk menampilkan grafik sensor-sensor pada smartphone Android.
   status : Ada sejak rencana kerja skripsi.
   hasil :

   Analisis grafik data sensor-sensor pada Android dilakukan dengan membuat suatu aplikasi perekam sensor-sensor yang ada pada *smartphone* Android terlebih dahulu. Aplikasi ini akan merekam nilai-nilai yang dihasilkan dari sebagian sensor-sensor pada android setiap ada perubahan. Pada skripsi ini, nilai sensor-sensor yang dibutuhkan adalah sensor *accelerometer*, *gyroscope*, *rotation vector*, dan *geomagnetic rotation*. Aplikasi menyimpan nilai sensor-sensor menggunakan format CSV *(Comma Separated Values)*. Format CSV adalah format penulisan suatu file dengan menggunakan koma (,) sebagai pemisah antara nilai-nilai pada file. Dari data yang diperoleh oleh apliasi tersebut dibuatkan grafiknya menggunakan aplikasi Microsoft Excel. Penjelasan dari setiap grafik akan dijelaskan pada bagian analisi grafik dari sensor-sensor pada *smartphone* ketika mengangguk dan menggeleng.

3. Menganalisis aplikasi-aplikasi sejenis.
   status : Ada sejak rencana kerja skripsi.
   hasil :

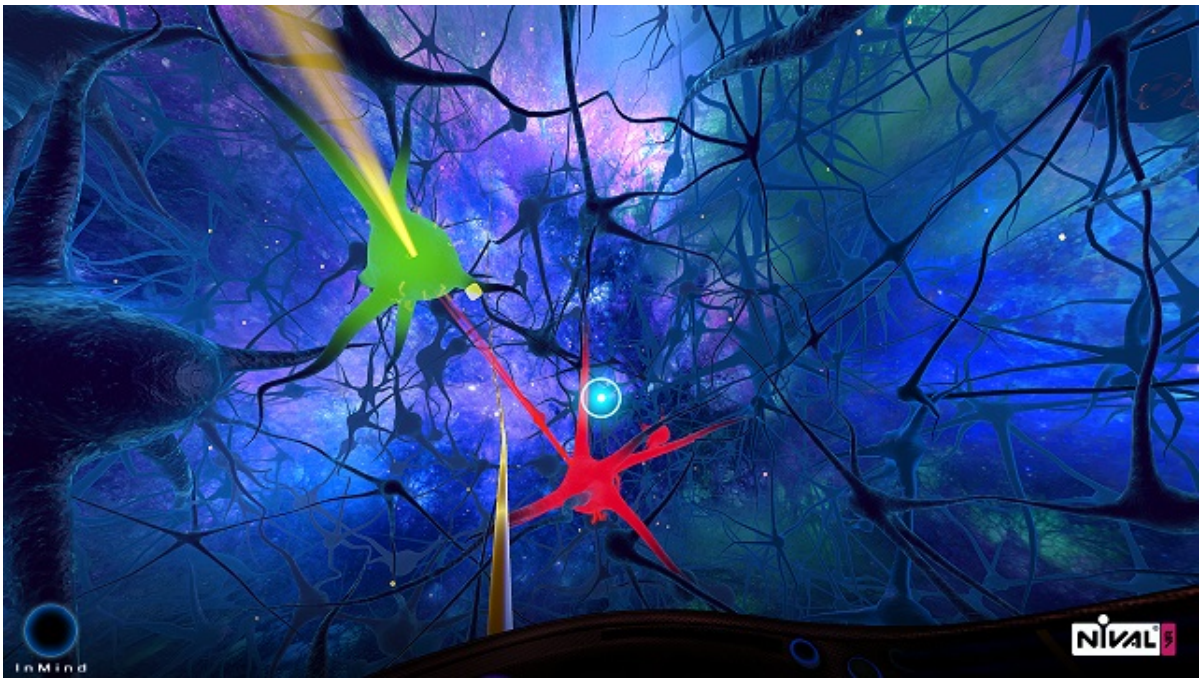
   Aplikasi sejenis pada perangkat *Virtual Reality* Google Cardboard hanyalah ada satu aplikasi saja. Aplikasi tersebut adalah aplikasi InMind VR. Aplikasi sejenis pada perangkat *Virtual Reality* Oculust

Rift adalah Trial of the Rift Drifter dengan Asunder yang di kembangkan oleh AldinDynamics [5]. Aplikasi sejenis pada *Virtual Reality* Oculust Rift tidak dapat dianalisis karena penulis tidak memiliki perangkat Oculust Rift. **Analisis InMind VR**

InMind VR merupakan permainan *Virtual Reality* yang seakan-akan pengguna berada dalam sel-sel otak seorang manusia. Permainan ini dimainkan dengan mengarahkan arah pandang kepala ke arah sel-sel neuron yang dapat menyebabkan gangguan mental (Gambar 9). Akan ada 50 sel neuron yang dapat menyebabkan gangguan mental. Sel-sel yang dapat menyebabkan gangguan mental adalah sel-sel yang berwarna merah seperti pada Gambar 10.

Gambar 9: *Screenshot* task yang diberikan aplikasi InMind VR.

Pada permulaan permainan pengguna diberi pertanyaan apakah pengguna sudah siap untuk melakukan permainan tersebut seperti yang ditunjukkan pada Gambar 11.



Gambar 10: *Screenshot* aplikasi InMind VR ketika permainan berlangsung.

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

(a) Mengangguk secara pelan.

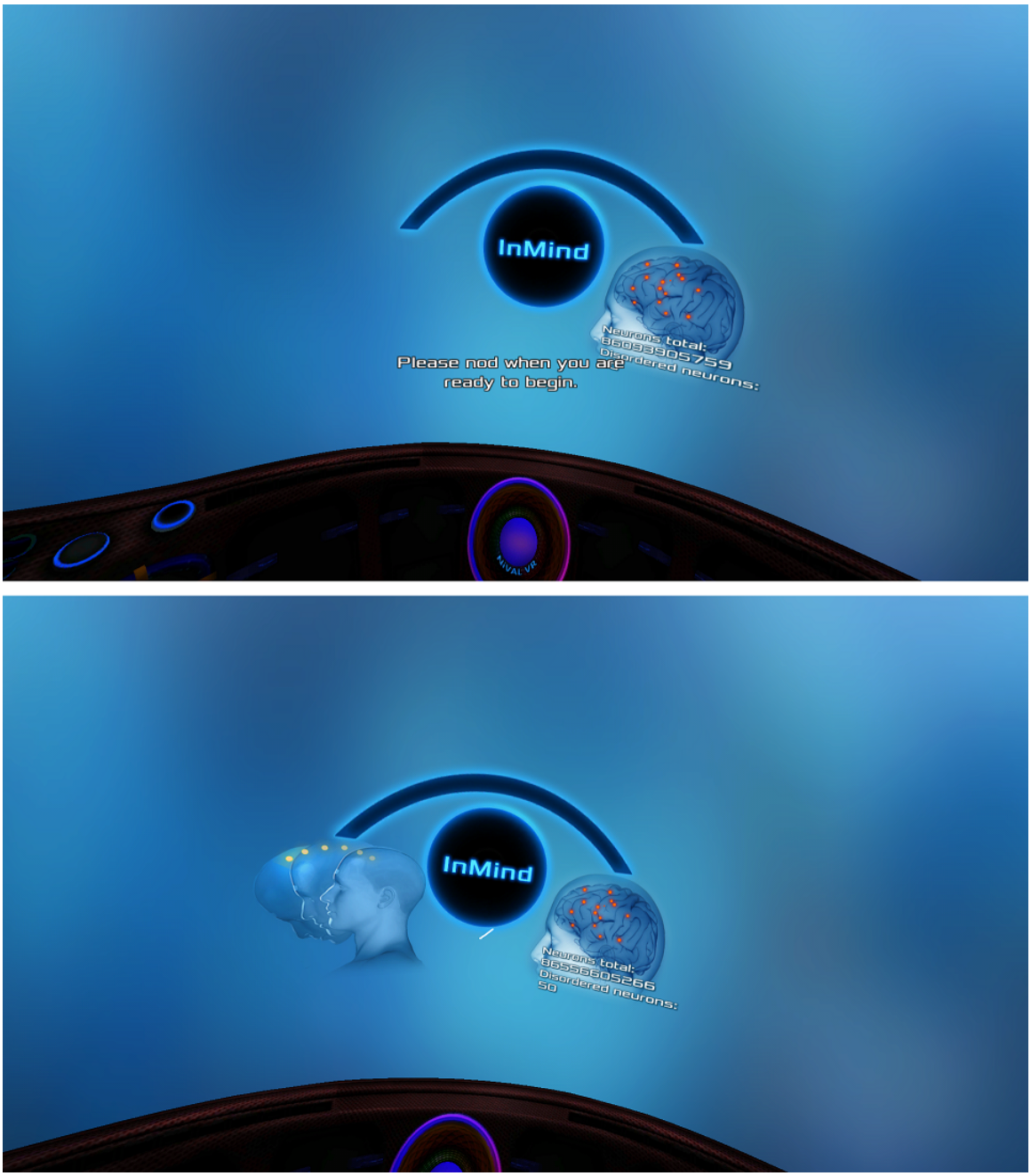
(b) Mengangguk yang diawali dengan gerakan ke atas.

(c) Melihat kebawah saja tanpa membalikkan kepala ke posisi semula.

(d) Tidak menggerakkan kepala sama sekali.

Dari keempat percobaan mengangguk yang dilakukan, hanya percobaan pertama dengan percobaan ketiga yang terdeteksi mengangguk. Aplikasi ini dapat disimpulkan dengan hanya menggerakkan kepala kebawah saja sudah dapat dianggap mengangguk oleh aplikasi ini. Pada percobaan kedua dan keempat terjadi keganjilan dari pendeteksian anggukkan. Pada percobaan kedua dan keempat aplikasi tetap akan melanjutkan permainan setelah beberapa detik berlalu. Sel-sel yang dapat menyebabkan gangguan mental adalah sel-sel yang berwarna merah seperti pada Gambar 10.

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



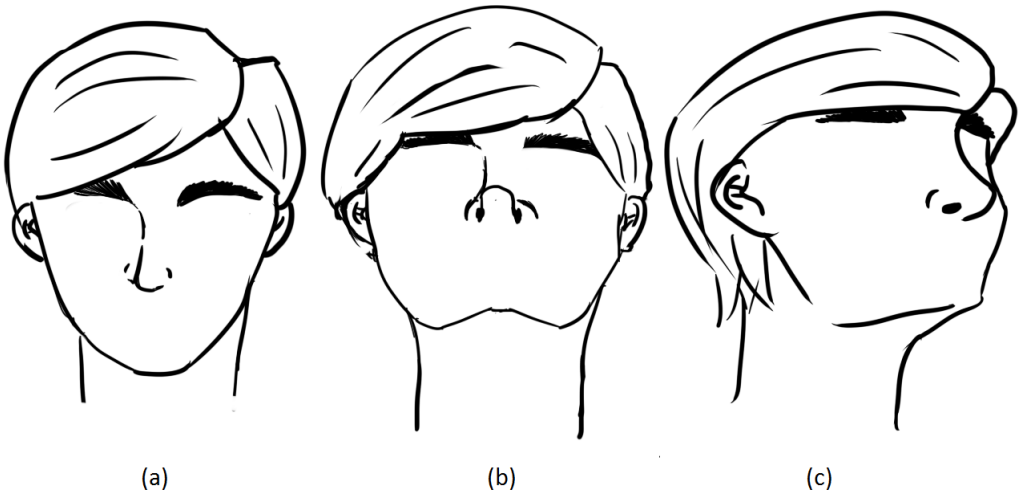
Gambar 11: *Screenshot* aplikasi InMind VR ketika meminta pengguna untuk mengangguk jika telah siap.

4. Merekam dan menganalisis grafik dari sensor-sensor pada *smartphone* ketika mengangguk dan menggeleng.
   status : Ada sejak rencana kerja skripsi.
   hasil :

   - **Perekaman Data Sensor**

   Pada analisis ini akan dilakukan perekaman mengganguk dan menggeleng dengan sensor-sensor pada Android. Perekaman-perekaman ini akan dilakukan pada tiga kondisi muka pengguna. Kondisi muka yang pertama adalah kondisi muka pengguna ketika menghadap ke depan, digambarkan dengan Gambar 12 bagian (a). Kondisi muka yang kedua adalah kondisi muka pengguna keita menghadap ke atas sekitar 45° dari pandangan muka menghadap ke atas digambarkan dengan Gambar 12 bagian (b). Kondisi muka yang ketiga adalah kondisi muka ketika menghadap serong ke kiri atas digambarkan

dengan Gambar 12 bagian (c). Anggukan yang dilakukan oleh pengguna hanya sebanyak satu kali mengangguk ke bawah saja. Sedangkan dalam menggeleng akan bergerak ke kiri terlebih dahulu dan ke kanan setelahnya dan diakhiri pada posisi muka kembali ke posisi awal.



Gambar 12: Gambar untuk mendeskripsikan posisi dari muka sebelum melakukan gerakan menggeleng atau mengangguk. Gambar (a) adalah kondisi muka ketika sedang menghadap ke depan. Gambar (b) adalah kondisi muka ketika sedang menghadap ke atas. Gambar(c) adalah kondisi muka ketika sedang menghadap ke serong kiri atas.

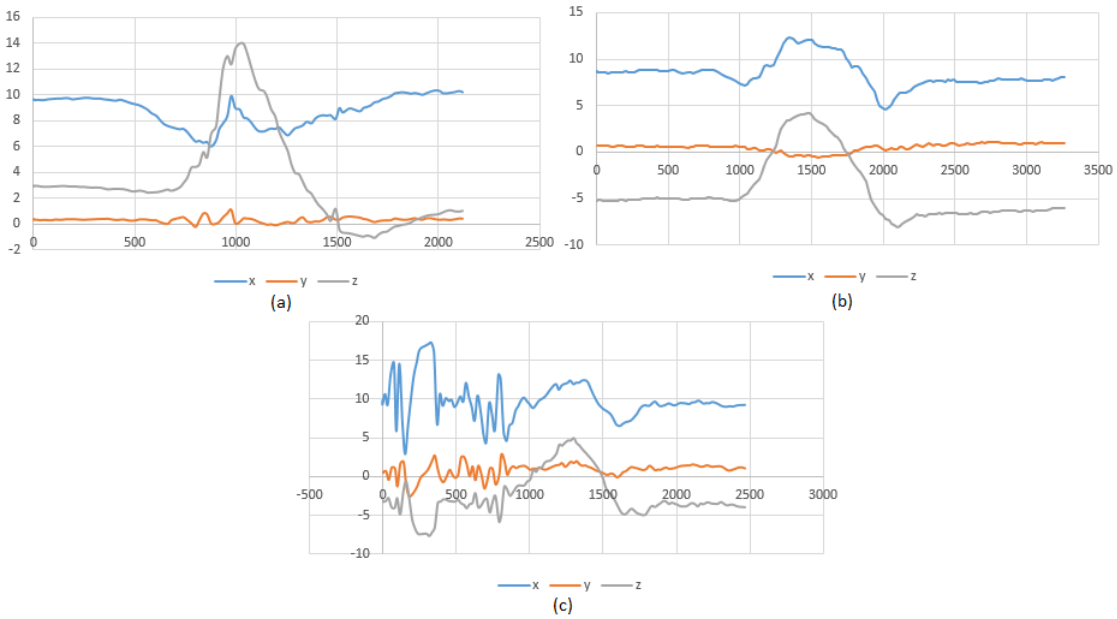
Grafik-grafik yang akan ditunjukkan pada bab ini akan memiliki beberapa karakteristik. Sumbu y pada grafik yang akan ditunjukkan akan merepresentasikan besar nilainya, Sedangkan sumbu x akan merepresentasikan waktunya. Grafik yang ditunjukkan akan memiliki beberapa nilai, tergantung dari jumlah nilai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

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

Perekaman Grafik Sensor *Accelerometer*

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

Gambar 13 merupakan grafik-grafik yang terbentuk dari nilai yang di terima oleh sensor *accelerometer* ketika pengguna sedang mengangguk. Pada Gambar 13 grafik (a) terlihat nilai z menaik dan nilai x menurun ketika sedang mengangguk. Tetapi nilai x kembali menaik ketika nilai z sudah hampir mencapai nilai tertinggi. Sedangkan nilai y terlihat cukup konstan di sekitar angka 0. Pada grafik (b) terlihat nilai x dengan y memiliki pola yang serupa ketika mengangguk. Kedua nilai menaik ketika



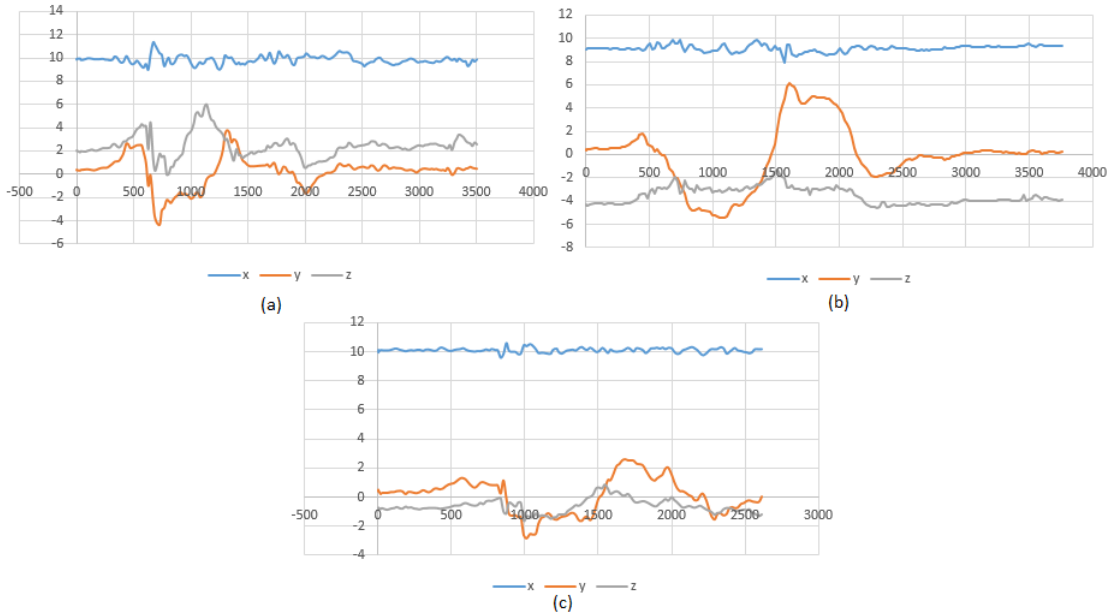
Gambar 13: Grafik nilai kuaternion dari sensor *accelerometer* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.

pengguna sedang mengangguk. Nilai x bermulai dari 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 13 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 13 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 14 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 14 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 terpengaruhi gerakan menggeleng.

Perekaman Grafik Sensor *Gyroscope*



Gambar 14: 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.

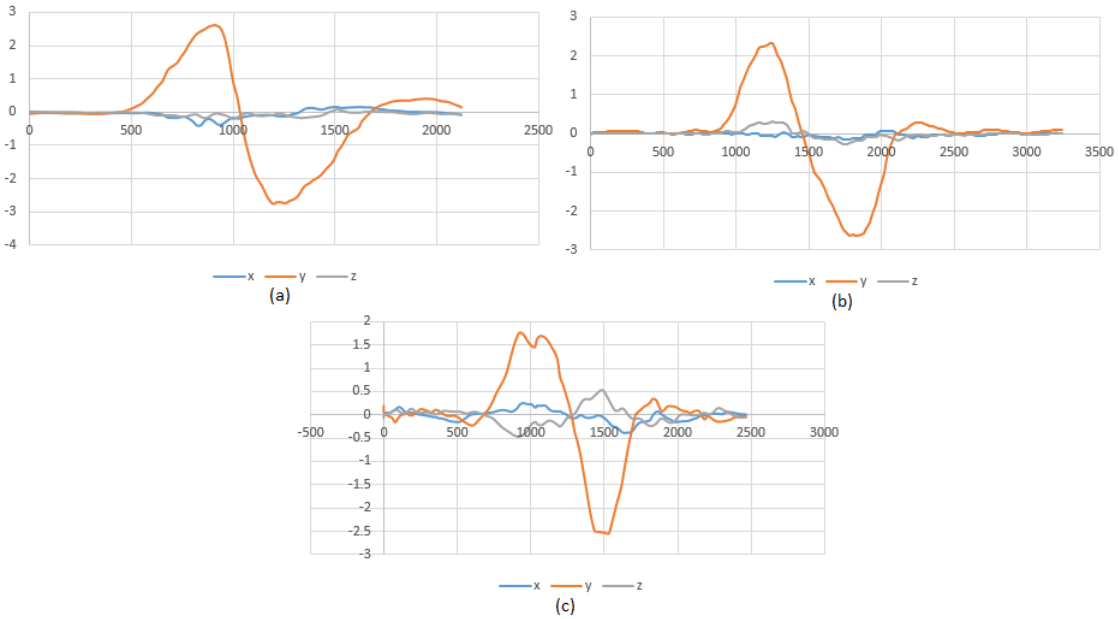
Perekaman menggunakan sensor *gyroscope* akan mendapatkan percepatan angular yang terjadi setiap waktunya. Berbeda dengan sensor *accelerometer* yang dapat terpengaruhi oleh lingkungan sekitar seperti gravitasi dan percepatan lainnya, sensor ini hanya akan merekam perputaran yang terjadi pada perangkat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

Gambar 15 menunjukkan nilai-nilai sensor *gyroscope* ketika pengguna sedang mengangguk. Grafik (a) membentuk sebuah bukit dan lembah pada nilai y. Bukit yang terjadi disini menunjukkan ketika pengguna menggerakan kepalanya kebawah, 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 15. Begitu pula pada grafik (c) yang memiliki pola yang serupa dengan yang grafik-grafik sebelumnya.

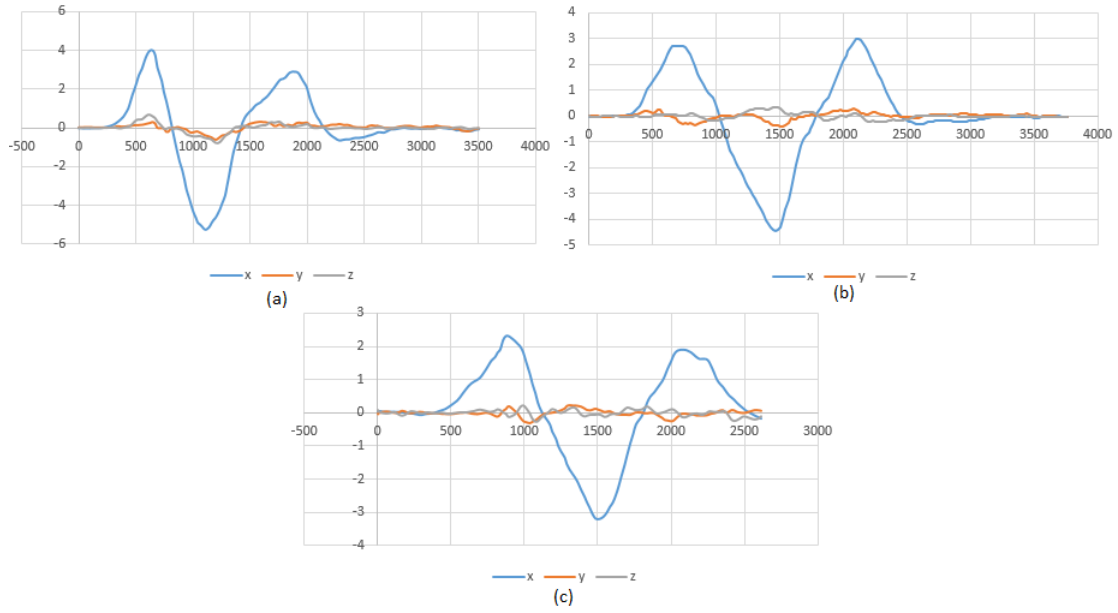
Gambar 16 menunjukkan nilai-nilai sensor *gyroscope* ketika pengguna sedang menggeleng. Pada grafik (a) nilai yang mengalami kenaikan dan penurunan adalah nilai x dan nilai-nilai lainnya cenderung berada pada nilai 0. Nilai x membentuk 2 buah bukit dan 1 buah lembah. Bukit pertama terjadi ketika pengguna menggerakan kepalanya ke kiri. Lembah pertama terjadi ketika pengguna menggerakan kepalanya ke kanan. Bukit kedua terjadi ketika pengguna menggerakan kepalanya 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 mempengaruhi sensor *gyroscope* dalam mendeteksi gerakan kepala. Nilai-nilai yang dikembalikan oleh sensor *gyroscope* memiliki pola grafik yang jauh lebih rapih 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 di pengaruhi gerakan kepala pada saat mengangguk. Oleh karena itu sensor gryoscope lebih baik dalam mendeteksi gerakan yang terjadi pada perangkat android.

Perekaman Grafik Sensor *Rotation Vector*

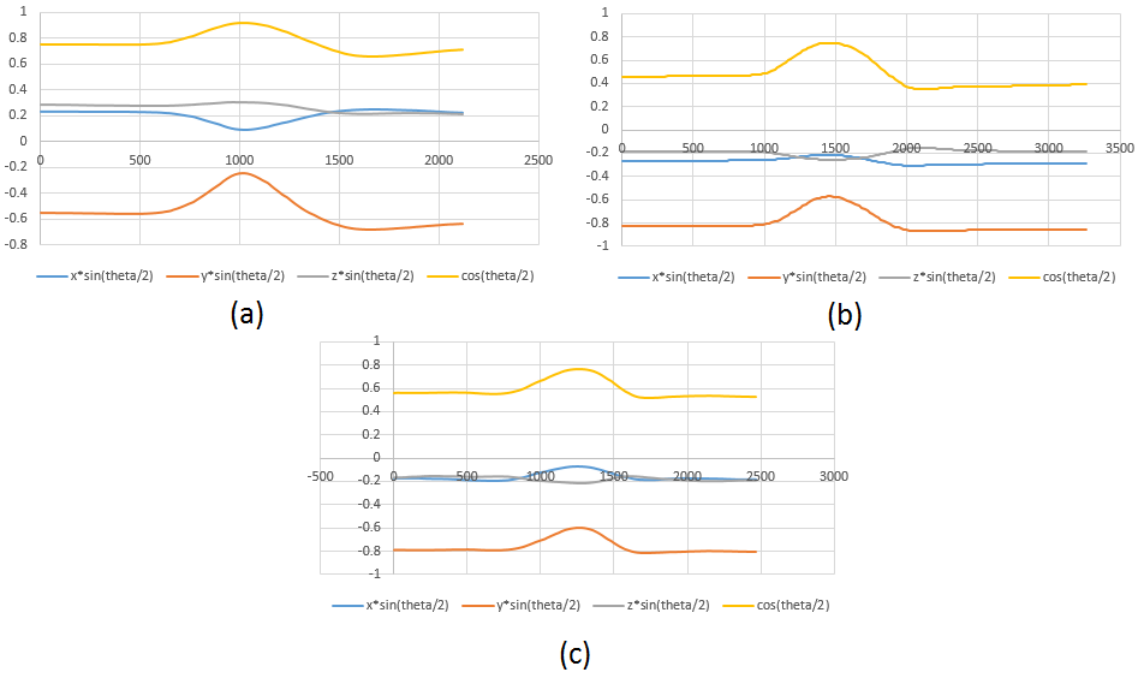


Gambar 15: Gambar grafik nilai sensor *gyroscope* ketika pengguna mengangguk dengan posisi muka awal (a) menghadap ke depan, (b) menghadap ke atas, (c) menghadap ke kiri atas.



Gambar 16: 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.

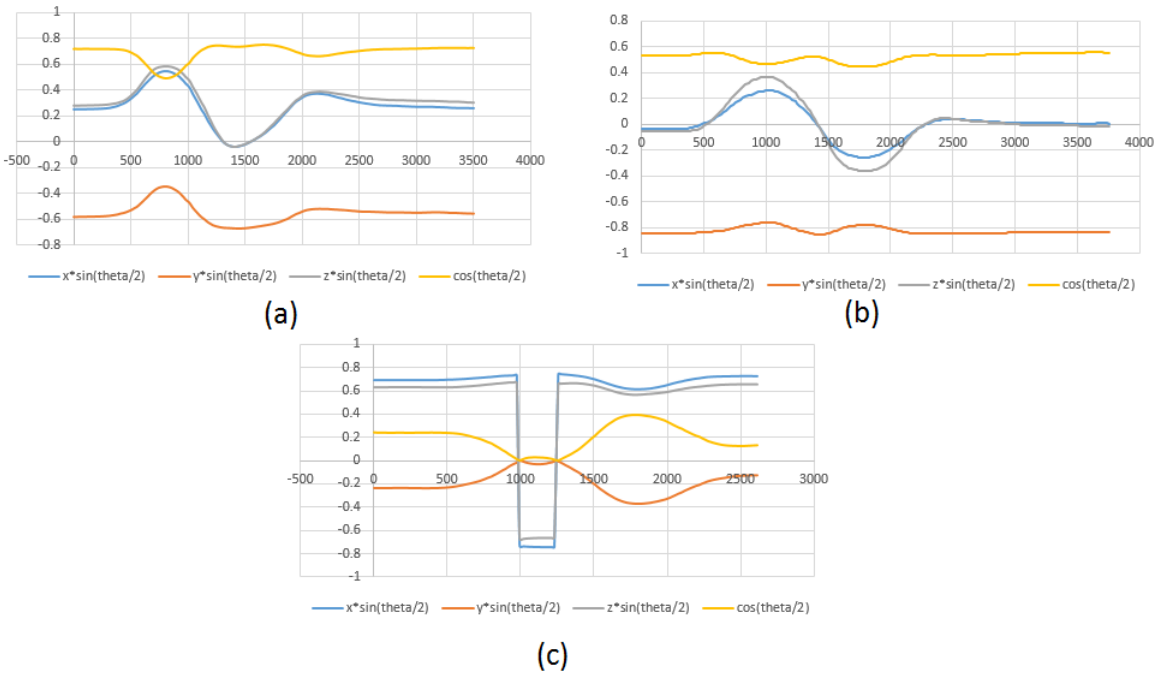
Perekaman menggunakan sensor *rotation vector* akan mendapatkan sebuah kuaternion yang merepresentasikan perputaran yang terjadi pada perangkat Android. Perputaran ini akan dideskripsikan dengan suatu vektor sebagai sumbu putarnya dan sudut perputarannya. Berbeda dengan sensor *gyroscope* yang merekam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



Gambar 17: 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.

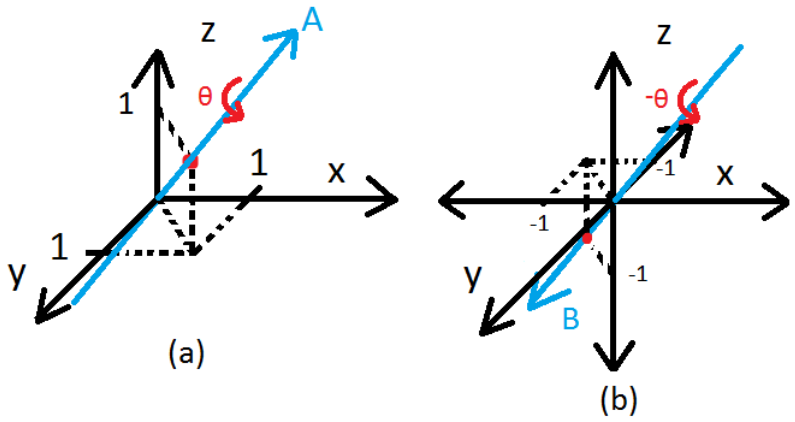
Gambar 17 menunjukkan nilai-nilai sensor *rotation vector* ketika pengguna sedang mengangguk. Pada grafik (a) terbentuk sebuah bukit pada garis berwarna jingga dengan kuning, dan lembah pada garis berwarna biru. Garis yang berwarna abu cenderung stabil di angka 0.3. Grafik (b) menunjukkan pola yang mirip 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 sebuat 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 pendeteksian gerakan kepala.

Gambar 18 menunjukkan nilai-nilai sensor *rotation vector* ketika pengguna sedang menggeleng. Pada grafik (a) terbentuk sebuh bukit yang diikuti dengan lembah pada garis berwarna biru dengan abu-abu. Garis berwarna kuning membentuk suatu lembah yang setelahnya cenderung konstan. Berbeda pada garis kuning yang membentuk bukit kemudian setelahnya cenderung konstan. Grafik (b) memiliki kemiripan dengan grafik (a), garis biru dengan garis abu-abu memiliki pola yang sama yaitu membentuk bukit dengan lembah ketika pengguna menggelengkan kepala. Garis kuning dengan jingga cenderung konstan, berbeda dengan grafik (a) yang membentuk bukit atau lembah. Pada grafik (c) garis-garis membentuk suatu pola yang tidak normal. Garis kuning membentuk sebuah lembah, tetapi membentuk suatu bukit kecil ketika nilainya mencapai nilai 0 pada milidetik ke 1000. Garis jingga memiliki pola yang berlawanan dengan garis kuning. Pada saat garis kuning dengan garis jingga mencapai angka 0 perubahan drastis pun terjadi pada garis biru dengan abu-abu. Pada milidetik ke 1000



Gambar 18: 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.

garis biru dengan abu mengalami perubahan nilai yang sangat drastis. Kedua nilai tersebut berubah dari nilai yang berkisar diantara 0.6 sampai 0.7 menjadi berkisar diantara -0.7 sampai -0.8. Kasus ini tidak berarti sensor sedang mengalami kegagalan akurasi (*accuracy fail*), tetapi memang seperti itulah karakteristik dari perputaran menggunakan kuaternion pada android. Perputaran yang terjadi pada batas mendekati sebelum terjadinya dengan setelah perubahan nilai yang drastis memiliki hasil perputaran yang sama. Hal ini disebabkan karena melakukan perputaran dengan vektor sebagai sumbu dapat memiliki 2 buah nilai yang sejenis. Dua buah nilai tersebut akan menghasilkan perputaran yang sama ketika arah vektor dengan arah putarnya di balikkan seperti yang ditunjukkan pada Gambar 19. Sepertinya pada sistem android nilai cos($theta/2$) didesain agar tidak bernilai negatif, sehingga nilai-nilai yang lainnya akan mengalami perubahan nilai yang drastis ketika nilai cos($theta/2$) mencapai angka 0.



Gambar 19: Dua buah rotasi yang identik dengan nilai kuaternion yang berbeda.

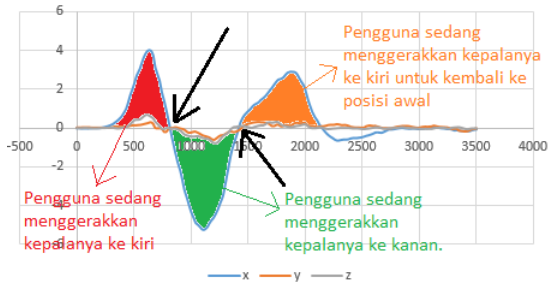
- Analisis Data Sensor untuk Mendeteksi Gerakan Kepala

Dari ketiga hasil pecobaan pada sensor-sensor pada bab 4 dapat disimpulkan bahwa sensor *gyroscope* adalah sensor yang terbaik untuk mendeteksi gerakan kepala. Data dari sensor *accelerometer* akan susah untuk digunakan dalam mendeteksi gerakan kepala karena terganggu dengan aktivitas-aktivitas diluar gerakan pengguna yang juga ikut terekam oleh sensor *accelerometer*. Data dari sensor *rotation vector* juga akan lebih rumit dibandingkan sensor *gyroscope*. Hal ini karena perputaran yang di rekam oleh sensor *rotation vector* merekam kondisi putar pada suatu saat. Hasil rekaman ini akan mempersulit pada saat pendeteksian gerakan kepala karena harus melakukan proses untuk menghitung kecepatan kepala bergerak, agar dapat membedakan gerakan mengangguk atau menggeleng atau sekedar menoleh biasa. Dalam mendeteksi gerakan mengangguk dengan menggeleng banyak batas-batas yang perlu diperhatikan. Batas-batas tersebut untuk mengetahui apakah pengguna benar-benar mengangguk atau menggeleng atau sekedar menoleh biasa. Batas-batas yang perlu diperhatikan adalah:

- Kecepatan pengguna menoleh.
- Jarak waktu pada saat pengguna melakukan melawan arah gerakan.
- Simpangan terbesar kepala saat mengangguk atau menggeleng.

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

Gerakan menggeleng atau mengangguk biasanya memiliki selang waktu yang sangat sempit, karena pengguna biasanya langsung melawan arah secara langsung ketika sedang mengangguk ataupun menggeleng. Nilai ini dapat diperoleh dengan menghitung jarak antara bukit dengan lembah yang terbentuk pada grafik seperti yang dijelaskan pada Gambar 20. 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 20: Deskripsi grafik pada saat pengguna menggeleng. Yang ditunjuk oleh panah berwarna hitam adalah selang waktu yang terjadi saat pengguna melawan arah gerakan kepala.

Simpangan terbesar ini juga penting untuk dijadikan batasan-batasan dalam mendeteksi gerakan mengangguk ataupun menggeleng. Simpangan kepala yang sangat kecil dapat diragukan untuk dianggap sebagai gerakan mengangguk atau menggeleng. Simpangan ini dapat diperoleh dengan menghitung luas yang dibentuk dari bukit atau lembah yang terbentuk pada grafik. Contoh pada Gambar 20 simpangan terbesar yang terjadi ketika pengguna menggerakkan kepalanya pertama kali ke kiri adalah luas pada bidang yang diarsir merah. Simpangan terbesar yang terjadi ketika pengguna menggerakan kepalanya ke kanan adalah luas bidang yang diarsir berwarna hijau dikurangi dengan luas pada

bidang yang diarsir merah. Pengurangan ini dilakukan karena luas bidang yang diarsir berwarna hijau merupakan simpangan yang terjadi setelah kepala sudah menghadap ke kiri. Begitu pula dengan luas bidang yang diarsir berwarna jingga yang akan dikurangi dengan hasil pengurangan luas sebelumnya.

5. Menganalisis metode pendeteksi gerakan kepala.
   status : Ada sejak rencana kerja skripsi.
   hasil :

   Seperti yang sudah dijelaskan pada bagian Android Sensor Framework, data akan didapatkan setiap ada perubahan nilai pada sensor dengan rentang waktu tertentu, bergantung dengan konfigurasinya. Pendeteksian ini membutuhkan data yang *real-time*, sehingga akan lebih baik jika menggunakan konfigurasi kecepatan pengambilan data setiap 20.000 mikrodetik. Penggunaan konfigurasi dengan kecepatan 0.000 mikrodetik tidak terlalu baik, karena akan menggunakan kemampuan processor yang sangat tinggi.

   Algoritma Mendeteksi Gerakan Mengangguk

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

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

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

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

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

Algorithm 1 Nod Detection Algoritm

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

Algoritma 1 akan mengembalikan nilai true jika pengguna sedang mengangguk. Baris ke-2 hingga baris ke-8 adalah untuk mengecek kecepatan putar sekarang, apakah sudah melampaui batas yang ditentukan atau belum. Untuk mengetahui titik potong secara akurat dapat menggunakan rumus persamaan garis dari dua buah titik yang dinotasikan dengan Notasi 5.

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1} \tag{5}$$

dengan,

$$\begin{aligned} y_1 =& lastYAngSpeed \\ y_2 =& yAngSpeed \\ x_1 =& lastT \\ x_2 =& currT \end{aligned}$$

Notasi 5 dijabarkan sesuai dengan penjabaran 6 sehingga menjadi rumus yang berada pada algoritma 1 baris ke-10. Nilai titik potong x ini juga menjadi indikator waktu untuk rentang waktu dari suatu bukit atau lembah, yang akan digunakan untuk mendapatkan jarak waktu pada saat pengguna melawan arah gerakan. Baris ke-14 hingga ke-25 pada algoritma 1 adalah untk mengecek apakah simpangan yang terjadinya sudah melewati batas yang ditetapkan atau belum dan mencatatnya ke dalam attribut `passLimitDownDeviation` atau `passLimitUpDeviation`. Pada baris ke-26 hingga ke-28 untuk memulai menghitung luas yang baru. Baris ke-32 dan baris ke-33 untuk menghitung luas ketika tidak berpotongan dengan sumbu x. Baris ke-36 untuk pengecekan jarak antara bukit dengan lembah dapat diselesaikan dengan mengurangi waktu mulai bukit dengan waktu akhir lembah. Sehingga hasil untuk mengecek apakah semua batas sudah terpenuhi dapat di selesaikan dengan operasi (AND) biasa.

$$\begin{aligned} y =& 0 \\ \frac{0 - y_1}{y_2 - y_1} =& \frac{x - x_1}{} \\ \frac{-y_1}{y_2 - y_1} x_2 - x_1 =& x - x_1 \\ x =& (\frac{-y_1}{y_2 - y_1} x_2 - x_1) + x_1 \end{aligned} \tag{6}$$

Algoritma Mendeteksi Gerakan Menggeleng

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

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

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

Sebagian besar algoritma untuk mendeteksi gerakan menggeleng memiliki kemiripan dengan algoritma untuk mendeteksi gerakan mengangguk. Pada algoritma menggeleng data waktu mulai dan berakhir-

Algorithm 2 Shook Detection Algoritm

```
1: function DETECTSHOOK(xAngSpeed)
2:     currT ←current Time
3:     if xAngSpeed > angSpeedLimit then
4:         currPassLimitLeft ← true
5:     else if xAngSpeed < angSpeedLimit ∗ −1 then
6:         currPassLimitRight ← true
7:     if X values intersect with x(time) axis then
8:         xIntersect ← ((−lastXAngSpeed/(xAngSpeed − lastXAngSpeed)) ∗ (currT − lastT)) + lastT
9:         endTCurrMotion ← xIntersect
10:        areaBeforeIntersect ← ((xIntersect − lastT)/1000) ∗ lastXAngSpeed/2
11:        calcArea ← calcArea + areaBeforeIntersect
12:        if currPassLimitRight then
13:            deviationMotionRight ← calcArea
14:            if deviationMotionRight < deviationLimit then
15:                valleyTimes.add(start and end time valley)
16:        else if currPassLimitLeft then
17:            deviationMotionUp = calcArea
18:            if deviationMotionUp > deviationLimit ∗ −1 then
19:                hillTimes.add(start and end time hill)
20:        calcArea ← 0
21:        areaAfterIntersect ← ((currT − xIntersect)/1000) ∗ xAngSpeed/2
22:        calcArea ← calcArea + areaAfterIntersect
23:        startTCurrMotion ← xIntersect
24:        currPassLimitRight ← false
25:        currPassLimitLeft ← false
26:    else
27:        calcArea ← calcArea + ((lastXAngSpeed + xAngSpeed)/1000) ∗ (currT − lastT)/2
28:    lastXAngSpeed ← xAngSpeed
29:    lastT ← 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:    elsereturn false
```

nya suatu bukit atau lembah disimpan pada attribut dengan tipe data *List*. Attribut-attribut yang menggunakan tipe data *List* ini adalah `valleyTimes` dan `hillTimes`. Data-data waktu terjadinya bukti atau lembah yang dimasukkan ke dalam *List* ini diartikan sudah memenuhi syarat dari batas-batas simpangan dan kecepatan sudutnya. `List` ini akan di kosongkan kembali jika salah satu *List*-nya sudah lebih dari dua atau kedua *List* tersebut sudah memiliki isi sebanyak dua, atau lebih. Untuk pengecekan jarak waktu antara lembah dengan bukit akan dilakukan sebanyak dua kali. Pengecekan pertama yaitu untuk pengecekan jarak waktu antara lembah atau bukit pertama dengan kedua. Pengecekan kedua yaitu untuk pengecekan jarak waktu antara lembah atau bukit kedua dengan ketiga. Penghitungan jarak waktu antara lembah atau bukit pertama dengan kedua dapat dilakukan dengan mengurangi waktu dimulainya lembah atau bukit kedua dengan waktu berakhirnya lembah atau bukit pertama. Begitu pula dengan penghitungan jarak waktu antara lembah atau bukit kedua dengan ketiga.

6. Merancang aplikasi untuk mendeteksi gerakan kepala
   status : Ada sejak rencana kerja skripsi.
   hasil : berdasarkan analisis singkat, tidak dilakukan analisis lebih jauh karena tidak diperlukan struktur data baru, karena sudah disediakan oleh OpenSteer versi terbaru

7. Mengimplementasikan algoritma pendeteksi gerakan kepala ke aplikasi *Virtual Reality*
   status : Ada sejak rencana kerja skripsi.
   hasil :

8. Melakukan pengujian terhadap fitur-fitur yang sudah dibuat.
   status : Ada sejak rencana kerja skripsi.
   hasil :

9. Menulis dokumen skripsi
   status : Ada sejak rencana kerja skripsi.
   hasil : Dokumen skripsi telah terisi hingga bab 3 dikurangi analisis aplikasi sejenis dan algoritma pendeteksi mengangguk.

Pustaka

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

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

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

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

[5] "Aldin dynamics." `http://www.aldindynamics.com/`. [Online; diakses 22-November-2016].

3 Pencapaian Rencana Kerja

Persentase penyelesaian skripsi sampai dengan dokumen ini dibuat dapat dilihat pada tabel berikut :

1*	2*(%)	3*(%)	4*(%)	5*	6*(%)
1	10	10			10
2	5	5			5
3	5	5			5
4	10	10			10
5	10	10			10
6	10		10		
7	20		20		
8	10		10		
9	20	10	10	menulis dokumen skripsi hingga bab 3 pada S1	10
Total	100	50	50		50

Keterangan (*)

1 : Bagian pengerjaan Skripsi (nomor disesuaikan dengan detail pengerjaan di bagian 5)

2 : Persentase total

3 : Persentase yang akan diselesaikan di Skripsi 1

4 : Persentase yang akan diselesaikan di Skripsi 2

5 : Penjelasan singkat apa yang dilakukan di S1 (Skripsi 1) atau S2 (skripsi 2)

6 : Persentase yang sidah diselesaikan sampai saat ini

Bandung, 24/11/2016

Ega Prianto

Menyetujui,

Nama: Pascal Alfadian
Pembimbing Tunggal