BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis

Analisis bertujuan untuk mengidentifikasi permasalahan – permasalahan yang ada pada sistem serta menentukan kebutuhan dari sistem yang dibangun. Analisis tersebut meliputi analisis masalah, analisis kebutuhan data, analisis sistem.

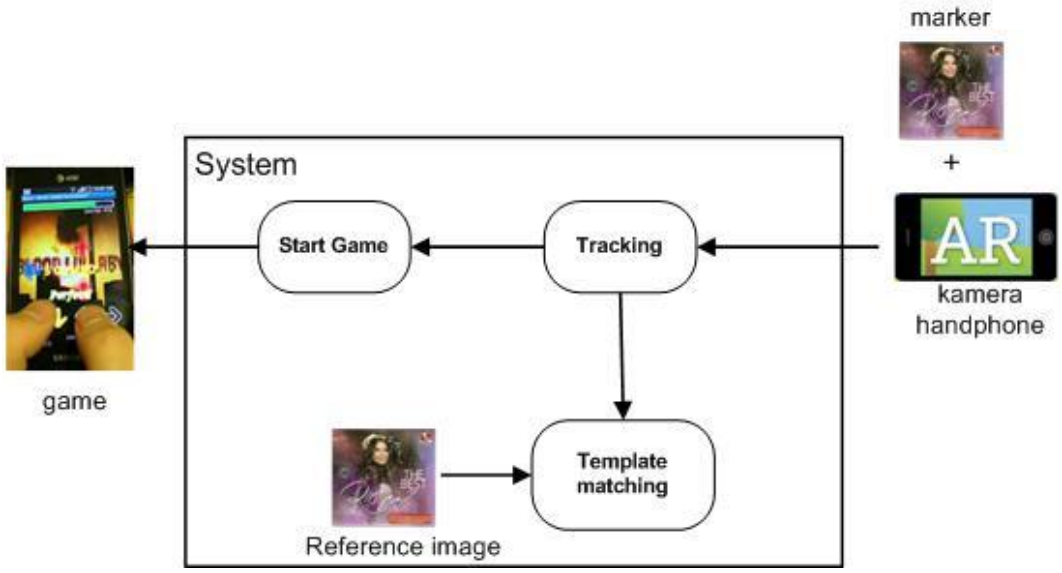
3.1.1 Analisis Masalah

Identifikasi masalah adalah langkah awal dari analisis sistem. Langkah ini diperlukan untuk mengetahui pemasalahan apa saja yang terjadi pada sistem. Oleh karena itu langkah pertama adalah menganalisis masalah yang muncul. Identifikasi masalah yang dimaksud disini adalah bagaimana cara meningkatkan kembali penjualan album fisik (CD/DVD musik). Karena dilihat dari laporan CNN (*Cable News Network*) dan IFPI ***(*** *International Federation of the Phonographic Industry*) terjadinya penurunan tingkat penjualan album fisik sebanyak 5%**.** Sebagai alternatif untuk masalah penjualan album fisik, perlu adanya suatu nilai tambah dalam proses penjualan album fisik tersebut.

3.1.2 Analisis Sistem

Dalam tahap analisis sistem akan ditentukan *requirement* (kebutuhan – kebutuhan) dari sistem yang akan dibangun, mengidentifikasi dan mengevaluasi permasalahan – permasalahan yang terjadi untuk membangun "*game rhythm*

mobile menggunakan *augmented reality*". Adapun sistem yang akan dibangun terlihat pada gambar 3.1 gambaran sistem keseluruhan.



Gambar 3-1 Gambaran sistem yang akan dibangun

3.1.2.1 *Analisis* Augmented reality *terhadap* Game

Seperti penjelasan umum dari bab-bab sebelumnya *augmented reality* pada aplikasi ini digunakan sebagai media interaksi tambahan dimana *augmented reality* berfungsi untuk menjembatani lingkungan virtual dan nyata. Namun secara teknis ada beberapa tahapan, berikut akan dijelaskan mengenai implementasi *augmented reality* pada *game* pada bagian *tracking*, *loading models* (3D *object*), dan interaksi pengguna.

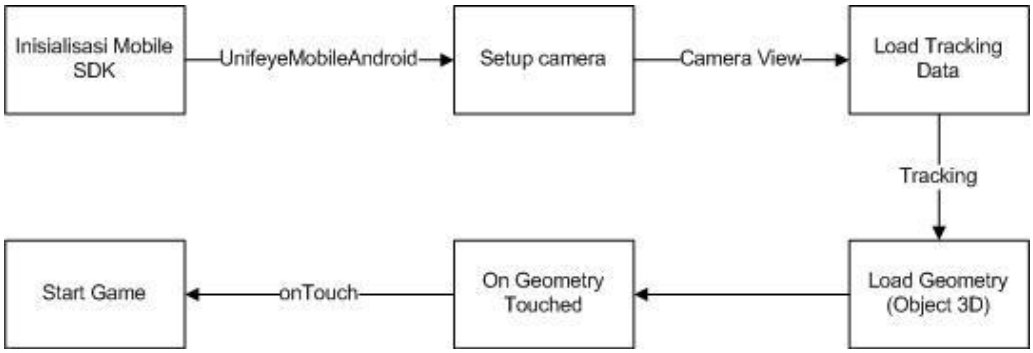


Diagram 3-1 Diagram *augmented reality* terhadap *game*

Penjelasan dari diagram *augmented reality* terhadap *game* yaitu :

a. Inisialisasi *Mobile* SDK

Proses inisialisasi ini terjadi saat *Activity* AdvancedContentActivity dibuat (saat tombol mulai di klik) oleh android, pada *method* onCreate dengan memanggil *method* createMobileSDK. Pada kasus ini tidak terdapat kelas yang langsung dapat di buat dengan "*new*" namun terdapat *static factory class* sebagai gantinya. Objek tersebut akan memberikan *return value* berupa referensi ke *Mobile*SDK dan bertipe IUnifeyeMobileAndroid.

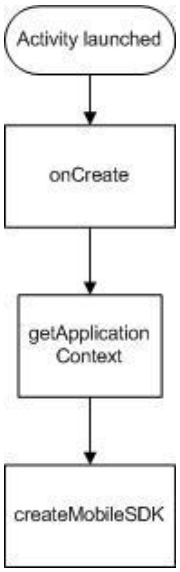


Diagram 3-2 Proses inisialisasi

b. *Setup* kamera

Sebelum dapat mengaktifkan kamera, *content-view* harus sudah ada (pada proses inisialisasi MobileSDK). Untuk mengaktifkan kamera pada MobileSDK sudah disediakan *method* activateCamera.

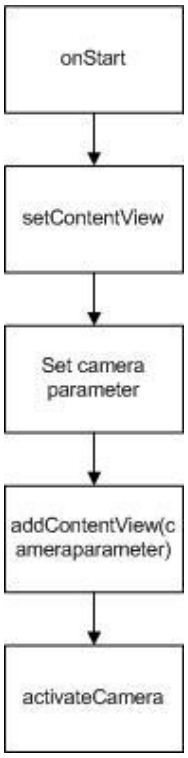


Diagram 3-3 Proses setup kamera

Data yang diperlukan sebagai parameter terdapat pada kelas Configuration. *Method activate* kamera sendiri akan mengembalikan nilai vector dengan dimensi actual dari kamera. Berikut konfigurasi yang direkomendasikan oleh metaio :

1) *deviceID* : bernilai 0 dimana ini menandakan kamera utama yang akan digunakan. Untuk nilai 1 sendiri digunakan untuk *front facing* kamera (kamera depan).
2) *Resolution* : rekomendasi dari metaio adalah 480x320 dimana merupakan *tradeoff* terbaik antara kualitas visual dan performansi.

d. *Load Tracking* Data

Proses ini dilakukan dengan memanggil *method* loadTrackingData. *Method* ini akan mencari berkas *tracking* data pada direktori *assets*, setelah berkas *path* ditemukan kemudian *method* setTrackingData akan dibuat.

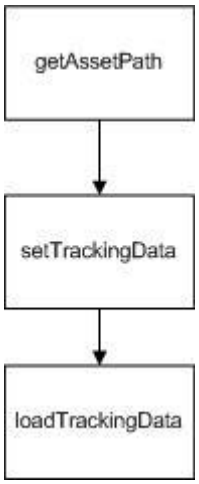


Diagram 3-4 Proses *load tracking* data

e. *Load Geometry* (Objek 3D)

Saat proses load tracking data selesai selanjutnya sistem akan memulai proses tracking, kemudian saat proses tracking berhasil akan melakukan proses memuat objek 3D yang tahap prosesnya seperti pada gambar 3-5.

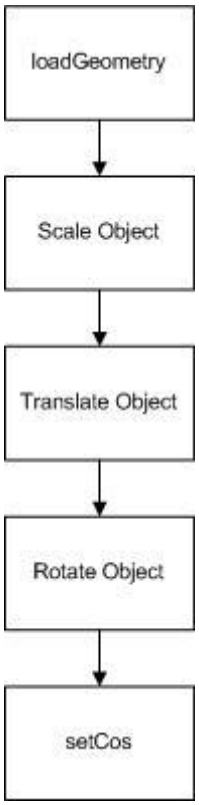


Diagram 3-5 Proses *load geometry*

Pada l*oad geometry* (*box* 3D) terbagi menjadi 2 bagian yaitu : *markeless* dan proses *tracking*. Kedua bagian tersebut dapat dijelaskan sebagai berikut :

Markerless

Markerless merupakan salah satu teknologi *tracking* yang disediakan oleh metaio. *Tracking* jenis ini menggunakan gambar sebagai *marker*. Proses *tracking* ini menggunakan tekstur gambar (berkas jenis gambar) yang disimpan dalam *database* sebagai sumber referensinya, dan membandingkan tekstur yang tertangkap oleh kamera dengan tekstur gambar yang ada di *database marker* (TrackingData_MarkerlessRobust.xml).

Database marker berupa berkas berjenis ".xml", dimana didalam berkas ini terdapat konfigurasi *marker* yang akan digunakan. Pada berkas ini berisi

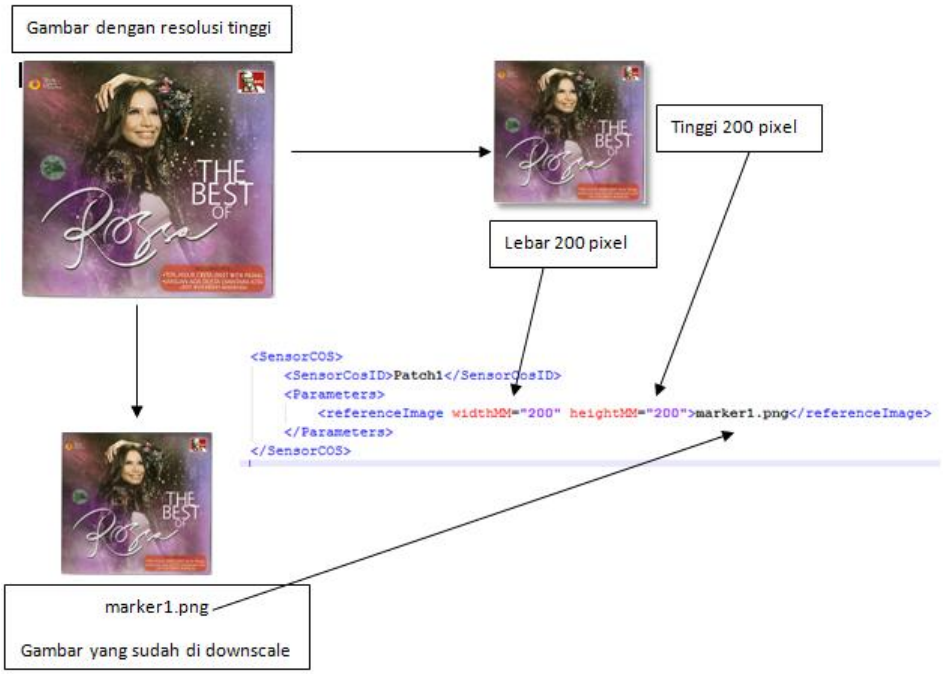
informasi mengenai *reference image* yang akan digunakan pada proses *tracking* (nama berkas, tinggi dan lebar gambar) dan jenis *tracking* yang akan digunakan.

Pada *tracking markeless*, metaio menyediakan dua macam metode yakni *fast* dan *robust*. Aplikasi *game rhythm mobile* menggunakan *augmented reality* menggunakan metode *robust* karena proses *tracking* pada metode ini dapat lebih mudah terjadi meskipun dengan situasi pencahayaan yang berubah – ubah, hal ini dikarenakan pada penelitian ini digunakan *marker* yang tercetak pada bahan yang dapat atau mudah memantulkan cahaya. Selain itu dengan menggunakan metode *robust* hasil *tracking* akan lebih stabil dan metode ini bekerja dengan sangat baik pada target yang bertekstur dibandingkan dengan *fast*.

Untuk proses *tracking* yang stabil, salah satu komponen yang perlu diperhatikan adalah ukuran (tinggi dan lebar) dari berkas gambar dan target dari *marker* yang akan digunakan.

Proses sebelum *marker* dicetak :

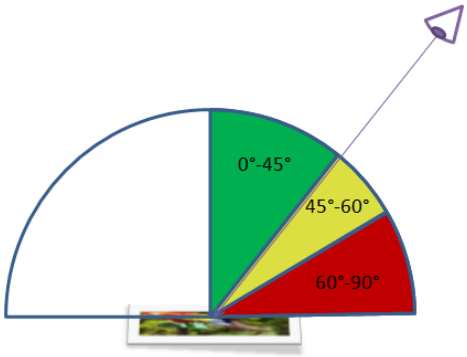
1. Resolusi gambar memiliki resolusi sangat tinggi.
2. *Rescale* gambar untuk memiliki resolusi misalnya 200x200 piksel dimana akan digunakan untuk disimpan di database marker.



Gambar 3-2 Proses pembuatan target *marker*

Tracking dapat bekerja dengan baik jika pengguna memiliki target *marker* yang tercetak pada kertas yang non-reflektif (tidak memantulkan cahaya) dan berada pada permukaan yang datar. Selain itu kondisi pencahayaan harus simpel (terang tetapi cahayanya menyebar dan tidak langsung).

Selain itu pengguna juga harus dapat melihat bagian besar dari gambar target pada tampilan kamera, dan dengan sudut pandang antara 0º - 45º (terlihat pada gambar 3-2 Sudut pandang terhadap *marker*.



Gambar 3-3 Sudut pandang terhadap *marker*

Berikut beberapa daftar target *marker* yang tidak cocok untuk tracking *markerless* :

1. Memiliki detil visual yang sedikit (minim resolusi).
2. Teks
3. Gambar dengan kontras hitam/putih yang tinggi.

Proses tracking

Proses *tracking* yang terjadi adalah dengan pembandingan *texture* gambar pada *database* dengan tampilan pada *marker* teknik ini disebut juga dengan *template matching*. Metode *Template matching* digunakan untuk mendeteksi suatu pattern dengan mengkalkulasikan *normalized cross correlation* antara *reference marker* dan target *marker*, dari hasil kalkukasi normalisasi tersebut akan menghasilkan nilai yang akan menunjukan tingkat kesamaan, dimana angka yang mendekati nilai 1 maka antara *reference marker* dan target *marker* memiliki tingkat kesamaan yang tinggi.

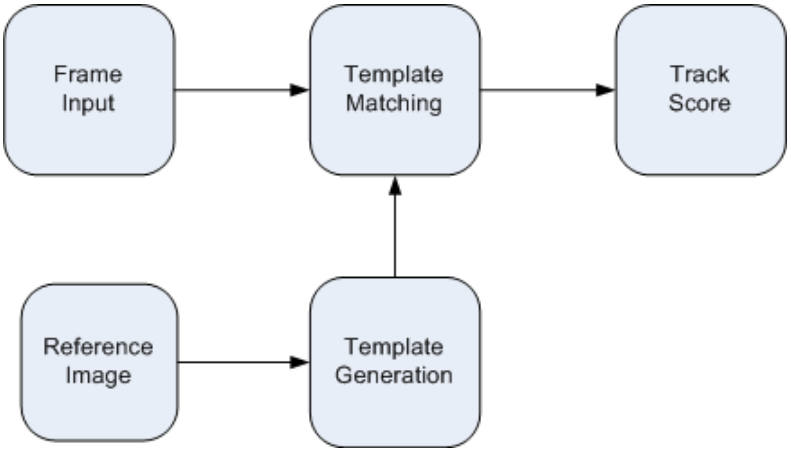
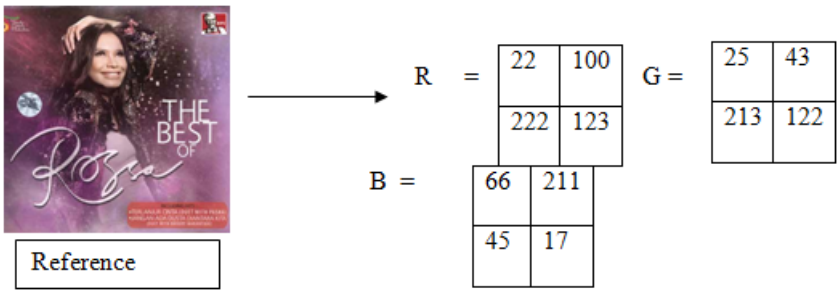


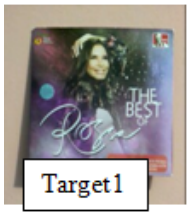
Diagram 3-6 Proses *template matching*

Frame input berasal dari *frame* tampilan kamera. *Reference image* yang berada pada *database marker* dibuat *template*. Lalu metode *template matching* dilakukan dengan membandingkan *pattern* dari *frame* dengan *template pattern* dari *reference image* kemudian memberikan hasil perhitungan *track score*.

Proses didalam *template matching* sendiri sebenarnya yang terjadi adalah mengkalkulasi dengan metode normalisasi *cross correlation* dimana memang metode ini biasa digunakan dalam *image processing*.

Berikut merupakan nilai pixel dari *reference image* dan target *marker* :





R =

64	80
200	100

G =

8	88
188	99

B =

77	199
77	100



R =

23	212
78	65

G =

60	30
103	80

B =

55	200
100	155

Dari nilai-nilai tersebut kemudian akan dicari perhitungan normalisasi *cross correlation*, terlebih dahulu dicari nilai *mean* (rata – rata), dan nilai *standart* deviasi.

Nilai Mean (nilai rata – rata) dari pixel reference image :

$$\overline{R_r} = \frac{22 + 100 + 222 + 123}{4} = 116.75$$

$$\overline{G_r} = \frac{25 + 43 + 213 + 122}{4} = 100.75$$

$$\overline{B_r} = \frac{66 + 211 + 45 + 17}{4} = 84.75$$

Maka dapat dicari *standart* deviasi (σ) untuk *reference image* yaitu :

$$\mathrm{R_r}\ \sigma = \sqrt{\frac{1}{4}((22 - 116.75)^2 + (100 - 116.75)^2 + (222 - 116.75)^2 + (123 - 116.75)^2)}$$

$$= 82.41106$$

$$\mathrm{G_r}\ \sigma = \sqrt{\frac{1}{4}((25 - 100.75)^2 + (43 - 100.75)^2 + (213 - 100.75)^2 + (122 - 100.75)^2)}$$

$$= 85.87733$$

$$\mathrm{B}_r\ \sigma = \sqrt{\frac{1}{4}((66-84.75)^2 + (211-84.75)^2 + (45-84.75)^2 + (17-84.75)^2)}$$

$$= 86.52697$$

Untuk mencari tingkat kesamaan menggunakan normalisasi *cross correlation* antara *reference image* dengan target1 maka terlebih dahulu menghitung nilai *mean, standart* deviasi pada target1 yaitu :

Nilai mean (nilai rata – rata) dari pixel target1 :

$$\overline{R_{t1}} = \frac{64+80+200+100}{4} = 111$$

$$\overline{G_{t1}} = \frac{8+88+188+99}{4} = 73.25$$

$$\overline{B_{t1}} = \frac{77+199+77+100}{4} = 113.25$$

Perhitungan untuk *standart* deviasi pada target1 :

$$\mathrm{R}_{t1}\ \sigma = \sqrt{\frac{1}{4}((64-111)^2 + (80-111)^2 + (100-111)^2 + (100-111)^2)}$$

$$= 61.13373$$

$$\mathrm{G}_{t1}\ \sigma = \sqrt{\frac{1}{4}((8-95.75)^2 + (88-95.75)^2 + (188-95.75)^2 + (99-95.75)^2)}$$

$$= 73.66761$$

$$\mathrm{B}_{t1}\ \sigma = \sqrt{\frac{1}{4}((77-113.25)^2 + (199-113.25)^2 + (77-113.25)^2 + (100-113.25)^2)}$$

$$= 58.18577$$

Selanjutnya mencari tingkat kesamaan antara *reference image* dengan target1 menggunakan normalisasi *cross correlation.*

$$NCC\ R_{r-t1} = \frac{1}{4-1}$$

$$\times \frac{\left((64-111)(22-116.75)+\ (80-111)(100-116.75)+(100-111)(222-116.75)+(100-111)(123-116.75)\right)}{82.41106 \times 61.13373}$$

= 0.944206

$$NCC\ G_{r-t1} = \frac{1}{4-1}$$

$$\times \frac{\left((8-95.75)(25-100.75)+\ (88-95.75)(43-100.75)+(188-95.75)(213-100.75)+(99-95.75)(122-100.75)\right)}{85.87733 \times 73.66761}$$

= 0.923053

$$NCC\ B_{r-t1} = \frac{1}{4-1}$$

$$\times \frac{\left((77-113.25)(66-84.75)+\ (199-113.25)(211-84.75)+(77-113.25)(45-84.75)+(100-113.25)(17-84.75)\right)}{86.52697 \times 58.18577}$$

= 0.9166

Nilai total NCC antara *reference image* dan target1

$$= \frac{0.944206\ + 0.923053\ + 0.9166}{3} = 0.927953$$

Sedangkan untuk mencari tingkat kesamaan menggunakan normalisasi *cross correlation* antara *reference image* dengan target2 maka terlebih dahulu menghitung nilai *mean, standart* deviasi pada target2 yaitu :

$$\overline{R_{t2}} = \frac{23+212+78+65}{4} = 94.5$$

$$\overline{G_{t2}} = \frac{60+30+103+80}{4} = 68.25$$

$$\overline{B_{t2}} = \frac{55+200+100+155}{4} = 127.5$$

Perhitungan untuk *standart* deviasi pada target2 :

$$\mathrm{R}_{t2}\ \sigma = \sqrt{\frac{1}{4}((23-94.5)^2+(212-94.5)^2+(78-94.5)^2+(65-94.5)^2)}$$

= 81.77408

$$G_{t2}\ \sigma = \sqrt{\frac{1}{4}((60-68.25)^2 + (30-68.25)^2 + (103-68.25)^2 + (80-68.25)^2)}$$

$$= 30.96638$$

$$B_{t2}\ \sigma = \sqrt{\frac{1}{4}((55-127.5)^2 + (200-127.5)^2 + (100-127.5)^2 + (155-127.5)^2)}$$

$$= 63.3114$$

Selanjutnya mencari tingkat kesamaan antara *reference image* dengan target1 menggunakan normalisasi *cross correlation.*

$$NCC\ R_{r-t2} = \frac{1}{4-1}$$

$$\times \frac{\left((23-94.5)(22-116.75) + (212-94.5)(100-116.75) + (78-94.5)(222-116.75) + (65-94.5)(123-116.75)\right)}{82.41106 \times 81.77408}$$

$$= 0.142725$$

$$NCC\ G_{r-t2} = \frac{1}{4-1}$$

$$\times \frac{\left((60-68.25)(25-100.75) + (30-68.25)(43-100.75) + (103-68.25)(213-100.75) + (80-68.25)(122-100.75)\right)}{85.87733 \times 30.96638}$$

$$= 0.875446$$

$$NCC\ B_{r-t2} = \frac{1}{4-1}$$

$$\times \frac{\left((55-127.5)(66-84.75) + (200-127.5)(211-84.75) + (100-127.5)(45-84.75) + (155-127.5)(17-84.75)\right)}{86.52697 \times 63.3114}$$

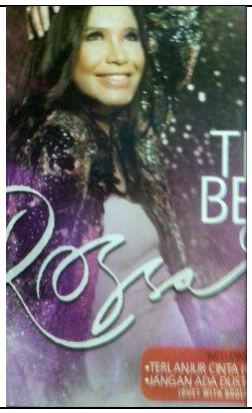
$$= 0.59281$$

Nilai total NCC antara reference image dan target2

$$= \frac{0.142725 + 0.875446 + 0.59281}{3} = 0.53699$$

Dari kedua perhitungan antara *reference image* – target1 dan *reference image* – target2 dapat disimpulkan bahwa *reference image* - target1 memiliki tingkat kesamaan yang tinggi dengan nilai 0.9, dibanding *reference image* - target2 yang hanya memiliki nilai 0.5. Sehingga *object* 3D akan ditampilkan pada target1.

Tabel 3-1 Proses *tracking* menggunakan *Normalized Cross Correlation* (NCC)

<table>
<tr><td colspan="4">Proses tracking dengan menggunakan normalized cross correlation (NCC)</td></tr>
<tr><td>Target marker</td><td>Reference Image</td><td>Hasil template matching dengan NCC</td><td>Tampilan AR sesuai hasil NCC</td></tr>
<tr><td></td><td rowspan="2"></td><td>0.9</td><td></td></tr>
<tr><td></td><td>0.5</td><td>Tampilan AR dengan bentuk box 3D tidak muncul.</td></tr>
</table>

Tabel 3-1 menjelaskan proses tracking yang terjadi, dimana perbandingan antara tampilan pada kamera dengan gambar referensi, selanjutnya *object* 3D akan ditampilkan pada nilai hasil perbandingan yang paling tinggi. Untuk hasil yang baik target *marker* harus terdeteksi oleh kamera lebih dari 70%.

f. *On Geometry Touched* (Interaksi pengguna)

Maksud dari interaksi pengguna disini adalah reaksi dari objek 3D saat *event* sentuh pada layar di perangkat terjadi. Untuk merealisasikan hal tersebut yang perlu dilakukan adalah meng-override *method* onGeometryTouched() yang disediakan oleh kelas ARViewAvtivity. *Method* tersebut secara otomatis akan terpanggil saat pengguna menyentuh layar tepat pada objek 3D.

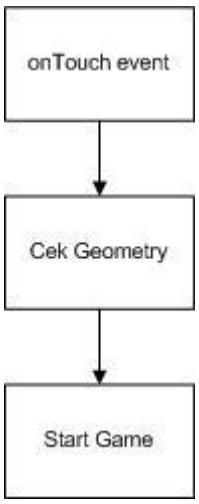


Diagram 3-7 Proses interaksi *augmented reality*

Pada *method* inilah hubungan antara AR dan *game* terjadi, dimana saat objek tersentuh, maka *method* onGeometriTouched() akan mengkonfigurasi properti dari *game*, kemudian memanggil *method* startGameCheck melalui setResult, dimana setResult ini merupakan nilai balikan untuk *method* activityForResult yang berada pada kelas MenuActivity.

3.1.2.2 *Analisis* Game

Game rhythm ini termasuk jenis *game casual* yaitu permainan musik yang berorientasi pada interaksi pengguna dengan *beat*. *Game* ini membutuhkan musik sebagai pengiring permainan. Lagu / musik yang dipilih sesuai dengan album

CD/DVD artis. Pada *game* ini dibutuhkan interaksi pengguna dengan menekan tombol dalam urutan yang ditentukan di layar pada waktu yang tepat, layar menunjukkan tombol mana yang akan ditekan oleh pengguna dan *game* akan memberikan kepada pengguna berupa poin / skor sesuai dengan tingkat akurasi dan sinkronisasi dengan *beat*.

Dalam bermain *game rhythm mobile* menggunakan *augmented reality* ada beberapa aturan bermain (dapat dilihat pada *flowchart game*) yaitu :

1. Pengguna tidak boleh menutupi *marker* saat memilih salah satu lagu dari *box* 3D
2. Apabila selesai memilih salah satu lagu dari *box* 3D maka pengguna akan langsung bermain *game rhythm*.
3. Permain akan selesai memainkan *game rhythm* apabila lagu yang dipilih telah selesai dimainkan atau life bar berwarna merah yang menandakan *game over*
4. Apabila pengguna selesai memainkan *game* atau *game over* maka aplikasi akan menampilkan skor pengguna *game rhythm*.

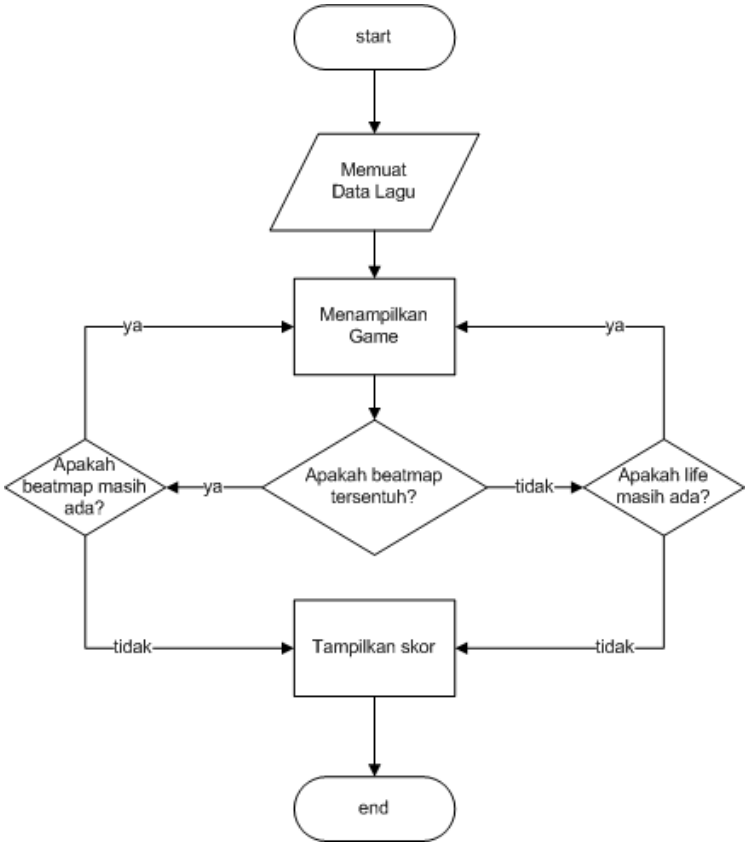


Diagram 3-8 *Rules game*

Game rhtyhm terdiri dari 4 *beat* yang berupa tanda panah kanan, kiri, atas, bawah (→ ← ↑ ↓) yang akan jatuh ke bawah layar dan mencapai kotak berisi tanda panah yang sama. Pengguna harus menekan tombol yang sesuai pada saat yang tepat ketika *beat* masih dalam kotak tanda panah. Setiap kali pengguna menekan tombol dimana *beat* yang turun ke bagian kotak tanda panah maka pengguna akan mendapatkan skor. Skor tersebut akan bervariasi tergantung kecepatan jari menekan tombol, jika penekanan tombol terlalu cepat sebelum *beat* masuk ke dalam kotak tanda panah maka pengguna tidak mendapatkan skor, dan apabila pengguna menekan tombol setelah *beat* melewati kotak tanda panah maka akan dinyatakan kehilangan (*miss*) dan pengguna akan mendapatkan pengurangan skor / poin. Apabila pengguna menekan tombol tepat saat *beat* di

posisi kotak tanda panah maka pengguna mendapatkan skor / poin yaitu *perfect* dimana *beat* sesuai dengan posisi tanda panah yang ada di kotak tanda panah, *great* dimana *beat* mendekati posisi tanda panah yang ada di kotak tanda panah. Untuk penilaian *good* dimana *beat* berada di kotak tanda panah namun tidak tepat atau tidak mendekati posisi tanda panah yang ada di kotak, disini pengguna tidak mendapatkan penambahan ataupun pengurangan skor / poin. Disini pengguna juga bisa mendapatkan nilai *combo* apabila pengguna dapat melakukan permainan dengan nilai *perfect* atau *great* secara terus menerus. Saat penilaian *combo,* pengguna melakukan kesalahan , maka skor / poin akan disesuaikan dengan penilaian dari awal diakumulasi dengan skor / poin sebelumnya.

Game ini akan selesai apabila lagu yang dipilih akan selesai dinyanyikan namun apabila lagu yang dipilih belum selesai dinyanyikan akan dinyatakan *game over*. *Game* ini terdapat *life bar* untuk pengguna, dimana *life bar* ini sebagai indikator apakah *game* selesai dimainkan atau dinyatakan *game over*. *Life bar* akan berkurang apabila pengguna mendapat penilaian *Miss*. Sedangkan saat pengguna mendapatkan penilaian *perfect* atau *great* maka *life bar* pada pengguna akan bertambah sampai batas maksimal. Indikator *life bar* terdapat di atas tampilan *game*, dimana warna hijau jika *life bar* terisi lebih dari 50% dan berwarna kuning jika *life bar* kurang dari 50% dan berwarna merah jika *life bar* akan habis. Jika *life bar* kosong maka permainan akan berakhir dan akan muncul skor.

3.1.3 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional dilakukan untuk mengetahui spesifikasi kebutuhan untuk sistem. Spesifikasi kebutuhan melibatkan analisis perangkat keras/hardware, analisis perangkat lunak/*software*, analisis pengguna/pengguna.

3.1.3.1 *Analisis Perangkat Keras* / Hardware

Analisis kebutuhan nonfungsional dari segi perangkat keras yaitu perangkat keras fitur pendukung aplikasi. Untuk perangkat keras fitur pendukung aplikasi dapat dilihat pada Tabel 3.1 Perangkat Keras Pendukung aplikasi.

Tabel 3-2 Perangkat Keras Pendukung aplikasi

No	Perangkat Keras	Spesifikasi
1	Layar	*Touchscreen*, *accelerometer* sensor, *multitouch*
2	Baterai	Standard
3	*Speaker*	Standard
4	*Processor*	800 Mhz, 32 MB RAM (*without* ROM memory).
5	Kamera	3 MP *Resolution*

3.1.3.2 *Analisis Perangkat Lunak* / Software

Analisis yang telah dilakukan dalam kebutuhan *software* agar *game rhythm* dapat berjalan dengan baik. Adapun perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *game rhythm* adalah sistem operasi android versi 2.3.3 (*Gingerbread*). Android versi 2.3.3 (*Gingerbread*) dipilih karena sistem operasi ini sudah memiliki dukungan *multitouch* yang telah stabil dibandingkan dengan versi android dibawahnya (ada beberapa pernyataan di internet yang menyatakan versi android di bawah 2.3.3 masih terdapat kekurangan pada fitur *multitouch*-nya), fitur *multitouch* ini nantinya akan digunakan pada aplikasi dengan *beat-beat* tertentu (*hold beat* yang disertai dengan *tap beat*, atau sebaliknya). Selain itu seperti yang disebutkan pada situs *developer* android, bahwa pada versi *Gingerbread* penanganan *event touch* lebih cepat dan efisien sehingga berdampak pada aplikasi menjadi lebih responsif.

Beberapa perangkat lunak pendukung yang harus di *install* yaitu :

1. *Windows* sebagai sistem operasi
2. *Eclipse*
3. *Android SDK*
4. *Metaio mobile SDK*

3.1.3.3 *Analisis Pengguna*

Suatu aplikasi dapat berjalan optimal apabila pengguna memiliki kemampuan untuk menjalankan aplikasi yang bersangkutan. Untuk memainkan *game* setidaknya pengguna harus memiliki:

1. Usia : minimal 12 tahun
2. Pendidikan : minimal SD, atau sederajat
3. Ketrampilan : mampu menggunakan petunjuk yang ada di aplikasi, dapat menjalankan aplikasi pada perangkat android versi 2.3.3.
4. Pengalaman : pernah menggunakan *handphone touchscreen* dengan sistem operasi android dan mengetahui tentang *augmented reality*

3.1.4 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses yang akan diterapkan dalam sistem yang akan dibangun sesuai dengan kebutuhan proses bisnis.

Analisis kebutuhan fungsional dimodelkan dengan menggunakan *Unified Modeling Language* (UML). Tahap – tahap pemodelan dalam analisis menggunakan UML antara lain *Usecase diagram*, *Usecase scenario*, *activity diagram*, *sequence diagram* dan *class diagram*.

3.1.4.1 Use Case Diagram

Analisis *game rhythm* terdiri dari analisis prosedur mulai *game*, prosedur pengaturan, prosedur petunjuk. Adapun proses dari setiap prosedur yang terdapat pada *game rhythm* digambarkan dengan menggunakan *usecase* diagram yang

terlihat pada diagram 3.1 *Usecase* Diagram *Game rhythm*. Pada *usecase* ini terdapat dua actor yaitu pengguna dan *kamera.*

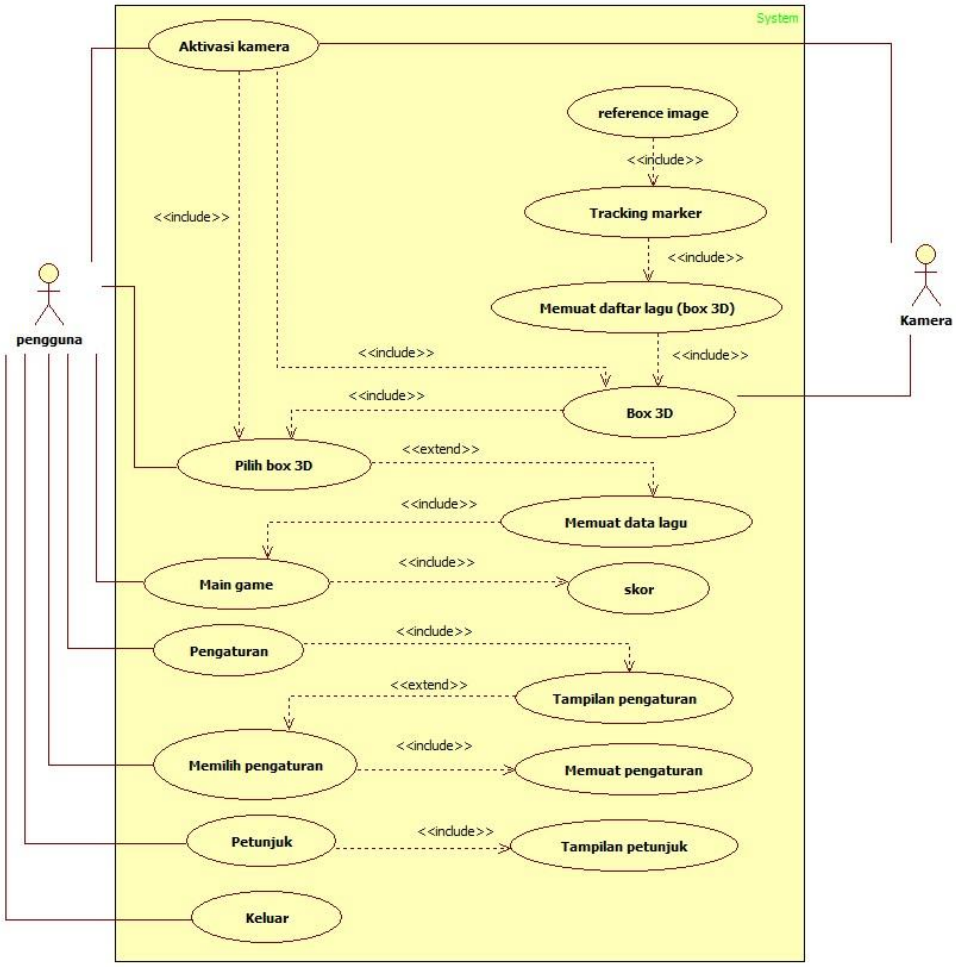


Diagram 3-9 *Usecase Game rhythm*

Setiap proses yang terdapat pada u*secase* diagram *game rhythm* (diagram 3-4 *Usecase game rhythm*) dapat dijelaskan lebih detail dengan menggunakan activity diagram yang terdiri dari *activity diagram* mulai *game, activity diagram* pengaturan, *activity diagram* petunjuk.

1. Prosedur Mulai *Game*

Prosedure mulai menjelaskan tentang tahapan – tahapan yang dilakukan pengguna untuk bisa bermain *game rhythm*.adapun prosedur mulai pada *game rhythm* yaitu :

a. Sistem menampilkan menu utama dari mulai *game*
b. Pengguna memilih menu mulai dan memposisikan *cover* CD/DVD musik tepat di depan kamera
c. Sistem akan melakukan *tracking marker* untuk memuat daftar lagu yang terdapat pada CD/DVD musik yang nantinya akan digunakan di saat *game rhythm* akan dimainkan.
d. Pengguna dapat memilih lagu dari daftar lagu yang ada sesuai dengan keinginan dari pengguna.
e. Sistem memuat lagu yang telah dipilih oleh pengguna dan pengguna dapat memulai *game rhythm* dengan latar lagu yang telah dipilih. Setelah pengguna selesai memainkan *game rhythm* maka pengguna dapat melihat hasil dari permainan/skor dari *game* dan lagu yang telah dipilih.

2. Prosedur Pengaturan

Pada prosedur pengaturan terdapat menu pengaturan kecepatan dan pengaturan kesulitan, dimana kedua pengaturan ini digunakan untuk menaikkan atau menurunkan tingkat kecepatan *beat* atau tingkat kesulitan dari *beat* pada saat *game rhythm* dimainkan oleh pengguna. Adapun prosedur dari pengaturan kecepatan yaitu :

a. Pengguna memilih menu pengaturan kecepatan
b. Sistem menampilkan menu pengaturan kecepatan yaitu 1.00, 1.25, 1.50, 1.75, 2.00 dst. Disini pengguna dapat memilih salah satu tingkat kecepatan yang diinginkan.
c. Sistem mengatur tingkat kecepatan *beat* dari *game rhythm* yang telah dipilih oleh pengguna.

Sedangkan prosedur dari pengaturan kesulitan yaitu :

a. Pengguna memilih menu pengaturan kesulitan
b. Sistem menampilkan menu pengaturan kesulitan yaitu *easy*, *medium*, *hard* dan pengguna dapat memilih salah satu tingkat kesulitan yang diinginkan.
c. Sistem mengatur tingkat kesulitan *beat* dari *game rhythm* yang telah dipilih oleh pengguna.

3. Prosedur Petunjuk

Prosedur petunjuk hanya menampilkan bagaimana penggunaan atau cara bermain *game rhythm* , cara pengaturan *game rhythm* baik pengaturan kecepatan maupun pengaturan kesulitan.

4. Prosedur Keluar

Prosedur keluar menampilkan notifikasi apakah pengguna akan keluar dari aplikasi atau tidak.

3.1.4.2 Usecase *Skenario*

Usecase Skenario merupakan deskripsi urutan langkah – langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor. Berdasarkan use case diagram maka use case scenario untuk aplikasi *game rhythm* terdiri dari skenario mulai *game*, skenario pengaturan kecepatan, scenario pengaturan kesulitan, scenario petunjuk.

1. Usecase Skenario Mulai *Game*

Skenario mulai menjelaskan cara memulai *game* dari proses mulai sampai muncul tampilan kamera.

Tabel 3-3 *Usecase* Skenario Mulai

IDENTIFIKASI	
Nomor	UCS-01
Nama	*Usecase* mulai
Deskripsi	Menjelaskan tentang proses mulai
Aktor	Pengguna
Kondisi Awal	Aktor telah menjalankan *game rhythm* di *handphone*
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol mulai	2. Memproses *request* dari aktor
	3. Kamera aktif
Kondisi Akhir : Tampilan kamera akan muncul	

2. *Usecase* Skenario tampilan *box* 3D

Skenario ini menjelaskan *tracking marker* tampilan *box* 3D yang berisi daftar lagu yang akan oleh pengguna.

Tabel 3-4 *Usecase* Skenario tampilan *box* 3D

IDENTIFIKASI	
Nomor	UCS-01

Nama	*Usecase* tampilan *box* 3D
Deskripsi	Menjelaskan tentang proses menampilkan *box* 3D
Aktor	Kamera
Kondisi Awal	Aktor telah menjalankan *game rhythm* di *handphone*
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Kamera aktif	2. Proses *tracking* dengan *reference image*
	3. Memuat daftar lagu (*box*3D)
4. Menampilkan *box* 3D	
Kondisi Akhir : Tampilan *box* 3D akan muncul	

3. *Usecase* Skenario Pilih Lagu

Skenario ini menjelaskan pilih lagu yang diinginkan aktor sebelum memainkan *game*.

Tabel 3-5 *Usecase* Skenario Pilih lagu

IDENTIFIKASI	
Nomor	UCS-02
Nama	*Usecase* pilih lagu (*box* 3D)
Deskripsi	Menjelaskan tentang proses pilih lagu
Aktor	Pengguna
Kondisi Awal	Tampilan *box* 3D sudah dimuat
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Memilih *box* 3D	2. Memuat data lagu yang dipilih
	3. Menampilkan tampilan *game*
Kondisi Akhir : Tampilan *game* muncul	

4. *Usecase* Skenario Main *Game*

Skenario ini menjelaskan main *game* dengan memuat data lagu yang dipilih oleh pengguna.

Tabel 3-6 *Usecase* Skenario Main game

IDENTIFIKASI	
Nomor	UCS-03
Nama	*Usecase* main *game*
Deskripsi	Menjelaskan tentang proses main *game*.
Aktor	Pengguna
Kondisi Awal	Tampilan *game* muncul
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
	1. Memuat data lagu (*beat* dan musik) yang dipilih
	2. Menampilkan *game* (*beat*)
3. Aktor memainkan *game*	4. Sistem mencatat poin *game*
Kondisi Akhir : Tampilan permainan muncul dan permainan dimulai	

5. Use Case Skenario Pengaturan

Skenario ini menjelaskan cara mengatur tingkat kecepatan dan tingkat kesulitan yang terdapat pada *game*.

Tabel 3-7 *Usecase* skenario pengaturan

IDENTIFIKASI	
Nomor	UCS-05
Nama	*Usecase* pengaturan
Deskripsi	Menjelaskan tentang proses pengaturan pada *game*
Aktor	Pengguna
Kondisi Awal	Aktor telah menjalankan *game rhythm* di *handphone*
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol pengaturan	2. Menerima *request* dari aktor
	3. Menampilkan menu tingkat kesulitan dan tingkat kecepatan
Kondisi Akhir : Menampilkan tampilan pengaturan	

a. *Usecase* Skenario Pengaturan Kecepatan

Skenario ini menjelaskan cara mengatur tingkat kecepatan *beat* yang diinginkan aktor pada saat memainkan *game*.

Tabel 3-8 *Usecase* Skenario Pengaturan Kecepatan

IDENTIFIKASI	
Nomor	UCS-06
Nama	*Usecase* pengaturan kecepatan
Deskripsi	Menjelaskan tentang proses pengaturan kecepatan pada *game*
Aktor	Pengguna
Kondisi Awal	Aktor telah menjalankan *game rhythm* di *handphone*
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol pengaturan	2. Menerima *request* dari aktor
	3. Menampilkan menu pengaturan
4. Aktor memilih menu pengaturan kecepatan	5. menampilkan tingkat pengaturan kecepatan.
6. Aktor memilih tingkat kecepatan	7. Mengatur tingkat kecepatan *beat* pada *game*
Kondisi Akhir : Kembali ke menu pengaturan dan tingkat kecepatan di perbaharui	

b. *Usecase* Skenario Pengaturan Kesulitan

Skenario ini menjelaskan cara mengatur tingkat kesulitan *beat* yang diinginkan aktor atau banyaknya *beat* yang muncul pada saat memainkan *game*.

Tabel 3-9 *Usecase* Skenario Pengaturan Kesulitan

IDENTIFIKASI	
Nomor	UCS-07
Nama	*Usecase* pengaturan kesulitan
Deskripsi	Menjelaskan tentang proses pengaturan kesulitan pada *game*
Aktor	Pengguna
Kondisi Awal	Aktor telah menjalankan aplikasi di *handphone*
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol pengaturan	2. Menerima *request* dari aktor
	3. Menampilkan menu pengaturan
4. Aktor memilih menu pengaturan	5. menampilkan tingkat pengaturan

kesulitan	kesulitan.
6. Aktor memilih tingkat kesulitan	7. Mengatur tingkat kesulitan *beat* pada *game*
Kondisi Akhir : Kembali ke menu pengaturan dan tingkat kesulitan di perbaharui	

6. *Usecase* Skenario Petunjuk

Skenario ini menjelaskan bagaimana cara penggunaan *game rhythm* dari cara memainkan *game*, mengatur *game*, dimana agar aktor mengerti fungsi dari setiap menu yang ada pada *game*.

Tabel 3-10 *Usecase* Skenario Petunjuk

IDENTIFIKASI	
Nomor	UCS-08
Nama	*Usecase* petunjuk
Deskripsi	Menjelaskan tentang petunjuk pada *game*
Aktor	Pengguna
Kondisi Awal	Aktor telah menjalankan aplikasi di *handphone*
SKENARIO	
Aksi Aktor	**Reaksi Sistem**
1. Aktor menekan tombol petunjuk	2. Menampilkan informasi penggunaan *game*
Kondisi Akhir : Menampilkan halaman petunjuk	

7. *Usecase* skenario Keluar

Skenario ini merupakan proses yang dilakukan oleh aktor untuk keluar dari sistem.

Tabel 3-11 *Usecase* skenario Keluar

IDENTIFIKASI	
Nomor	UCS-08
Nama	*Usecase* keluar
Deskripsi	Menjelaskan proses keluar dari aplikasi
Aktor	Pengguna
Kondisi Awal	Aktor telah menjalankan aplikasi di *handphone*
SKENARIO	

Aksi Aktor	Reaksi Sistem
1. Aktor menekan tombol petunjuk	2. Menampilkan notifikasi keluar
3. Aktor memilih pilihan ya atau tidak	4. Sistem memproses pilihan
Kondisi Akhir : Menampilkan notifikasi keluar	

3.1.4.3 Activity Diagram

Activity diagram merupakan diagram yang memodelkan aliran kerja dari urutan aktifitas suatu proses yang mengacu pada *use case* diagram. *Activity* diagram yang terdapat pada *game rhythm* yaitu activity diagram mulai *game*, *activity* diagram pengaturan dan *activity* diagram petunjuk. Adapaun penjelasan dari masing – masing *activity* diagram tersebut yaitu :

1. *Activity* Diagram Mulai *game*

Activity diagram mulai *game* menjelaskan alur atau cara memulai *game* yang digunakan pada *handphone*/ *smartphone* dengan OS android 2.3.3 dan menggunakan *marker cover* CD/DVD musik untuk memilih lagu yang nantinya akan diset sebagai latar lagu untuk permainan *game rhythm*.

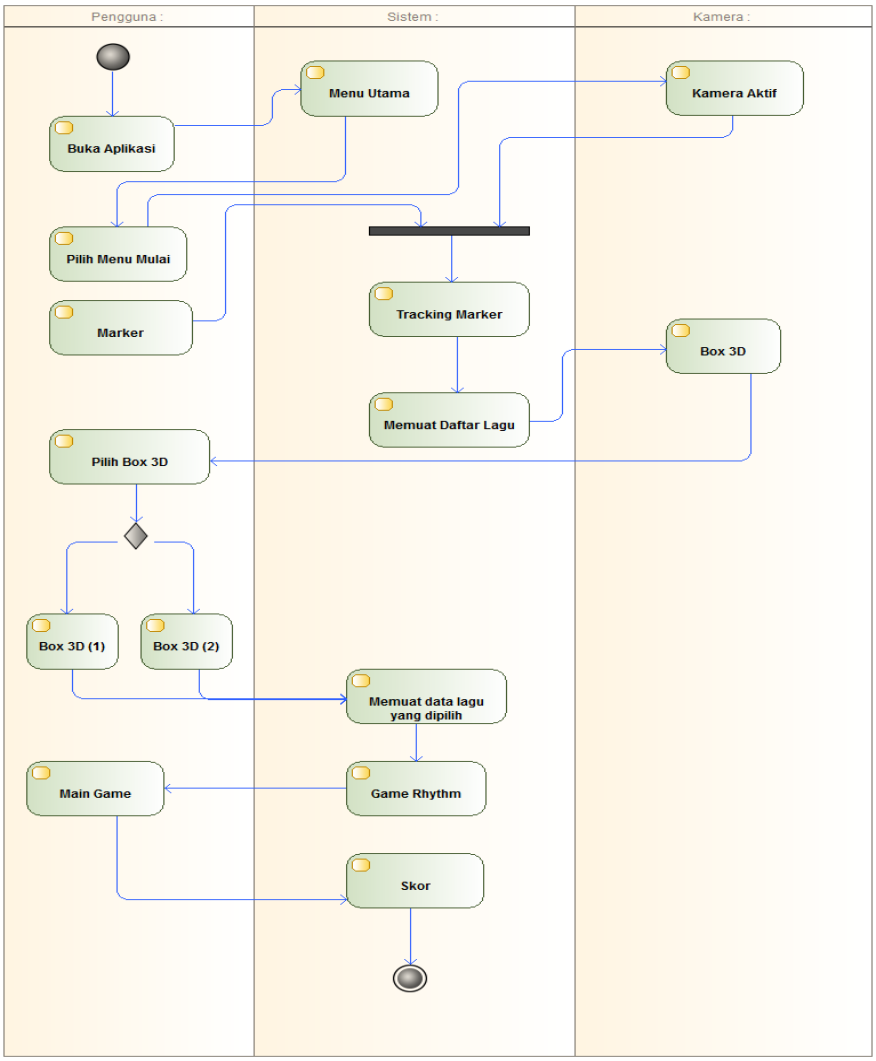


Diagram 3-10 *Activity* Diagram Mulai *Game*

2. *Activity* diagram pengaturan

Activity diagram pengaturan akan menjelaskan pengaturan untuk mengatur kecepatan dan mengatur kesulitan *beat* pada *game rhythm*. Untuk mengatur kecepatan *beat* pada *game rhythm* dapat dilihat pada diagram 3.6 *Activity* diagram pengaturan.

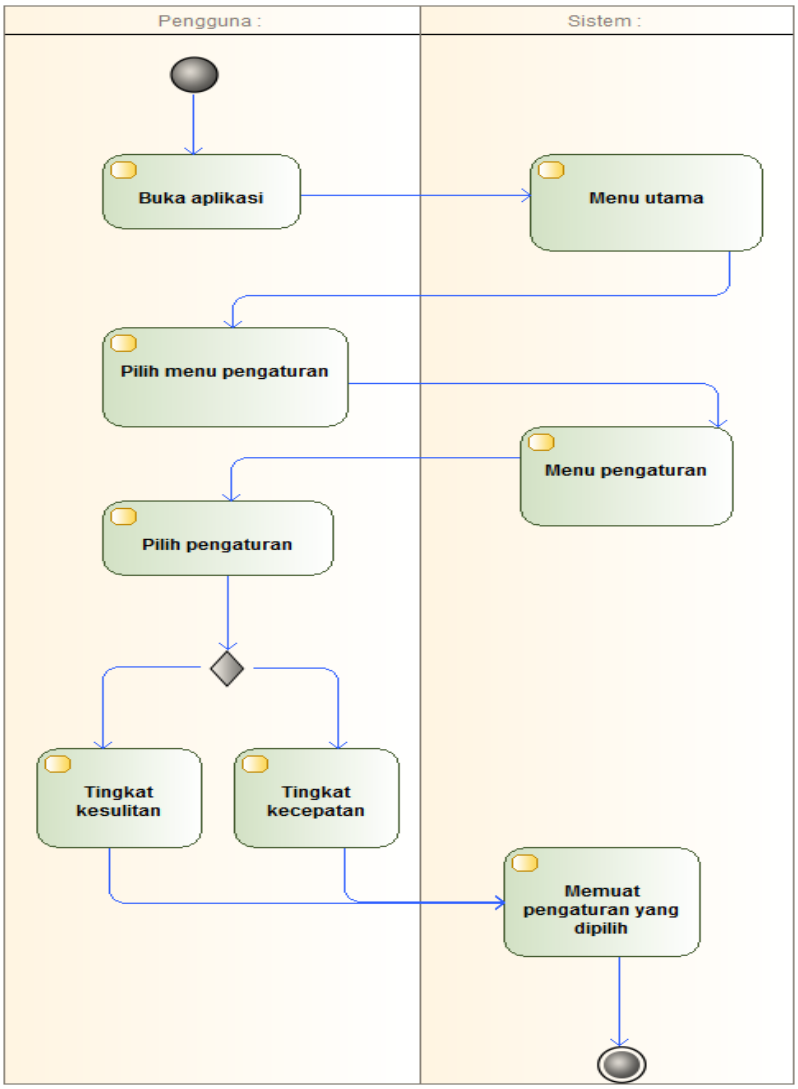


Diagram 3-11 *Activity* Diagram Menu Pengaturan

3. *Activity* diagram petunjuk

Activity diagram help akan menjelaskan aliran kerja atau workflow dari *game rhythm* dimana help berisi penggunaan dari menu – menu yang ada pada *game rhythm* seperti mulai *game*, pengaturan, petunjuk.

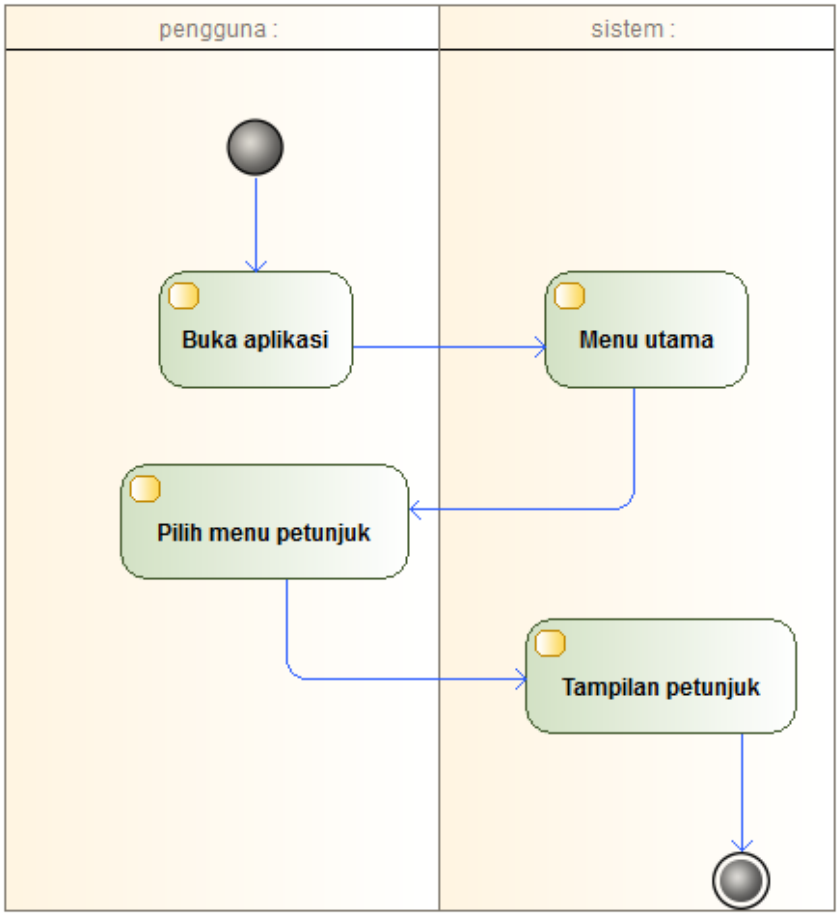


Diagram 3-12 *Activity* Diagram Petunjuk

4. *Activity* Diagram Keluar

Activity diagram keluar akan menjelaskan aliran kerja atau *workflow* dari *game rhythm* dimana keluar akan memberikan notifikasi.

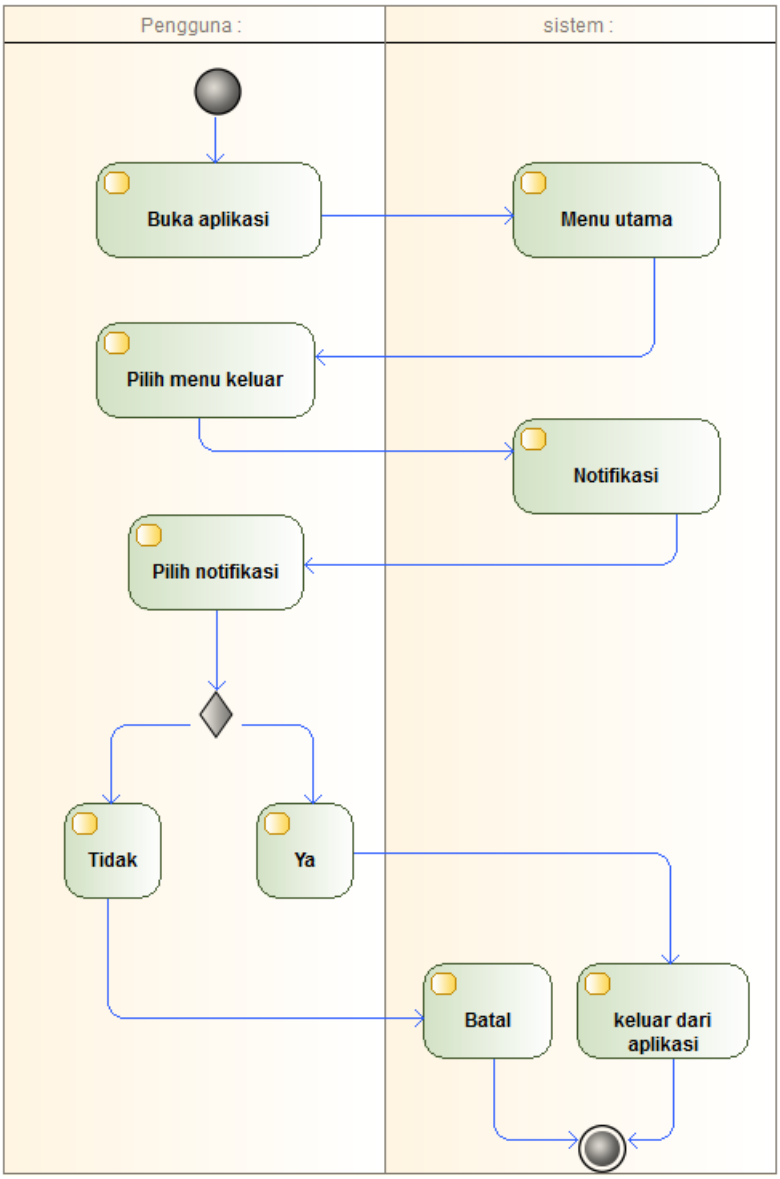


Diagram 3-13 *Activity* Diagram Keluar

3.1.4.4 Sequence Diagram

Sequence diagram menjelaskan gambaran interaksi antar objek dalam urutan waktu. Interaksi ini berupa pengiriman serangkaian data antar objek-objek yang

saling berinteraksi. *Sequence diagram* pada aplikasi ini terdiri dari *Sequence diagram* menu utama, mulai *game* , pengaturan dan instruksi.

3.1.4.4.1 Sequence *Diagram Mulai* Game

Sequence diagram mulai *game* menunjukkan rangkaian interaksi, pesan yang dikirim antar objek pada proses dimana *game* dimulai.

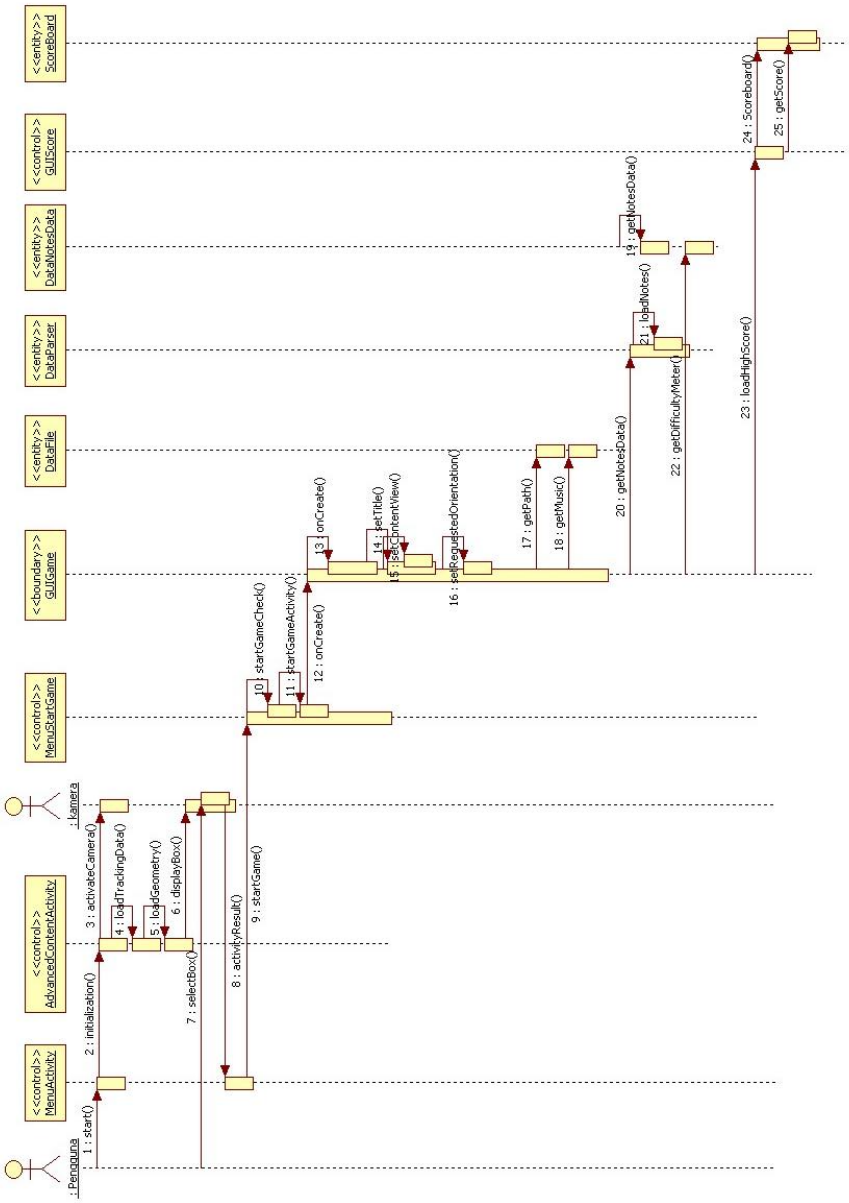


Diagram 3-14 *Sequence* Diagram Mulai *Game*

3.1.4.4.2 *Sequence Diagram Pengaturan*

Sequence diagram pengaturan menunjukkan rangkaian interaksi, pesan yang dikirim antar objek pada proses pengaturan *game*.

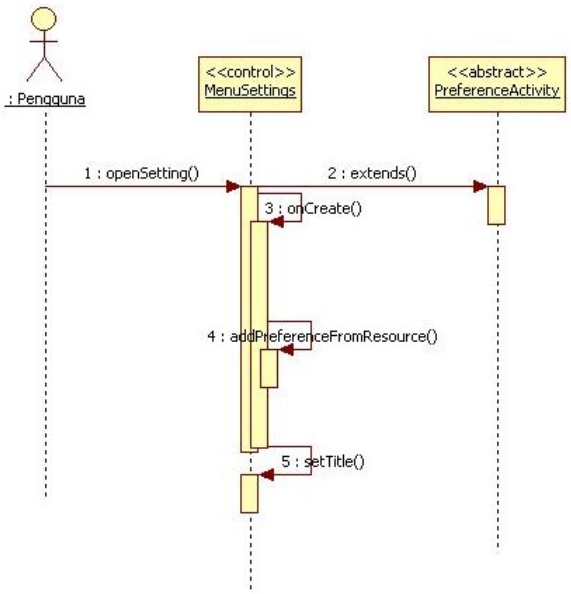


Diagram 3-15 *Sequence* Diagram Pengaturan

3.1.4.4.3 *Sequence Diagram Petunjuk*

Sequence diagram petunjuk menunjukkan rangkaian interaksi, pesan yang dikirim antar objek pada proses dimana halaman petunjuk ditampilkan.

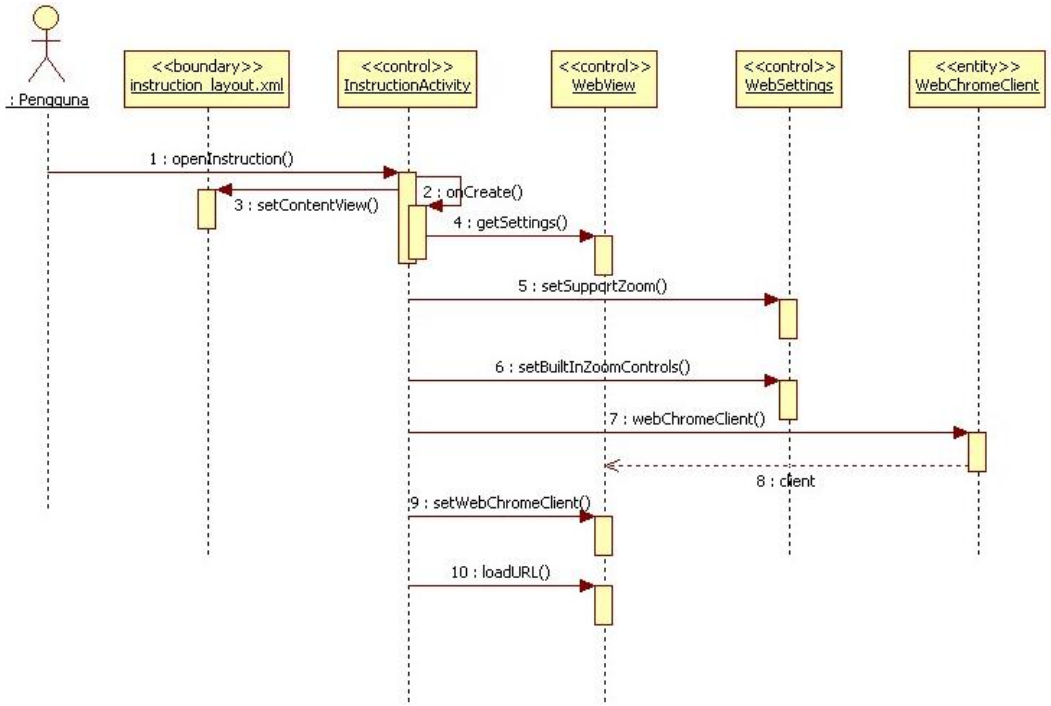


Diagram 3-16 *Sequence* Diagram Petunjuk

3.1.4.4.4 Sequence *Diagram Keluar*

Sequence diagram keluar menunjukkan rangkaian interaksi, pesan yang dikirim antar objek pada proses dimana notifikasi akan ditampilkan.

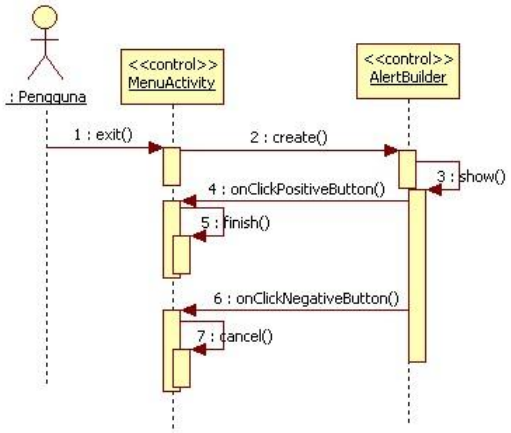


Diagram 3-17 *Sequence* Diagram Keluar

3.1.4.5 Deskripsi Class Diagram

Diagram kelas adalah diagram yang digunakan untuk menampilkan beberapa kelas yang ada dalam perangkat lunak yang sedang dikembangkan. Diagram kelas menggambarkan struktur dan deskripsi kelas beserta hubungan kelas yang satu dengan yang lain.

3.1.4.5.1 Class Diagram *Mulai Game*

Kelas ini dipanggil melalui *method* startGameCheck() pada saat object 3D pada AR di klik, lalu sebelum memulai *game* kelas ini akan melakukan beberapa pengecekan terlebih dahulu diantaranya:

a Mengecek apakah media penyimpanan sudah terpasang

b Menampung (*buffer*) pengaturan dari lokasi berkas *game* yang akan dimuat

c Mengecek apakah lokasi *beat* valid atau tidak

Saat semua pengecekan sudah dilalui maka *method* startGameCheck() akan memanggil *method* startGame() untuk memulai *game*.

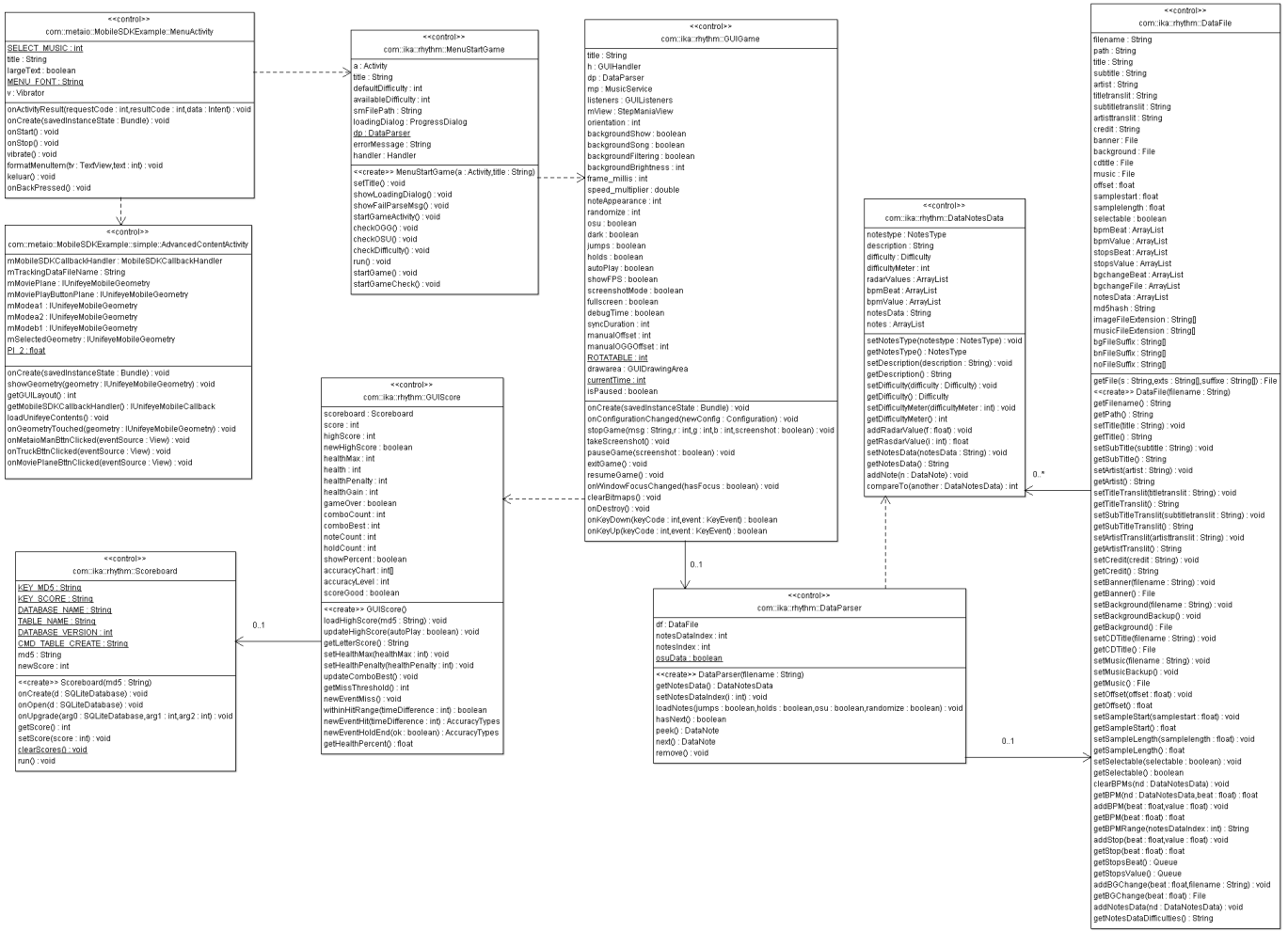
Pada *method* startGame() akan menampilkan loadingdialog yang menandakan resource *game* sedang di muat, setelah itu *game* akan dimulai. Kelas MenuStartGame ini memiliki relasi berupa dependency terhadap kelas GUIGame, yang mana kelas GUIGame tersebut digunakan untuk menampilkan tampilan *game*.

Kelas GUIGame memiliki relasi asosiasi (*bidirectional*) terhadap kelas DataParser, dimana kelas DataParser akan digunakan untuk mengambil data-data dari berkas stepmania, data tersebut berupa judul, tingkat kesulitan, *beat*, dll. DataParser sendiri juga memiliki relasi asosiasi terhadap DataFile, dimana datafile bertindak sebagai peng-ekstrak data dari berkas stepmania (.SM), melalui kelas DataFile-lah DataParser mendapatkan data-data berupa judul, tingkat kesulitan, *beat*, dll.

Selain itu terdapat juga kelas DataNote, Difficulty, dan NotesType, yang masing-masing kelas tersebut digunakan untung mengekstrak data pada berkas stepmania.

Berikut gambar dari class diagram mulai *game* :

Diagram 3-18 *Class* Diagram Mulai *Game*



3.1.4.5.2 Class *Diagram Pengaturan* Game

Berikut gambar dari class diagram pengaturan *game* :

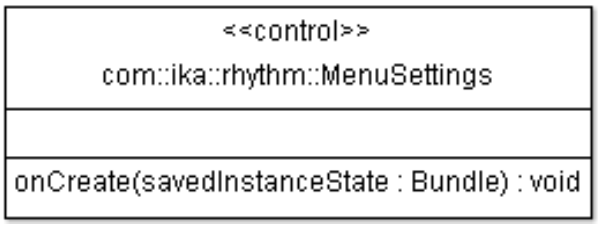


Diagram 3-19 *Class* Diagram Pengaturan *Game*

Kelas ini merupakan *activity* untuk merubah konfigurasi *game*, berupa tingkat kesulitan dan kecepatan *beat*. Kelas ini adalah *subclass* dari kelas PreferenceActivity yang merupakan bawaan dari android. Resource dari pengaturan yang dapat diubah berasal dari *file* settings.xml yang berada pada direktori res pada android.

3.1.4.5.3 *Class Diagram Petunjuk* Game

Kelas ini digunakan untuk memulai *activity* yang menampilkan halaman instruksi/help, halaman instruksi sendiri merupakan webview, yang mana berupa tampilan halaman web. Halaman yang ditampilkan adalah index.html yang terdapat pada direktori *assets* android.

Berikut gambar dari class diagram petunjuk *game* :

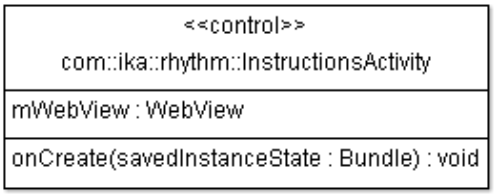


Diagram 3-20 *Class* Diagram Petunjuk *Game*

3.1.4.5.4 Class *Diagram Keluar*

Berikut merupakan gambar dari *class* diagram keluar, yang berfungsi sebagai menu notifikasi untuk keluar. Method utama pada class diagram keluar ini adalah method "keluar()", dimana nantinya akan muncul tampilan notifikasi saat menu keluar dipilih atau pengguna menekan tombol *back*.

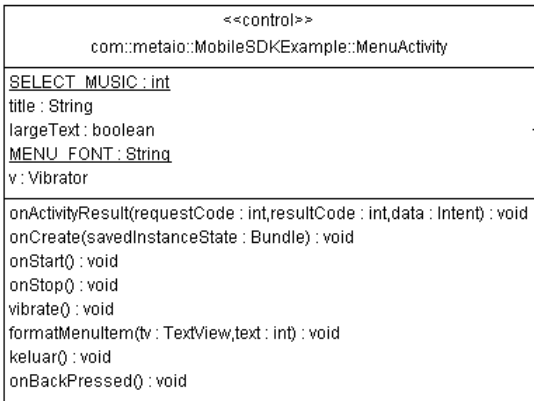


Diagram 3-21 *Class* Diagram Keluar

Tabel 3-12 Deskripsi *class* diagram

Class	Jenis Class	Deskripsi
MenuActivity	Control	Class yang berisikan proses-proses untuk tampilan menu utama
InstructionActivity	Control	Class yang berisikan proses-proses untuk tampilan petunjuk
MenuSettings	Control	Class yang berisikan proses-proses untuk merubah pengaturan *game*
AdvancedContentActivity	Control	Class yang mengatur interaksi AR
ARViewActivity	Control	Abstract Class yang digunakan oleh Class AdvancedContent*Activity*, yang berisi *method* untuk AR
MenuStartGame	Control	Class yang digunakan untuk proses validasi sebelum *game* dimulai
DataParser	Entity	Class yang digunakan untuk mem-buffer informasi note/*beat*
DataFile	Entity	Class yang digunakan untuk mem-buffer informasi dari file sm
DataParserSM	Entity	Class yang berisi fungsi-fungsi untuk mem-buffer file sm
GUIGame	Control	Class yang berisikan fungsi-fungsi untuk memulai *game*
GUIDrawingArea	Control	Class yang berisikan fungsi-fungsi untuk menampilkan tampilan *game*
GUIFallingObject	Control	Class yang berisikan fungsi-fungsi untuk menampilkan object note/*beat*
GUIFallingArrow	Control	Class yang berisikan fungsi-fungsi untuk menampilkan note/*beat*
DataNotesData	Entity	Class yang berfungsi untuk mem-buffer informasi data note/*beat*
DataNote	Entity	Class yang berfungsi untuk mem-buffer informasi note/*beat*
GUIListeners	Control	Class yang berfungsi untuk menunggu inputan pengguna
GUIHandler	Control	Class yang berfungsi untuk mengolah event yang terjadi
GUIFallingHold	Control	Class yang berfungsi proses-proses saat jenis note/*beat* yang keluar adalah yang panjang

GUIHandlerTap	Control	Class yang berfungsi mengatur event saat area kotak panah ditekan
GUITapArrow	Control	Class yang berfungsi untuk mendeteksi apakah note/*beat* ditekan
GUIScoreDisplay	Control	Class yang berisi proses-proses untuk menampilkan skor di akhir permainan
GUIScore	Control	Class yang berisi fungsi-fungsi untuk menghitung skor
Scoreboard	Control	Class yang berisi fungsi-fungsi untuk melakukan perubahan skor

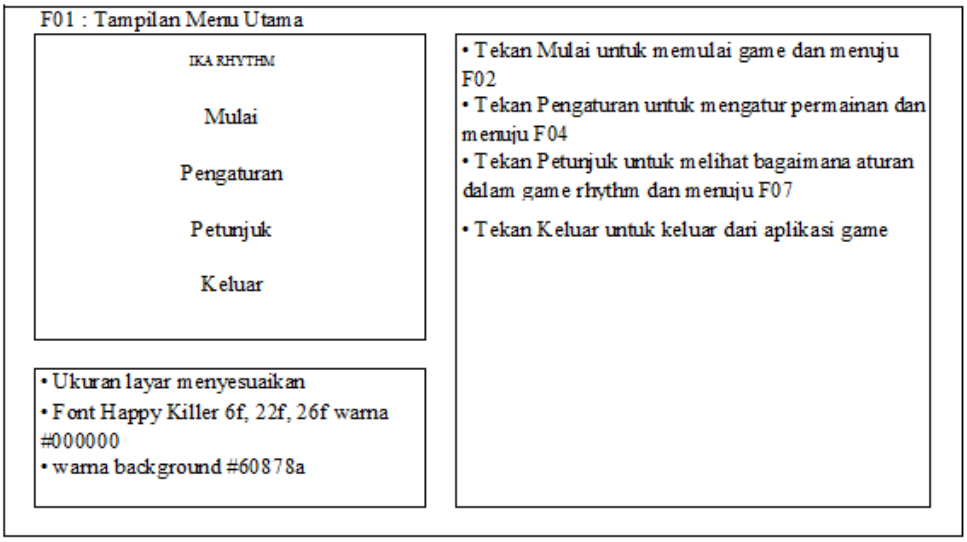
3.2 Perancangan Antarmuka

Interface atau antar muka merupakan tampilan dari suatu program aplikasi yang berperan sebagai media komunikasi yang digunakan sebagai sarana berdialog antara program dengan pengguna. Sistem yang akan dibangun diharapkan menyediakan *interface* yang mudah dipahami dan digunakan oleh pengguna.

Perancangan antarmuka mendeskripsikan rencana tampilan dari setiap form yang akan digunakan pada aplikasi *game rhythm*. Perancangan antarmuka pada *game rhythm* terdiri dari perancangan menu utama, perancangan mulai, perancangan pengaturan dan perancangan petunjuk.

3.2.1 Perancangan Menu Utama

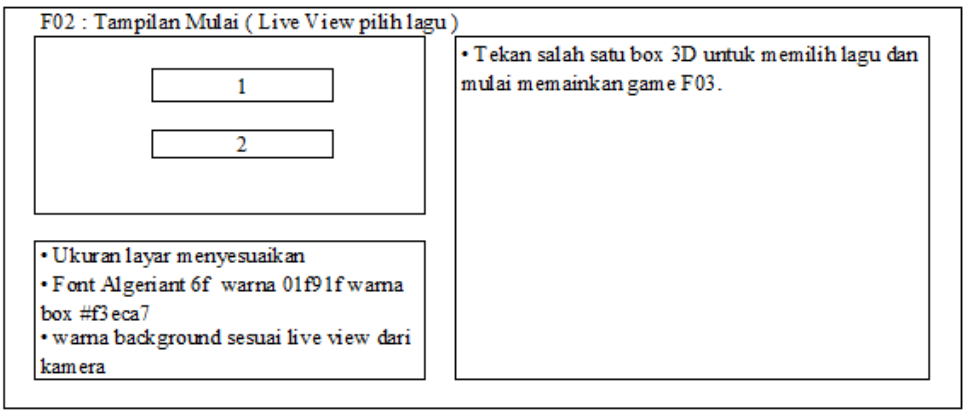
Perancangan menu utama mendeskripsikan rencana tampilan dari awal aplikasi *game rhythm* dibuka. Berikut merupakan gambaran dari perancangan antarmuka aplikasi *game rhythm*.



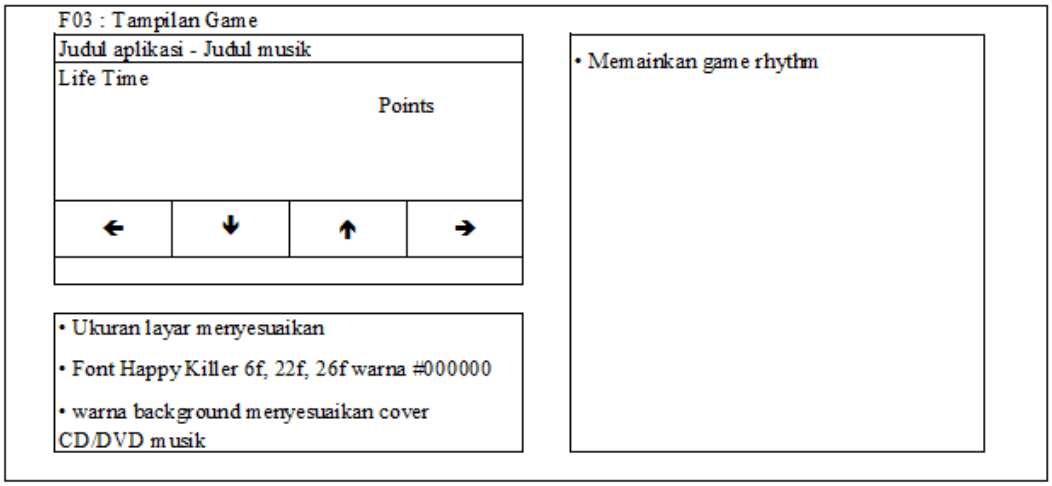
Gambar 3-1 Perancangan Menu utama

3.2.2 Perancangan Mulai *Game*

Perancangan pengaturan mendeskripsikan rencana tampilan dari permulaan permainan *game rhythm* Berikut merupakan gambaran dari perancangan mulai dari aplikasi *game rhythm*.



Gambar 3-2 Perancangan Mulai *Game*

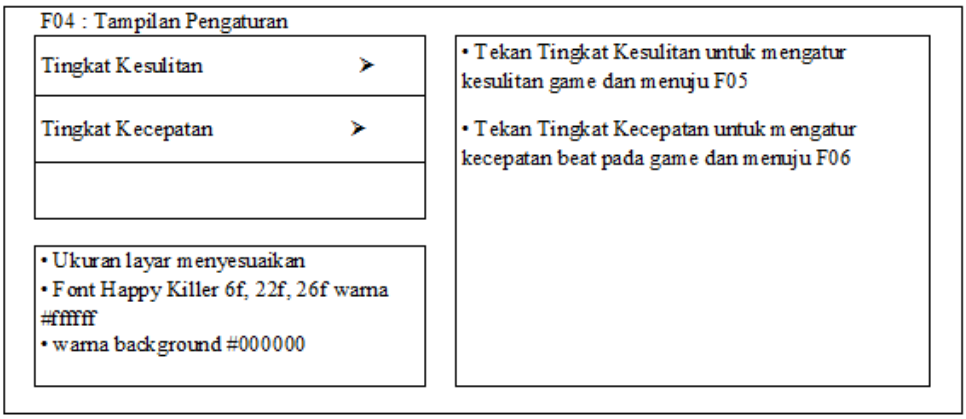


Gambar 3-3 Perancangan main *game*

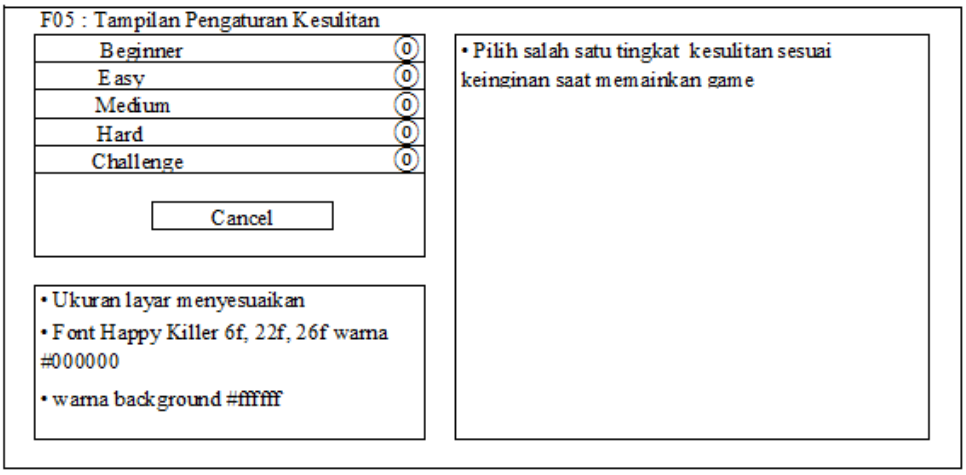
Pada saat *live view* akan muncul bentuk persegi panjang 3 dimensi yang tiap kotak telah diberikan nomor sesuai nomor urut lagu yang terdapat pada *cover* CD / DVD. Setalah memilih kotak / lagu yang dinginkan (dengan meng – *double* klik kotak tersebut) maka secara otomatis pengguna akan langsung memainkan *game rhythm* dengan latar lagu yang telah dipilih dan pengaturan yang telah dsesuaikan ataupun pengaturan yang *default* (pengaturan yang telah di atur oleh pengguna sebelumnya). Apabila pengguna telah selesai memainkan *game rhythm* maka akan muncul skor yang diperoleh selama memainkan *game rhythm*.

3.2.3 Perancangan Pengaturan

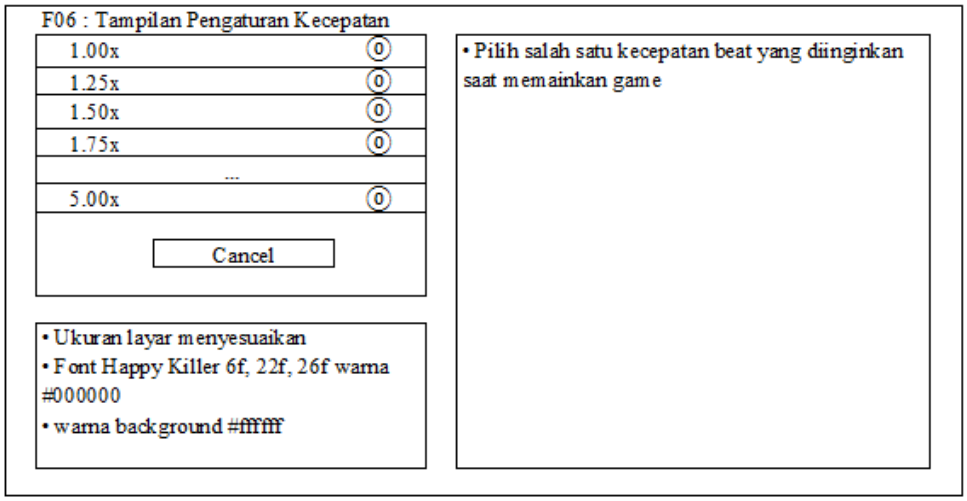
Perancangan pengaturan mendeskripsikan rencana tampilan dari pengaturan aplikasi *game rhythm* yaitu perancangan pengaturan kecepatan dan perancangan pengaturan kesulitan. Berikut merupakan gambaran dari perancangan pengaturan dari aplikasi *game rhythm*.



Gambar 3-4 Perancangan Pengaturan



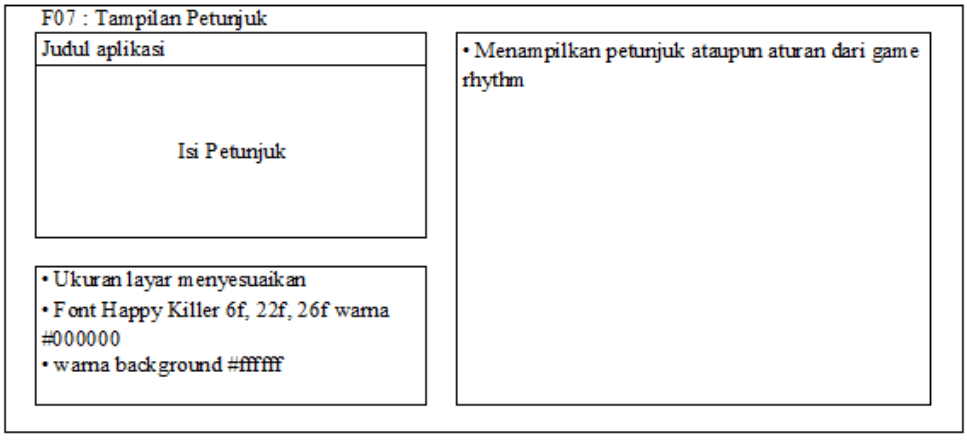
Gambar 3-5 Perancangan Pengaturan Kesulitan



Gambar 3-6 Perancangan Pengaturan Kecepatan

3.2.4 Perancangan Petunjuk

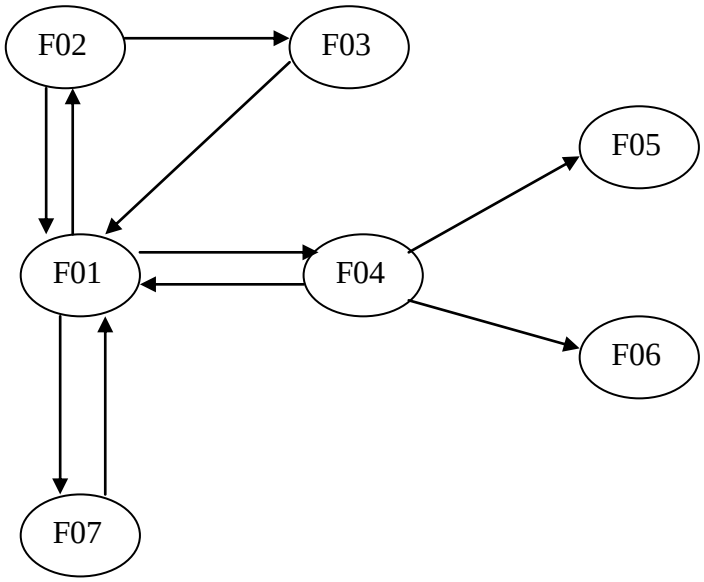
Perancangan petunjuk mendeskripsikan rencana tampilan dari petunjuk penggunaan aplikasi *game rhythm*. Berikut merupakan gambaran dari perancangan petunjuk dari aplikasi *game rhythm*.



Gambar 3-7 Perancangan Petunjuk

3.3 Jaringan Semantik

Jaringan semantik merupakan gambaran grafis yang menunjukkan antar objek (lingkaran) dan garis yang menggambarkan formasi antarobjek tersebut. Adapun jaringan semantik untuk aplikasi *game rhythm* dapat dilihat pada gambar 3.7.



Gambar 3-8 Jaringan Semantik *Game rhythm*