Implementasi Java Interface pada Pembuatan Aplikasi Multimedia Berbasis Android

Agung Prihandono dan Eddy Nurraharjo
Program Studi Teknik Informatika, Universitas Stikubank Semarang
email: agunganak@gmail.com, eddynurraharjo@gmail.com

Abstrak

Penelitian ini menunjukkan salah satu teknik pemrograman dalam pemanfaatan kelas interface pada bahasa Java untuk pembuatan aplikasi yang dijalankan pada perangkat sistem berbasis android. Dalam pembuatannya, aplikasi dipecah menjadi beberapa modul. Hal ini dilakukan dengan tujuan untuk memudahkan dalam pengembangan kode program itu sendiri. Kelas interface ini bertugas sebagai perantara antar kelas juga dengan sistem android.

Kata kunci: Java Interface, Multimedia, Android

PENDAHULUAN

Teknologi mobile berkembang sangat pesat, masing-masing vendor berlomba-lomba mengusung teknologi baru, baik teknologi perangkat keras maupun perangkat lunak. Perkembangan tersebut bertujuan salah satunya untuk memudahkan dan memanjakan para calon penggunanya. Salah satu teknologi yang perkembangannya sangat pesat adalah perangkat mobile berbasis android.

Android merupakan sebuah sistem operasi yang sebelumnya dikembangkan khusus untuk telepon genggam. Namun memiliki kemampuan yang setara dengan sistem operasi sebuah komputer pribadi yang berbasis pada sistem operasi Linux.

Android didasarkan pada sistem *open source*. Sehingga para pengembang punya kesempatan yang sama untuk berkreasi mengembangkan dan memanfaatkan sumber daya teknologi yang tersedia pada sebuah perangkat.

Java merupakan salah satu bahasa pemrograman paling popular. Maka dengan kemampuan yang dimilikinya, pengembang dapat dengan mudah membangun suatu kerangka kerja aplikasi yang dinamis. Yang dapat menyesuaikan dengan perkembangan teknologi tanpa harus merombak secara total dari aplikasi yang pernah dibuat ketika aplikasi tersebut akan dikembangkan ke versi berikutnya, atau ketika ingin menambahkan fitur-fitur baru. Salah satunya adalah dengan memanfaatkan *interface* dari bahasa pemrograman java. Penggunaan *interface* sangat membantu dalam mendesain dan membangun kerangka kerja / framework dari suatu aplikasi di android.

Selain perkembangan teknologi pada perangkat mobile, hal yang selalu berubah lainnya adalah konten. Melalui penelitian ini, bermaksud untuk menganalisis bagaimana cara android menangani konten-konten multimedia yang semakin berubah baik dari segi ukuran maupun kualitas konten. Seperti konten gambar yang memiliki resolusi beragam, atau konten audio bahkan animasi.

Dengan memanfaatkan interface yang dimiliki bahasa pemrograman java, diharapkan agar penanganan masing-masing konten dapat dilakukan secara modular. Sehingga memudahkan ketika terjadi perubahan spesifikasi konten, tidak harus mengubah seluruh sistem.

Permasalahan yang diteliti adalah sebagai berikut:

a. Bagaimana sistem android menangani konten multimedia yang berupa teks, gambar, audio, hingga animasi grafis.

b. Bagaimana pemanfaatan java interface dalam pengembangan aplikasi multimedia pada perangkat android

TUJUAN DAN MANFAAT

Penelitian ini bertujuan untuk mengetahui penanganan sistem android terhadap konten-konten grafis dalam pembuatan suatu aplikasi multimedia dengan mengoptimalkan fungsi interface yang dimiliki oleh bahasa pemrograman java. Interface digunakan sebagai perantara komunikasi antar kelas yang berjalan pada platform android. Dibuat secara modular sehingga akan memudahkan dalam pengembangan dan pembaruan kode program.

Manfaat yang ingin dicapai pada penelitian ini adalah untuk memberikan wawasan tentang bagaimana menerapkan interface pada sistem berbasis android. Sekaligus memberikan gambaran pembuatan kerangka-kerangka aplikasi multimedia.

1. Pembahasan

Pemrograman berorientasi objek memungkinkan kita untuk memodelkan suatu kasus di dunia nyata untuk diterapkan ke dalam sistem yang terkomputerisasi. Dengan mengetahui konsep dasar sebuah objek di dunia nyata, maka kita dapat merepresentasikan objek tersebut berdasarkan dua buah karakteristik yaitu keadaan sebuah objek dan perilakunya. Kelas merupakan sebuah purwarupa dari objek yang dibuat. Kelas berisikan definisi-definisi yang terdiri dari keadaan suatu objek ke dalam bentuk atribut dan perilaku objek ke dalam bentuk metode/*method*.

Suatu kelas dapat diakses secara langsung maupun melalui perantara sebuah *interface*. *Interface* dibuat untuk menjadi antarmuka bagi sebuah kelas dengan dunia luar. Interface berisikan definisi-definisi yang ringkas atas perilaku suatu objek yaitu kumpulan method yang saling berkaitan tetapi tidak memiliki badan program. Dengan demikian dunia luar tidak perlu tahu detail implementasi dari objek yang diakses. Hal ini menjadi salah satu kelebihan penggunaan interface pada sistem berbasis android yaitu mengizinkan kita untuk terkonsentrasi pada semantik dari aplikasi yang dibangun tanpa harus menunjukkan detail implementasi program dan mengizinkan kita untuk dapat mengubah cara implementasi secara mudah dan terpisah (Zechner dan Green, 2012).

Perantara lainnya adalah kelas abstrak. Kelas abstrak memiliki kesamaan dengan interface, yaitu untuk merancang suatu aplikasi berbasis objek tetapi bukan untuk membuat implementasi dari aplikasi yang akan digunakan. Keduanya memiliki karakteristik yang sama seperti kelas pada umumnya, yaitu berisi nama kelas, atribut dan method. Dikarenakan kelas abstrak dan interface tidak mengimplementasikan detail program, maka kelas abstrak dan interface hanya menyertakan nama-nama method tanpa badan program.

Namun di antara kelas abstrak dan interface punya beberapa perbedaan seperti ditunjukkan pada Tabel 1.

Tabel 1. Perbandingan interface dan kelas abstrak

No	Interface	Kelas abstrak
1	Dapat mendeklarasikan daftar method	Dapat mendeklarasikan daftar method
2	Tidak memiliki kode pada masing-masing method	Kode program pada masing-masing method dapat didefinisikan
3	Hanya mendeklarasikan konstanta	Dapat mendeklarasikan berbagai jenis tipe variabel
4	Tidak memiliki konstruktor	Dapat memiliki konstruktor
5	Mengizinkan pewarisan lebih dari satu	Hanya mengizinkan satu buah pewarisan
6	Tidak memiliki hirarki teratas	Selalu diwariskan dari kelas object
7	Mengizinkan interface induk	Hanya memiliki sebuah kelas induk

	lebih dari satu	

2. Implementasi Kelas *Interface*

Interface yang menangani konten multimedia untuk aplikasi pada perangkat android dibuat sebagai antarmuka antar proses yang terdapat di dalam aplikasi. Dengan menggunakan android developer tool sebagai alat bantu dalam membuat projek dan mengkompilasi menjadi sebuah aplikasi.

Pembuatan aplikasi dipecah ke dalam beberapa bagian yang selanjutnya diintegrasikan menjadi sebuah kerangka kerja/*framework* untuk memudahkan dalam membangun komunikasi antar proses di dalam sistem dengan platform android.

Setiap modul dapat memiliki setidaknya sebuah *interface* atau lebih. Dan masing-masing *interface* memiliki setidaknya satu buah kelas yang mengimplementasikannya.

2.1 Modul Masukan

Modul masukan ini menangani bagaimana pengguna dapat berinteraksi dengan aplikasi melalui beberapa alat masukan antara lain fungsi sentuh, penekanan pada tombol tertentu, atau masukan berupa pembacaan sensor asccelerometer.

Sebuah perangkat berbasis android, memiliki tiga buah metode masukan yang dikenal yaitu layar sentuh, papan ketik dan sensor (*accelerometer*). Masing-masing masukan tersebut dapat membangkitkan berbagai *event* tertentu. Untuk layar sentuh membangkitkan event sebagai berikut:

a. Touch-down: event yang terjadi ketika jari menyentuh layar

b. Touch-drag: event yang terjadi ketika jari digerakkan secara mengusap di atas layar.

c. Touch-up: event yang terjadi ketika jari diangkat/melepaskan dari layar.

Setiap event sentuh akan merekam informasi tentang posisi sentuh jari terhadap komponen antarmuka pengguna yang sedang aktif dan sebuah pointer indeks yang digunakan untuk merekam ketika jari yang menyentuh layar lebih dari satu jari.

Selanjutnya untuk papan ketik/*keyboard* akan membangkitkan dua buah event yaitu:

a. Key-down: event ketika sebuah tombol ditekan.

b. Key-up: even ketika sebuah tombol dilepas.

Event pada papan ketik ini menyimpan informasi tentang key-code dari tombol yang ditekan beserta karakter *unicode* yang sesuai dengan tombol tersebut.

Event yang terakhir yaitu pembacaan sensor accelerometer. Sensor ini memberikan nilai keluaran yang berkaitan dengan posisi perangkat terhadap gravitasi bumi. Yaitu nilai koodrinat x, y, dan z seperti yang ditunjukkan pada Gambar 1.

Gambar 1 Orientasi arah sensor sebuah perangkat android

Nilai untuk masing masing sumbu dinotasikan dengan meter perdetik kuadrat (m/s^2). Di mana merepresentasikan bahwa sebuah benda akan melakukan percepatan pada $9{,}8 m/s^2$ ketika benda tersebut jatuh bebas.

Dari tiga macam masukan di atas, *interface* yang dibuat untuk keperluan tersebut ditunjukkan pada kode program berikut ini:

```
public interface Input {

public interface Masukan {
        public static class EventTombol {
                public static final int
KEY_DOWN = 0;
                public static final int KEY_UP
= 1;
                public int type;
                public int keyCode;
                public char keyChar;
```

```
        }
    public static class EventSentuh {
        public static final int
TOUCH_DOWN = 0;
        public static final int
TOUCH_UP = 1;
        public static final int
TOUCH_DRAGGED = 2;
        public int type;
        public int x, y;
        public int pointer;}
    public boolean isKeyPressed(int
keyCode);
    public boolean isTouchDown(int
pointer);
    public int getTouchX(int pointer);
    public int getTouchY(int pointer);
    public float getAccelX();
    public float getAccelY();
    public float getAccelZ();
    public List<KeyEvent>
getKeyEvents();
    public List<TouchEvent>
getTouchEvents();
}
```

Gambar 2. Kode Program Masukan

2.2 Berkas I/O

Bagian ini menangani bagaimana aplikasi mengakses berkas eksternal yang diletakkan sebagai aset dari aplikasi. Meembaca atau menulis suatu berkas merupakan hal yang mendasar dalam pembuatan aplikasi multimedia. Dalam pemrograman java, mekanisme yang sering digunakan adalah dengan membuat instan dari InputStream dan OutputStream. Adapun interface yang menangani berkas I/O ditunjukkan pada kode program berikut ini.

```
public interface FileIO {
    public InputStream readAsset(String
fileName) throws IOException;
    public InputStream readFile(String
fileName) throws IOException;
    public OutputStream writeFile(String
fileName) throws IOException;
}
```

Gambar 3. Kode Program Berkas I/O

2.3 Audio

Bagian ini mengatur bagaimana aplikasi dalam memuat dan memainkan berkas audio. Tanpa melakukan berbagai pengolahan pada berkas audio. Dalam memainkan berkas audio, mekanismenya dibagi menjadi dua macam yaitu memainkan audio yang menjadi suara latar dari aplikasi dan memainkan suara efek. Perancangan kode interface ditunjukkan pada kode program berikut ini.

```
public interface Audio {
    public SuaraLatar newMusic(String
filename);
    public SuaraEfek newSound(String
filename);
}
public interface SuaraEfek {
    public void play(float volume);
    public void dispose();
}
public interface SuaraLatar {
    public void play();
    public void stop();
    public void pause();
    public void setLooping(boolean
looping);
    public void setVolume(float volume);
    public boolean isPlaying();
    public boolean isStopped();
    public boolean isLooping();
    public void dispose();
}
```

Gambar 4. Kode Program Audio

2.4 Grafis

Bagian ini merupakan bagian yang kompleks berkaitan dengan aplikasi yang hendak dibuat. Namun secara khusus bertugas mengelelola bagaimana aplikasi memuat dan menampilkan beragam citra ke layar. Grafis yang dibuat dalam bentuk 2 dimensi.

a. Koordinat Layar

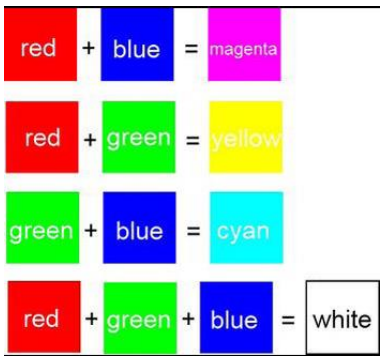
Sistem koordinat pada perangkat penampil elektronik berbasiskan raster. Atau yang lebih dikenal dengan sebutan *pixel*. Pixel memiliki dua buat atribut yaitu posisi dan warna. Posisi dari suatu pixel inilah yang merepresentasikan koordinat dari layar penampil tersebut. Di mana berupa nilai diskrit. Koordinat ini dimulai

dengan koordinat (0,0) dari pojok kiri atas dan selalu bernilai positif. Koordinat layar sebuah perangkat ditunjukkan pada gambar 5.

Gambar 5. Koordinat layar dan VRAM

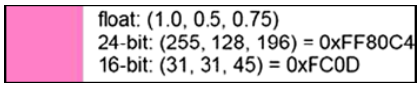
b. Warna

Model warna yang diterapkan pada pemrograman grafis adalah menggunakan model RGB. RGB merupakan singkatan dari warna merah, hijau dan biru. Untuk menghasilkan warna lain maka dapat diperoleh dengan mengkombinasikan nilai RGB seperti ditunjukkan pada Gambar 6.



Gambar 6. Penggabungan warna

Terdapat beberapa cara untuk mengkomputasikan warna sebuah pixel ke dalam bentuk kode yang kemudian disimpan oleh memori tergantung pada tipe data yang hendak digunakan. Dapat berupa bilangan pecahan maupun bilangan bulat positif. Tetapi hal tersebut akan mempengaruhi konsumsi memori. Salah satu contoh pengkodean warna ditunjukkan seperti pada Gambar 7.



Gambar 7. Pengkodean warna

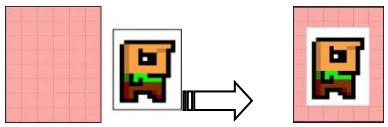
- Pengkodean RGB dengan tipe data pecahan 32-bit, sehingga satu buah pixel mendapat alokasi memori sebesar 12-byte dengan rentang nilai 0,0 – 1,0
- Pengkodean RGB 24-bit mendapat alokasi untuk satu pixel sebesar 3 atau 4 byte dengan intensitas berada pada rentang nilai 0 sampai 255. Terdapat dua buah urutan pengkodean yaitu RGB dan BGR yang dikenal dengan RGB888 atau BGR888 di mana angka delaman merepresentasikan jumlah bit untuk masing-masing elemen warna.
- Pengkodean RGB 16-bit mengalokasikan 2 byte untuk satu buah pixel. Warna merah dan biru memiliki intensitas dengan rentang nilai 0-31 sedangkan warna hijau memiliki intensitas dengan rentang nilai 0-63.

c. Format Citra dan Kompresinya

Pemilihan format dari citra yang akan digunakan sangat mempengaruhi ukuran dari citra itu sendiri. Sehingga akan berdampak pemborosan memori dan kinerja aplikasi menjadi lebih berat dalam memuat citra tersebut. Format yang paling umum dengan kompresi yang baik dan dapat diterapkan pada sistem android, adalah citra dengan tipe JPEG dan PNG.

d. Penggabungan Citra

Hal terakhir yang perlu menjadi perhatian dalam menampilkan citra ke layar adalah bagaimana menggabungkan dua buah citra. Contoh penggabungan ditunjukkan pada Gambar 8.



Gambar 8. Penggabungan dua citra

Terlihat bahwa citra hasil mengandung efek warna putih pada tepian citra dari karakter asli. Permasalahan ini dapat dipecahkan dengan mekanisme penggabungan gambar *alpha* (*alpha blending*). Pada kasus yang ditunjukkan gambar 8 secara teknis disebut sebagai *alpha masking*.

Dari tiga format pengkodean warna yang telah dijelaskan sebelumnya, kita dapat menyimpan pengkodean RGB 24-bit pada 32-bit bilangan bulat. Sehingga diperoleh 8 bit yang belum digunakan. 8 bit tersebut kemudian

dimanfaatkan untuk menyimpan nilai *alpha*, dengan rentang nilai dari 0 sampai 255. Di mana nilai 0 adalah transparan dan nilai 255 berwarna normal. Pengkodean ini kemudian dikenal dengan ARGB8888 atau BGRA8888.

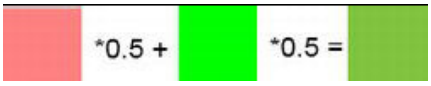
Salah satu Algoritma yang digunakan untuk menggabungkan dua buah citra dapat menggunakan persamaan 1. Namun untuk kesederhanaan logika komputasi, persamaan ini hanya digunakan untuk mengkomputasi nilai pixel R, G dan B tanpa mengkomputasi nilai pixel alpha.

red = src.red * src.alpha + dst.red * (1 – src.alpha)

blue = src.green * src.alpha + dst.green * (1 – src.alpha)

green = src.blue * src.alpha + dst.blue * (1 – src.alpha) …… (1)

Src dan *dst* adalah nilai pixel dari citra asal dan citra tujuan yang hendak digabung. Contoh penerapan persamaan 1 ditunjukkan pada Gambar 9. Di mana mengilustrasikan penggabungan dua buah citra dengan warna merah muda dan hijau terang, menghasilkan citra dengan warna hijau gelap.



Gambar 9. Penggabungan dua warna

Jika diperhatikan, bahwa persamaan 1 tersebut memiliki komputasi perkalian yang banyak. Jika memungkinkan hal ini harus dihindari dalam proses penggabungan citra terlebih jika media komputasi berupa perangkat mobile. Kita dapat melakukan proses pra-perkalian antara nilai pixel RGB dengan nilai *alpha*-nya.

Untuk menampilkan citra dengan API grafis harus memperhatikan persamaan yang sesuai. Citra yang hendak digunakan pun harus dipastikan apakah mengandung nilai *alpha* atau tidak. Berkaitan dengan hal ini, API grafis yang dimiliki sistem Android memberikan kemudahan dalam proses penggabungan citra.

Perancangan kode interface dibuat berdasarkan kebutuhan mengolah grafis sebagai berikut ini:

- Memuat citra dari media penyimpan kemudian menyimpannya ke memori
- Membersihkan *framebuffer* dan menyiapkannya dengan sebuah warna sehingga tidak ada sisa dari frame sebelumnya.
- Mengatur nilai pixel pada area pixel tertentu dengan warna tertentu
- Gambar garis dan persegi panjang pada *framebuffer*
- Gambar citra yang telah dimuat ke dalam memori sebelumnya pada *framebuffer*
- Baca ukuran dimensi dari *framebuffer*

Dari kebutuhan di atas, kemudian dibuat dalam bentuk kode interface seperti pada kode berikut ini.

```
public interface Graphics {
        public static enum PixmapFormat {
                ARGB8888, ARGB4444,
RGB565
        }
        public Pixmap newPixmap(String
fileName, PixmapFormat format);
        public void clear(int color);
        public void drawPixel(int x, int y, int
color);
        public void drawLine(int x, int y, int
x2, int y2, int color);
        public void drawRect(int x, int y, int
width, int height, int color);
        public void drawPixmap(Pixmap
pixmap, int x, int y, int srcX, int srcY,
                        int srcWidth, int
srcHeight);
        public void drawPixmap(Pixmap
pixmap, int x, int y);
        public int getWidth();
        public int getHeight();
}
```

```
public interface Pixmap {
        public int getWidth();
```

```
        public int getHeight();
        public PixmapFormat getFormat();
        public void dispose();
}
```

Gambar 9. Kode Program Mengolah Grafis

2.5 Pengelola Tampilan

Modul masukan ini Bertanggungjawab untuk mengelola pembuatan tampilan dan mengatur bagaimana menutup sebuah tampilan, memberhentikan atau menjalankan kembali sebuah tampilan. Sekaligus sebagai program utama yang menggabungkan interface-interface dan digunakan untuk membuat sebuah aplikasi multimedia.

Adapun fitur yang diperlukan dalam membangun aplikasi multimedia adalah sebagai berikut:

- Menyiapkan window dan antarmuka pengguna dan mengesetnya agar dapat menangkap window dan *event* masukan.
- Jalankan thread utama
- Rekam setiap aktivitas tampilan layar
- Berikan akses penuh untuk masing-masing modul lainnya

Kode interface untuk rancangan di atas ditunjukkan pada kode program berikut ini

```
public interface ApMultimedia {
        public Input getInput();
        public FileIO getFileIO();
        public Graphics getGraphics();
        public Audio getAudio();
        public void setScreen(Screen screen);
        public Screen getCurrentScreen();
        public Screen getStartScreen();
}
```

Gambar 10. Kode Program Mengelola Tampilan

Bagian terakhir adalah pembuatan kelas abstrak Layar. Pemilihan penggunaan kelas abstrak daripada menggunakan kelas interface adalah untuk proses implementasi pemakaian layar yang hanya dapat menampilkan satu tampilan dalam satu layar.

```
public abstract class Layar {
        protected final ApMultimedia apmult;
        public Screen(ApMultimedia apmult) {
                this.apmult = apmult;
        }
        public abstract void update(float
deltaTime);
        public abstract void present(float
deltaTime);
        public abstract void pause();
        public abstract void resume();
        public abstract void dispose();
}
```

Gambar 11. Kode Program Satu Tampilan dalam Satu Layar

3. Implementasi Aplikasi Multimedia

Implementasi dari masing-masing interface adalah sebagai berikut

a. Implementasi Masukan

```
public class ImplInput implements Input {
        AccelerometerHandler accelHandler;
        KeyboardHandler keyHandler;
        TouchHandler touchHandler;
        public ImplInput(Context context, View
view, float scaleX, float scaleY) {
        accelHandler = new
AccelerometerHandler(context);
        keyHandler = new
KeyboardHandler(view);
        if (Integer.parseInt(VERSION.SDK) <
5)
        touchHandler = new
SingleTouchHandler(view, scaleX, scaleY);
        else touchHandler = new
MultiTouchHandler(view, scaleX, scaleY);
        }
public boolean isKeyPressed(int keyCode)
{return keyHandler.isKeyPressed(keyCode);}
public boolean isTouchDown(int pointer)
{return touchHandler.isTouchDown(pointer);
        }
public int getTouchX(int pointer) {return
touchHandler.getTouchX(pointer);}
public int getTouchY(int pointer) {return
touchHandler.getTouchY(pointer);}
public float getAccelX() {return
accelHandler.getAccelX();}
```

```
public float getAccelY() {return
accelHandler.getAccelY();}
public float getAccelZ() {return
accelHandler.getAccelZ();}
public List <TouchEvent> getTouchEvents()
{return touchHandler.getTouchEvents();}
public List <KeyEvent> getKeyEvents()
{return keyHandler.getKeyEvents();}
}
```

Gambar 12. Kode Program Interface Masukan

b. Implementasi Berkas I/O

```
public class ImplFileIO implements FileIO {
        Context context;
        AssetManager assets;
        String externalStoragePath;
public ImplFileIO(Context context) {
 this.context = context;  this.assets =
context.getAssets();
this.externalStoragePath =
Environment.getExternalStorageDirectory()
                        .getAbsolutePath() +
File.separator;}
public InputStream readAsset(String fileName)
throws IOException {
        return assets.open(fileName);   }
public InputStream readFile(String fileName)
throws IOException {
        return new
FileInputStream(externalStoragePath +
fileName);       }
public OutputStream writeFile(String fileName)
throws IOException {
        return new
FileOutputStream(externalStoragePath +
fileName);       }
public SharedPreferences getPreferences() {
        return
PreferenceManager.getDefaultSharedPreference
s(context);       }
}
```

Gambar 13. Kode Progam Interface I/O

c. Implementasi Audio, SuaraEfek, SuaraLatar

```
public class ImplAudio implements Audio {
        AssetManager assets;
        SoundPool soundPool;
        public ImplAudio(Activity activity) {
        activity.setVolumeControlStream(Audio
Manager.STREAM_MUSIC);
        this.assets = activity.getAssets();
        this.soundPool = new SoundPool(20,
AudioManager.STREAM_MUSIC, 0);
        }
        @Override
        public Music newMusic(String
filename) {
        try {    } catch (IOException e) {

        }}
        @Override
        public Sound newSound(String
filename) {
        try {
        AssetFileDescriptor assetDescriptor =
assets.openFd(filename);
        int soundId =
soundPool.load(assetDescriptor, 0);
        return new ImplSound(soundPool,
soundId);
        } catch (IOException e) {
        throw new
RuntimeException("Couldn't load sound '" +
filename + "'");
        }}
}
```

Gambar 14. Kode Program Interface Audio

```
public class ImplSound implements SuaraEfek
{
        int soundId;      SoundPool soundPool;
public ImplSound(SoundPool soundPool, int
soundId) {
this.soundId = soundId;this.soundPool =
soundPool;}
        @Override
public void play(float volume)
{soundPool.play(soundId, volume, volume, 0, 0,
1);       }
        @Override
public void dispose()
{soundPool.unload(soundId);    }
}
```

Gambar 15. Kode Program Interface Suara Efek

```
public class ImplMusic implements SuaraLatar,
OnCompletionListener {
```

```
        MediaPlayer mediaPlayer;
        boolean isPrepared = false;
public ImplMusic(AssetFileDescriptor
assetDescriptor) {
        mediaPlayer = new MediaPlayer();
try {} catch (Exception e) {throw new
RuntimeException("Couldn't load music");
        }}
        @Override
public void play() {if
(mediaPlayer.isPlaying())return;}}
        @Override
public void stop()
{mediaPlayer.stop();synchronized (this)
{isPrepared = false;}}
        @Override
public void pause() {if
(mediaPlayer.isPlaying())mediaPlayer.pause();}
        @Override
public void setLooping(boolean isLooping)
{mediaPlayer.setLooping(isLooping);    }
        @Override
public void setVolume(float volume)
{mediaPlayer.setVolume(volume, volume);
        }
        @Override
public boolean isPlaying() {return
mediaPlayer.isPlaying();          }
        @Override
public boolean isStopped() {return
!isPrepared;      }
        @Override
public boolean isLooping() {return
mediaPlayer.isLooping();          }
        @Override
public void dispose() {if
(mediaPlayer.isPlaying())mediaPlayer.stop();
mediaPlayer.release();}
        @Override
public void onCompletion(MediaPlayer arg0)
{synchronized (this) {isPrepared = false;}}
}
```

Gambar 16. Kode Program Interface Suara Latar

d. Implementasi Grafis

```
public class ImplGraphics implements
Graphics{
        AssetManager assets;    Bitmap
frameBuffer;
        Canvas canvas; Paint paint;
        Rect srcRect = new Rect();       Rect
dstRect = new Rect();
        public ImplGraphics(AssetManager
assets, Bitmap frameBuffer) {
        this.assets = assets;this.frameBuffer =
frameBuffer;
        this.canvas = new
Canvas(frameBuffer);this.paint = new Paint();
        }
public Pixmap newPixmap(String fileName,
PixmapFormat format) {
        try {in = assets.open(fileName);
                …
                in.close();
        } catch (IOException e) {}}}
        return new AndroidPixmap(bitmap,
format);
        }
public void clear(int color) {
canvas.drawRGB((color & 0xff0000) >> 16,
(color & 0xff00) >> 8,
                (color & 0xff)); }
public void drawPixel(int x, int y, int color) {
        paint.setColor(color);canvas.drawPoint(
x, y, paint);      }
public void drawLine(int x, int y, int x2, int y2,
int color) {
        paint.setColor(color);canvas.drawLine(x
, y, x2, y2, paint);}
        public void drawRect(int x, int y, int
width, int height, int color) {
        paint.setColor(color);paint.setStyle(Styl
e.FILL);
        canvas.drawRect(x, y, x + width - 1, y +
width - 1, paint);}
        public void drawPixmap(Pixmap
pixmap, int x, int y, int srcX, int srcY,
                        int srcWidth, int
srcHeight) {
        srcRect.left = srcX;srcRect.top = srcY;
        srcRect.right = srcX + srcWidth - 1;
        srcRect.bottom = srcY + srcHeight - 1;
        dstRect.left = x;dstRect.top = y;
        dstRect.right = x + srcWidth –
1;dstRect.bottom = y + srcHeight - 1;
        canvas.drawBitmap(((AndroidPixmap)
pixmap).bitmap, srcRect, dstRect, null);}
        public void drawPixmap(Pixmap
pixmap, int x, int y) {
```

```
        canvas.drawBitmap(((AndroidPixmap)p
ixmap).bitmap, x, y, null);}
        public int getWidth() {return
frameBuffer.getWidth();}
        public int getHeight() {return
frameBuffer.getHeight();}
}
```

Gambar 17. Kode Program Interface Grafis

e. Implementasi Layar

```
public class AndroidApMultimedia extends
Activity implements ApMultimedia{
…
        }
        @Override
public void onResume()
{super.onResume();wakeLock.acquire();
        screen.resume();
        renderView.resume();    }
        @Override
public void onPause() {super.onPause();
        wakeLock.release();
        renderView.pause();      screen.pause();
        if (isFinishing())
        screen.dispose();}
        @Override
public Input getInput() {return input;}
        @Override
public FileIO getFileIO() {return fileIO;}
        @Override
public Graphics getGraphics() {return
graphics;          }
        @Override
public Audio getAudio() {return audio;}
        @Override
public void setScreen(Screen screen) {
        if (screen == null)
        throw new
IllegalArgumentException("Screen must not be
null");
        this.screen.pause();
        this.screen.dispose();
        screen.resume();
        screen.update(0);
        this.screen = screen;
        }
        @Override
public Screen getCurrentScreen() {       return
screen;}
        @Override
public Screen getStartScreen() {return null;
        }
}
```

Gambar 18. Kode Program Interface Layar

SIMPULAN

Penelitian ini menghasilkan sebuah struktur kerangka aplikasi yang selanjutnya dapat dugunakan untuk membangun suatu aplikasi multimedia secara terintegrasi. Pengembangan dilakukan secara modular sehingga memudahkan dalam pembaruan fitur dari aplikasi tanpa harus merubah struktur program. Selanjutnya diharapkan dapat dikembangkan menjadi sebuah framework yang lebih menyeluruh.

DAFTAR PUSTAKA

Friesen, J., 2010. Learn Java for Android Development. Apress Publishing, New York.

Lemay, L., Perkins, C., L., 1996. Teach Yourself Java in 21 days. Sams.net Publishing, Indianapolis.

Mahmoud, O., 2001. Learning Wireless Java. O'reilly & Association, California.

Nugroho, Y., 2009. Handout: Pengenalan Bahasa Java – Inheritance Interface, ITB, Bogor

Zechner, M., Green, R., 2012. Begining Android Games, Second Edition. Apress Media, New York

_http://docs.oracle.com/javase/tutorial/java/concepts/interface.html diakses pada 5 Mei 2015 pukul 10.10wib