# BAB II
# LANDASAN TEORI

## 2.1 Konsep Dasar Sistem

Suatu sistem dapat didefinisikan sebagai suatu kesatuan yang terdiri dari dua atau lebih komponen atau subsistem yang berinteraksi untuk mencapai suatu tujuan. Suatu sistem dapat terdiri dari sistem-sistem bagian(subsistem). Masing-masing subsistem terdiri sebuah subsistem-subsitem yang lebih kecil lagi atau terdiri dari komponen-komponen. Subsistem-subsistem saling berinteraksi dan saling berhubungan membentuk satu kesatuan sehingga tujuan atau sasaran sistem tersebut dapat tercapai. Interaksi dari subsistem-subsistem sedemikian rupa, sehingga dicapai suatu kesatuan yang terpadu atau terintegrasi.

### 2.1.1 Sistem

Adapun beberapa pengertian sistem menurut para ahli :

1. Menurut Ludwig Von Bartalanfy menyatakan bahwa sistem merupakan unsur yang saling terikat dalam satu antar relasi diantara unsur-unsur tersebut dengan lingkungan.
2. Menurut Antol Raporot menyatakan bahwa sistem adalah suatu kumpulan kesatuan dan perangkat hubungan satu sama lain.
3. Menurut L. Ackof menyatakan bahwa setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian-bagian dalam keadaan saling tergantung satu sama lain.

Dari ketiga penjelasan sistem menurut para ahli tersebut, maka dapat di simpulkan bahwa Sistem adalah merupakan sekumpulan unsur / elemen yang saling berkaitan dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan.

### 2.1.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifa-sifat tertentu, yaitu mempunyai komponen-komponen (*component*), batasan sistem (*boundary*), keluaran (*output*), pengolah (*process*) dan sasaran (*objective* atau *goal*). Berikut adalah beberapa elemen yang membentuk sebuah sistem :

a. Komponen sistem

   Komponen sistem adalah sejumlah komponen yang saling berinteraksi, yang saling bekerja sama untuk membentuk satu kesatuan.

b. Batas sistem

   Batas sistem adalah merupakan batas suatu sistem dengan sistem lainnya maupun dengan lingkungan luar sistem tersebut.

c. Lingkungan luar sistem

   Lingkungan luar sistem adalah segala sesuatu diluar batas sistem yang dapat mempengaruhi sistem.Lingkungan sistem dapat bersifat menguntungkan dan dapat juga merugikan terhadap sistem.Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus tetap di jaga. Sedangkan lingkungan luar yang merugikan harus ditahan dan dikendalikan, kalau tidak akan mengganggu lingkungan sistem.

d. Penghubung sistem

   Penghubung sistem merupakan sebuah penghubung antara sebuah subsistem dengan subsistem lainnya. Keluaran (*output*) dari suatu sistem akan menjadi sebuah masukan (*input*) terhadap subsistem lainnya.

e. Masukan sistem

   Masukan input sistem adalah segala sesuatu yang masuk ke dalam suatu sistem yang kemudian menjadi bahan yang nantinya menjadi bahan untuk diproses.

f. Keluaran sistem

Keluaran adalah merupakan hasil dari sebuah proses subsistem yang nantinya bisa juga di gunakan sebakai masukan terhadap sistem lainnya.

g. Pengolah sistem

Suatu sistem dapat memiliki satu bagian yang berfungsi sebagai pengolah atau dapat juga menjadikan sistem tersebut sebagai pengolahnya.

h. Sasaran sistem

Suatu sistem pasti mempunyai tujuan (*goal*) atau sasaran (*objective*).Jika suatu sistem tidak memiliki sasaran atau tujuan maka sistem tersebut tidak ada manfaatnya. Sasaran dari sistem sangat menentukan sekali masukan yang di perlukan sistem dan keluaran yang akan di hasilkan sistem. Suatu sistem dikatakan berhasil jika suatu sistem tersebut mengenai sasaran atau tujuan.

## 2.2 Perangkat Lunak / Aplikasi

Program yang didesain untuk membantupengguna supaya dapat lebih produktifatau membantu dengan tugas-tugas khusus. Perangkat lunak biasa dikelompokan menjadi beberapa kelompok, diantaranya :

a. Program Sistem

Program sistem adalah program yang mengendalikan operasi komputer. Sebagai interface antara user perangkat lunak aplikasi dan perangkat keras komputer.

b. Program Aplikasi

Program aplikasi merupakan program yang langsung dibuat oleh seorang programmer yang disesuaikan dengan kebutuhan seseorang ataupun untuk kebutuhan, biasanya menggunakan bantuan suatu bahasa pemrograman.

## 2.3 Rekayasa Perangkat Lunak (RPL)

Rekayasa Perangkat Lunak adalah merupakan sebuah metode yang di gunakan untuk mendapatkan hasil sebuah perangkat lunak yang handal serta ekonomis dan juga dapat seefisien mungkin di gunakan. Adapun pengertian yang dikemukakan oleh Fritz Bauer yang menyatakan bahwa penerapan dan pemanfaatan prinsip-prinsip rekayasa untuk menghasilkan perngkat lunak yang ekonomis yang handal dan bekerja secara efisisen pada mesin-mesin yang nyata.

### 2.3.1 Proses Pembangunan RPL

a. SDLC

   SDLC (*Software Development Life Cycle*) adalah sekumpulan kegiatan dan keterhubungannya satu sama lain untuk pembangunan dari sebuah perangkat lunak.

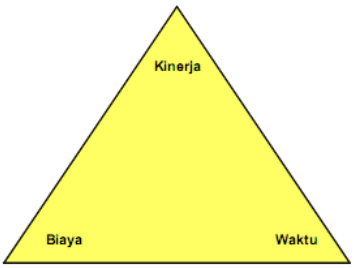
b. Pertanyaan yang sering muncul di dalam SDLC
   1. Aktifitas mana yang harus dipilih untuk proyek perangkat lunak ini?
   2. Hubungan apa yang dimiliki antar aktifitas-aktifitas di dalam SDLC?
   3. Bagaimana menjadwalkan aktifitas-aktifitas SDLC ini?

### 2.3.2 Proses Perangkat lunak

Suatu proses model adalah suatu representasi abstrak suatu model. Proses model menampilkan suatu deskripsi suatu proses dari beberapa perspektif tertentu. Proses PL merupakan aktifitas yang saling terkait (koheren) untuk menspesifikasikan, merancang, implementasi dan pengujian sistem perangkat lunak.

### 2.3.3 Tujuan Perangkat Lunak

Secara umum tujuan RPL tidak berbeda dengan bidang rekayasa yang lain. Hal ini dapat kita lihat pada Gambar di bawah ini.



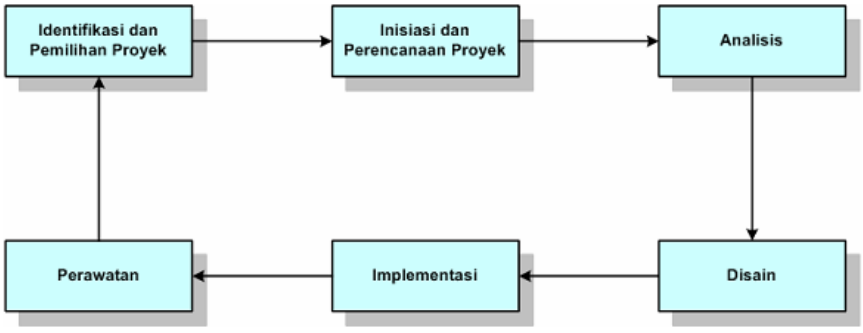

**Gambar 2.1. Tujuan RPL**

Dari Gambar 2.1 dapat diartikan bahwa bidang rekayasa akan selalu berusaha menghasilkan output yang kinerjanya tinggi, biaya rendah dan waktu penyelesaian yang tepat. Secara lebih khusus kita dapat menyatakan tujuan RPL adalah:

a. memperoleh biaya produksi perangkat lunak yang rendah
b. menghasilkan pereangkat lunak yang kinerjanya tinggi, andal dan tepat waktu
c. menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis platform
d. menghasilkan perangkat lunak yang biaya perawatannya rendah

### 2.3.4 Metode Rekayasa Perangkat Lunak

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untukmembantuprosespengembanganperangkatlunak.Model-modelinipadaumumnya mengacu pada model proses pengembangan sistem yang disebut ***System Development Life Cycle (SDLC)*** seperti terlihat pada Gambar berikut ini.




**Gambar 2.2. *System Development Life Cycle* (SDLC).**

1. Kebutuhan terhadap definisi masalah yang jelas. Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah. .
2. Tahapan-tahapan pengembangan yang teratur. Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum *analysis – design – coding – testing - maintenance*
3. *Stakeholder* berperan sangat penting dalam keseluruhan tahapan pengembangan. *Stakeholder* dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.
4. Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan.
5. Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis. Nilai dari sebuah perangkat lunak sebenarnya agak susah di-rupiah-kan. Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapatberupa penurunan biaya operasi, efisiensi penggunaan sumberdaya, peningkatan keuntungan organisasi, peningkatan "*image*" organisasi dan lain-lain.

### 2.3.5 Tahapan Rekayasa Perangkat Lunak

Meskipun dalam pendekatan berbeda-beda, namun model-model pendekatan memiliki kesamaan, yaitu menggunakan pola tahapan *analysis – design – coding(construction) – testing – maintenance*.

1. **Analisis sistem** adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi komponen-komponennya dengan

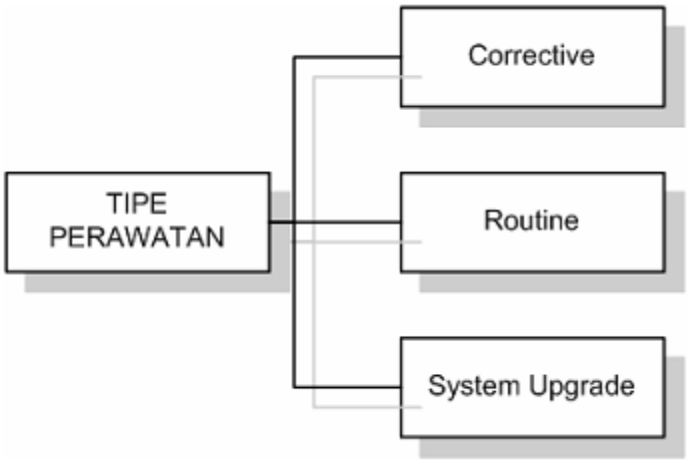
tujuan mempelajari seberapa bagus komponen-komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka.

Analisis mungkin adalah bagian terpenting dari proses rekayasa perangkat lunak. Karena semua proses lanjutan akan sangat bergantung pada baik tidaknya hasil analisis.Ada satu bagian penting yang biasanya dilakukan dalam tahapan analisis yaitu pemodelan proses bisnis.

2. **Model proses** adalah model yang memfokuskan pada seluruh proses di dalam sistem yang mentransformasikan data menjadi informasi (Harris, 2003). Model proses juga menunjukkan aliran data yang masuk dan keluar pada suatu proses. Biasanya model ini digambarkan dalam bentuk Diagram Arus Data (*Data Flow Diagram* / DFD). DFD meyajikan gambaran apa yang manusia, proses dan prosedur lakukan untuk mentransformasi data menjadi informasi.
3. **Disain perangkat lunak** adalah tugas, tahapan atau aktivitas yang difokuskan pada spesifikasi detil dari solusi berbasis computer (Whitten et al, 2004).Disain perangkat lunak sering juga disebut sebagai physical design. Jika tahapan analisis sistem menekankan pada masalah bisnis (business rule), maka sebaliknya disain perangkat lunak fokus pada sisi teknis dan implementasi sebuah perangkat lunak (Whitten et al, 2004).Output utama dari tahapan disain perangkat lunak adalah spesifikasi disain. Spesifikasi ini meliputi spesifikasi disain umum yang akan disampaikan kepada stakeholder sistem dan spesifikasi disain rinci yang akan digunakan pada tahap implementasi. Spesifikasi disain umum hanya berisi gambaran umum agar stakeholder sistem mengerti akan seperti apa perangkat lunak yang akan dibangun. Biasanya diagram USD tentang perangkat lunak yang baru merupakan point penting dibagian ini. Spesifikasi disain rinci atau kadang disebut disain arsitektur rinci perangkat lunak diperlukan untuk merancang sistem sehingga memiliki konstruksi yang baik,

proses pengolahan data yang tepat dan akurat, bernilai, memiliki aspek *user friendly* dan memiliki dasar-dasar untuk pengembangan selanjutnya.Desain arsitektur ini terdiri dari desain *database*, desain proses, desain *user interface* yang mencakup desain *input, output form* dan *report,* desain *hardware, software* dan jaringan. Desain proses merupakan kelanjutan dari pemodelan proses yang dilakukan pada tahapan analisis.

4. **Konstruksi** adalah tahapan menerjemahkan hasil disain logis dan fisik ke dalam kode-kode program komputer.
5. **Pengujian** sistem melibatkan semua kelompok pengguna yang telah direncanakan pada tahap sebelumnya. Pengujian tingkat penerimaan terhadap perangkat lunak akan berakhir ketika dirasa semua kelompok pengguna menyatakan bisa menerima perangkat lunak tersebut berdasarkan kriteria-kriteria yang telah ditetapkan.
6. **Perawatan dan Konfigurasi.** Ketika sebuah perangkat lunak telah dianggap layak untuk dijalankan, maka tahapan baru menjadi muncul yaitu perawatan perangkat lunak. Ada beberapa tipe perawatan yang biasa dikenal dalam dunia perangkat lunak seperti terlihat pada diagram di Gambar 2.3 di bawah ini :




**Gambar 2.3. Tipe-tipe perawatan**

## 2.4 Database

*Database* adalah kumpulan dari file-file yang saling berhubungan atau berelasi, relasi tersebut bisa di tunjukan dengan kunci dari tiap file yang ada. Satu *database* menunjukan satu kumpulan data yang dipakai dalam satu lingkup perusahaan atau instansi. Dalam satu file terdapat *record-record* yang sejenis, sama besar , sama bentuk yang merupakan sekumpulan *entity* yang seragam. Satu *record* terdiri dari *field-field* yang saling berhubungan untuk menunjukan bahwa *field* tersebut dalam satu pengertian yang lengkap dalam satu *record*.

Untuk mengidentifikasi dari sebuah *field* maka digunakan *atribute* atau merupakan judul dari satu kelompok *entity* tertentu. *Entity* adalah suatu objek yang nyata dan akan direkam.Set proram pengelola merupakan satu paket program yang dibuat agar memudahkan dan mengefisienkan pemasukan atau pembacaan informasi ke dalam sebuah *database*.

### 2.4.1 Entity

*Entity* adalah objek yang dapat berupa orang, kejadian maupun sebuah konsep yang informasinya direkam.

### 2.4.2 Atribut

Setiap *entity* memiliki sebuah atribut atau bisa dikatan bagian yang mewakili sebuah entity.

### 2.4.3 Record

*Record* adalah kumpulan elemen-elemen yang saling berkaitan yang menginformasikan tentang suatu entity secara lengkap.

### 2.4.4 File

*File* adalah kumpulan *record* yang sejenis yang memiliki panjang elemen yang sama , namun memiliki data *value* yang berbeda.

### 2.4.5 KeunggulanDatabase/ SyaratDatabase

*Penyusunan* sebuah database digunakan untuk mengatasi masalah-masalah pada penyusunan data yaitu:

1. Redudansi dan inkonsistensi data.
2. Kesulitan pengaksesan data.
3. Isolasi data untuk standardisasi.
4. Multiple user.
5. Masalah keamanan.
6. Masalah integrasi.
7. Masalah data independence.

## 2.5 Pemrograman Berorientasi Objek (Object-Oriented)

Bahasa C++, *Java*, VB.NET dan sejenisnya kita kenal sebagai bahasa pemograman berorientasi objek. Tetapi hanya dengan menggunakan bahasa tersebut bukan berarti kita sudah pasti membuat program yang berorientasi objek. Bagi yang pernah berkecimpung dalam pemrograman visual basic, pernahkah membuat class? . perlu diketahui bahwa kelas adalah tempat berkumpulnyaobjek yang merupakan cirikhas pemrograman berorientasi objek.

### 2.5.1 Analisis dan Desain Berorientasi Objek (Object-Oriented Analysis dan Design Process)

Pemrograman berorientasi objek bekerja dengan baik ketika di barengi dengan *Object-Oriented Analysis dan Design Process*(OOAD).(wampler, 2001:2) mengatakan jika kita membuat program berorientasi objek tanpa OOAD, ibarat membangun rumah tanpa terlebih dahulu menganalisa apa saja yang dibutuhkan oleh rumah itu, tanpa perencanaan, tanpa *blueprint*, tanpa menganalisis ruangan apa saja yang akan diperlukan, berapa besar rumah yang akan di bangun dan sebagainya.

#### 2.5.1.1 Objek (Object)

Orientasi objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan titik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak saat ini. Orientasi objek merupakan teknik pemodelan sistem riil yang

berbasis objek. Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata. Menurut (Douglas, 2004: bab 2.1) objek adalah entitas yang memiliki atribut, karakter (*behavior)* dan kadangkala disertai kondisi (*state)*. Objek dapat merepresentasikan sistem *real*, sistem yang merepresentasikan sesuatu dalam bentuk konsep bahkan bisa juga dalam bentuk visualisasi. Semuanya memiliki fitur atribut (untuk data) , *behavior* (operation atau method), keadaan (memori), identitas dan tanggung jawab. Proses menjabarkan sitem nyata menjadi objek dinamakan abstraksi (*abstraction*). Abstraksi mengeliminir aspek yang tidak perlu dalam suatu objek.

#### 2.5.1.2 Kelas (Class)

Kelas adalahpenggambaran suatu set objek yang memiliki atribut dan behavior yang sama. Kelas mirip dengan tipe data pada pemrograman non objek, tapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Kita dapat membentuk kelas baru yang lebih spesifik dari kelas general-nya. Kelas dan objek adalah merupakan jantung dari pemrograman berorientasi objek.

#### 2.5.1.3 Pembungkusan(Encapsulation)

(Nugroho, 2005:6) mengartikan pembungkusan sebagai penggabungan potongan-potongan informasi dan perilaku-perilaku spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi apa yang di sebut dengan objek.

#### 2.5.1.4 Pewarisan(Inheritance)dan Generalisasi/Spesialisasi

Menurut (Whitten, 2004: 411), konsep dimana metode dan atau atribut yang ditentukan didalam sebuah objek kelas dapat diwariskan atau digunakan lagi oleh objek kelas lainnya. Sedangkan generalisasi/spesialisasi merupakan teknik dimana atribut dan perilaku yang umum pada beberapa tipe kelas objek, dikelompokan (atau diabstraksi) ke dalam kelasnya sendiri (dinamakan *supertype*). Atribut dan metode kelas *supertype* kemudian diwariskan oleh kelas objek tersebut (dinamakan *subtype*).

#### 2.5.1.5 Polimorfisme

*Polimorfisme* berarti suatu fungsionalitas yang diimplementasikan dengan berbagai cara yang berbeda. Pada terminologi berorientasi objek, ini berarti kita dapat memiliki berbagai implementasi untuk sebagian fungsionalitas tertentu. Manfaat dari *polimorfisme* ialah kemudahan pemeliharaannya. Jika perlu menambahkan gambar baru (misal segitiga), maka cukup menambahkan fungsi baru (fungsi menggambar segitiga) sedangkan fungsi umumnya (fungsi gambar) tidak mengalami perubahan (Nugroho, 2005: 10).

### 2.5.2 Unified Modelling Language (UML)

*UML*singkatan dari *Unifed Modeling Language* yang berarti bahasa pemodelan standar. (Chonoles, 2003: bab1) mengatakan sebagai bahasa, berarti *UML* meiliki sintaks dan semantic. Ketika kita membuat model menggunakan konsep *UML* ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan yang lainnya harus mengikuti standar yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. *UML*diaplikasikan untuk maksud tertentu, biasanya antara lain digunakan untuk:

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan system.
4. Mendokumentasi system yang ada, proses-proses dan organisasinya.

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis timing diagram) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang *system* berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan system yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dan mengaplikasikan berbagai *system*. Sebagai perancang *system*, mau tidak mau pasti

akan menjumpai UML, baik kita kita sendiri yang membuat atau sekedar membaca diagram UML buatan orang lain (Pilone, 2005: bab1).

### 2.5.3 Diagram-Diagram UML

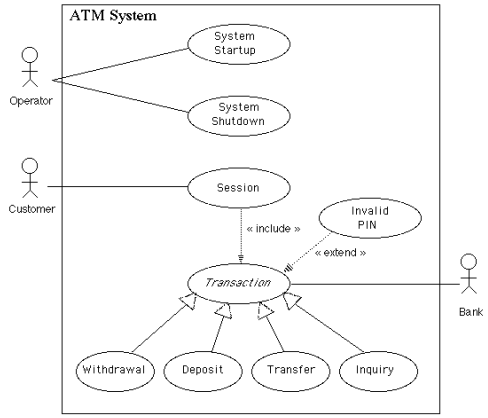
Model-model untuk diagram UML dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis, jenis diagram tersebut antara lain :

1. Diagram Kelas. Bersifat statis, diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi.
2. Diagram Paket (*Package Diagram*). Bersifat statis, diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
3. Diagram *Use-Case*. Bersifat statis, diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas).
4. Diagram Interaksi dan *Sequence* (urutan). Bersifat dinamis, diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram Komunikasi (*Communication Diagram*). Bersifat dinamis, diagram sebagai pengganti diagram kolaborasi UML 1.4 yang menekankan organisasi structural dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram*). Bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem , memuat status, transisi kejadian serta aktivitas.
7. Diagram Aktivitas (*Aktivity Diagram*). Brsifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem.
8. Diagram Komponen (*Component Diagram*). Bersifat statis. Digram komponen ini memperlihatkanorganisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya.

9. Diagram Deployment (*Deployment Diagram*). Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*).

### 2.5.4 UseCase Diagram

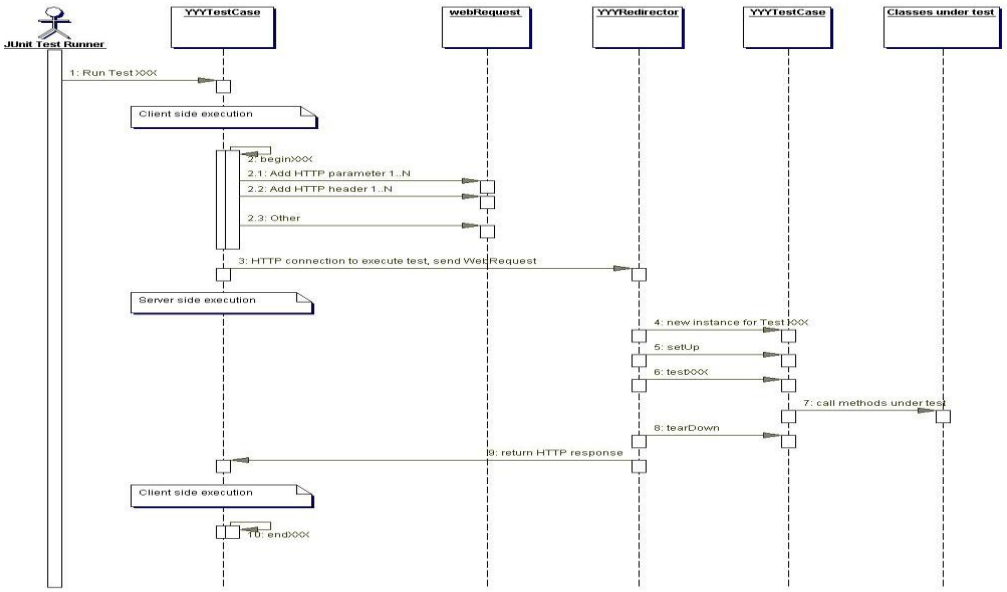
Menurut (Pilone, 2005:bab 7.1) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen, kejadian atau kelas. Sedangkan (Whitten, 2004: 258) mengartikan *use case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario), baik terotomasisasi maupun secara manual, untuk tujuan bisnis tunggal. Salah satu kontributor terhadap diagram *use case* dalam UML adalah Ivar Jacobsen. *Use case* menggambarkan *external view* dari sistem yang akan kita buat modelnya.(Pooley, 2003: 15) mengatakan bahwa model *use case*, tetapi yang perlu di ingat, diagram tidak identik dengan model karena model lebih luas dari diagram.




**Gambar 2.4 Diagram *Use Case***

### 2.5.5 Sequence Diagram

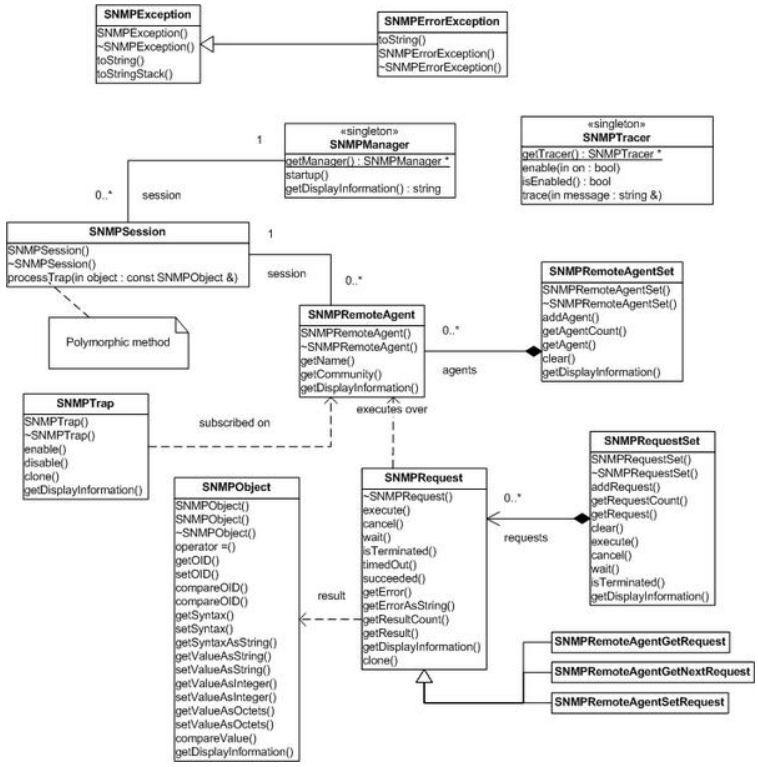
*Sequence diagram*menjelaskan secara detil urutan proses yang dilakukandalamsistemuntukmencapaitujuandari*use case.*Interaksiyang terjadiantar*class*,operasiapasajayangterlibat,urutanantaroperasi,dan informasiyangdiperlukan oleh masing-masingoperasi.




**Gambar 2.5*Sequence Diagram***

### 2.5.6 ClassDiagram

*Class* adalah suatu diagram yang memperlihatkan atau menampilkan struktur darisebuah sistem,sistem tersebut akan menampilkan sistem kelas,atribut dan hubungan antara kelasketika suatu sistem telah selesai membuat diagram.*Class* diagram menunjukkan hubungan antar*class*dalam sistem yang sedang dibangun dan bagaimana mereka salingberkolaborasi untuk mencapai suatu tujuan.




**Gambar 2.6 Class Diagram**

## 2.6 Citra Digital

Data atau informasi tidak hanya disajikan dalam bentuk teks, tapi juga dapat berupa gambar, *audio* (bunyi, suara, musik) dan *video*. Keempat macam data atau informasi ini sering disebut multimedia. Citra (*image*) istilah lain untuk gambar sebagai satu komponen multimedia memegang peranan penting sehingga bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi "sebuah gambar akan lebih bermakna dari seribu kata" *(a picture is more than a thousand words*). Maksudnya tentu sebuah gambar dapat memberikan informasi yang lebih banyak dari pada informasi tersebut disajikan dalam bentuk kata-kata (tekstual).

Istilah citra atau *image* yang pada umumnya digunakan dalam bidang pengolahan citra diartikan sebagai suatu fungsi kontinu dari intensitas cahaya *f(x,y)* dalam bidang dua dimensi dengan (*x,y*) menyatukan suatu koordinat

dangan nilai $f$ pada setiap titik menyatukan intensitas atau tingkatan kecerahan atau derajat keabuan (*brightness/gray level*). Suatu citra digital adalah suatu citra kontinyu yang diubah kedalam bentuk diskrit, baik koordinat maupun intensitas cahayanya. Kita dapat menganggap suatu citra digital sebagai suatu matriks, dimana indeks baris dan kolomnya menyatakan koordinat sebuah titik pada citra tersebut dan nilai masing-masing elemennya menyatakan intensitas cahaya pada titik tersebut.

## 2.7 Video digital

*Video digital* adalah menangkap, merekam, memproses, menyimpan, dan merekonstruksi suatu urutan dari beberapa gambar. *Videodigital* pada dasarnya tersusun atas serangkaian *frame*yang ditampilkan dalam kecepatan tertenu(*frame/detik*). Jika laju *frame* cukup tinggi, maka mata manusia melihatnya sebagai rangkaian yang kontinu. Masing-masing *frame*merupakan gambar/ citra digital. Suatu gambar digital direpresentasikan dengan sebuah matrik yang masing-masing elemennya merepresentasikan nilai intensitas.

## 2.8 Augmented Reality

Realitas tertambah, atau kadang dikenal dengan singkatan bahasa Inggrisnya AR (*Augmented Reality*), adalah teknologi yang menggabungkan benda maya dua dimensi dan ataupun tiga dimensi ke dalam sebuah lingkungan nyata tiga dimensi lalu memproyeksikan benda-benda maya tersebut dalam waktu nyata.Tidak seperti realitas maya yang sepenuhnya menggantikan kenyataan, realitas tertambah sekedar menambahkan atau melengkapi kenyataan.

### 2.8.1 SejarahAugmented Reality

Secara umum, *Augmented Reality* (AR) adalah penggabungan antara objek *virtual* dengan objek nyata. Sebagai contoh, adalah saat stasiun televisi, menyiarkan pertandingan sepak bola, terdapat objek *virtual*, tentang skor pertandingan yang sedang berlangsung. Menurut Ronald Azuma pada tahun 1997, *Augmented Reality* adalah menggabungakan dunia nyata dan virtual, bersifat interaktif secara *real time*, dan merupakan animasi 2D maupun 3D. Sejarah

tentang *Augmented Reality* dimulai dari tahun 1957-1962, ketika seorang penemu yang bernama Morton Heilig, seorang sinematografer, menciptakan dan memapatenkan sebuah simulator yang disebut Sensorama dengan *visual*, getaran dan bau. Pada tahun 1966, Ivan Sutherland menemukan *head-mounted display* yang dia klaim adalah jendela ke dunia virtual.

Tahun 1975 seorang ilmuwan bernama Myron Krueger menemukan *Videoplace* yang memungkinkan pengguna, dapat berinteraksi dengan objek *virtual* untuk pertama kalinya. Tahun 1989, Jaron Lanier, memeperkenalkan *Virtual Reality* dan menciptakan bisnis komersial pertama kali di dunia maya, Tahun 1992 mengembangkan *Augmented Reality* untuk melakukan perbaikan pada pesawat boeing, dan pada tahun yang sama, LB Rosenberg mengembangkan salah satu fungsi sistem AR, yang disebut *Virtual Fixtures*, yang digunakan di Angkatan Udara AS Armstrong Labs, dan menunjukan manfaatnya pada manusia, dan pada tahun 1992 juga, Steven Feiner, Blair Maclntyre dan dorée Seligmann, memperkenalkan untuk pertama kalinya Major Paper untuk perkembangan Prototype AR.

Pada tahun 1999, Hirokazu Kato, mengembangkan ArToolkit di HITLab dan didemonstrasikan di SIGGRAPH, pada tahun 2000, Bruce.H.Thomas, mengembangkan ARQuake, sebuah Mobile Game AR yang ditunjukan di InternationalSymposium on Wearable Computers.Pada tahun 2008, Wikitude AR Travel Guide, memperkenalkan Android G1 Telephone yang berteknologi AR, tahun 2009, Saqoosha memperkenalkan FLARToolkit yang merupakan perkembangan dari ArToolkit. FLARToolkit memungkinkan kita memasang teknologi AR di sebuah website, karena output yang dihasilkan FLARToolkit berbentuk Flash. Ditahun yang sama, *Wikitude Drive* meluncurkan sistem navigasi berteknologi AR di Platform Android. Tahun 2010, Acrossair menggunakan teknologi AR pada I-Phone 3GS.

Bidang-bidang yang pernah menerapkan teknologi *Augmented Reality* adalah :

1. Kedokteran (*Medical*): Teknologi pengambaran sangat dibutuhkan di dunia kedokteran, seperti misanya, untuk simulasi operasi, simulasi pembuatan

vaksin virus, dll. Untuk itu, bidang kedokteran menerapkan *Augmented Reality* pada visualisasi penelitian mereka.

2. Hiburan (*Entertainment*): Dunia hiburan membutuhkan *Augmented Reality* sebagai penunjang efek-efek yang akan dihasilkan oleh hiburan tersebut. Sebagai contoh, ketika sesorang wartawan cuaca memperkirakan ramalan cuaca, dia berdiri di depan layar hijau atau biru, kemudian dengan teknologi *Augmented Reality*, layar hijau atau biru tersebut berubah menjadi gambar animasi tentang cuaca tersebut, sehingga seolah-olah wartawan tersebut, masuk ke dalam animasi tersebut.
3. Latihan Militer (*Military Training*): Militer telah menerapkan *Augmented Reality* pada latihan tempur mereka. Sebagai contoh, militer menggunakan *Augmented Reality* untuk membuat sebuah permainan perang, dimana prajurit akan masuk kedalam dunia game tersebut, dan seolah-olah seperti melakukan perang sesungguhnya.
4. *Engineering Design*: Seorang *engineering design* membutuhkan *Augmented Reality* untuk menampilkan hasil *design* mereka secara nyata terhadap klien. Dengan *Augmented Reality* klien akan tahu, tentang spesifikasi yang lebih detail tentang *desain* mereka.
5. *Robotics* dan *Telerobotics*: Dalam bidang robotika, seorang operator robot, mengunnakan pengendari pengambaran *visual* dalam mengendalikan robot itu. Jadi, penerapan *Augmented Reality* dibutuhkan di dunia robot.
6. *Consumer Design*: *Virtual reality* telah digunakan dalam mempromsikan produk. Sebagai contoh, seorang pengembang menggunakan brosur *virtual* untuk memberikan informasi yang lengkap secara 3D, sehingga pelanggan dapat mengetahui secara jelas, produk yang ditawarkan.

### 2.8.2 Tujuan AR

Tujuan utama dari sistem *Augmented Reality* akan berwujud sebagai sebuah kacamata atau proyektor retina yang akan menyediakan tampilan informasi yang relevan, dipetakan ke lingkungan sekitarnya secara realtime. Misalnya, saat melihat sebuah restoran dengan kacamata *Augmented Reality*,

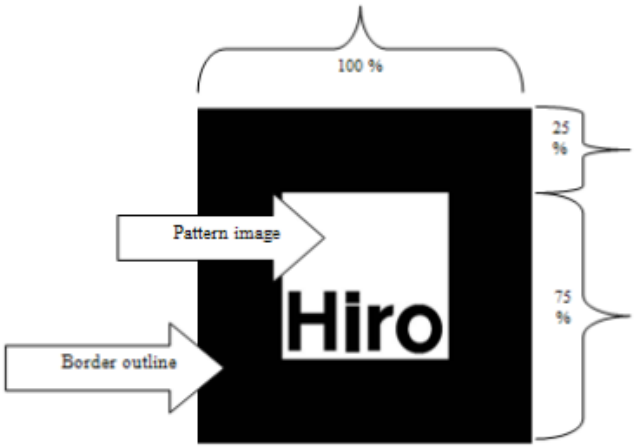
maka satu panggilan otomatis langsung ke database review atau menu dari website restoran tersebut. Seorang ilmuwan yang bekerja pada perusahaan farmasi bisa menggunakan kacamata untuk menampilkan model 3D dari berbagai molekul dan menggunakannya untuk memvisualisasikan obat yang lebih baik. Anak mungkin menggunakan jaringan yang terhubung kacamata AR untuk bermain *video* game kehidupan nyata yang memungkinkan menembakkan "laser" dari tangan mereka, meski kemungkinan agak terbatas.

*Augmented Reality* bergantung pada kemajuan teknologi miniaturisasi dan komputasi bergerak (*mobile computing*). Saat ini, teknologi sistem proyeksi yang efektif dan komputer kecil dan cepat masih kurang matang untuk benar-benar dapat membuat antarmuka *Augmented Reality* yang afektif, walaupun kita sudah semakin dekat pada kesempurnaan dengan riset yang dilakukan terus menerus oleh para ahli. Purwarupa telah banyak dibuat selama beberapa dekade, tetapi produk utama yang benar-benar layak belum ada. Salah satu teknologi yang menjanjikan adalah proyeksi retina, sehingga kacamata tidak digunakan lagi. Sistem proyeksi retina komersial sebenarnya sudah ada, namun resolusi dan palet warnanya masih sangat rendah.

*Augmented Reality* saat ini sebenarnya sudah ada dalam bentuk yang belum sempurna. Misalnya, komentator olahraga sering menggunakan pena cahaya untuk "menggambar" di lapangan sepak bola untuk memberikan bantuan visual untuk rekan komentatornya. Contoh lain yaitu gambar iklan yang sering tampil dilapangan bulutangkis ketika ditayangkan di TV, seakan gambar iklan ini melekat pada lapangan. Gambar iklan sponsor ini terus berganti-ganti selama pertandingan. Ini menunjukkan bukti dari konsep AR meski masih terbatas.

## 2.9 Marker

*Marker* adalah suatu pola yang dibuat dalam bentuk gambar yang akan dikenali oleh webcam. *Marker* adalah kunci dari AR. Informasi *marker* akan digunakan untuk menampilkan sebuah objek. *Marker* juga merupakan gambar yang terdiri atas border outline dan pattern image seperti terlihat pada gambar dibawah ini.




**Gambar 2.7 Marker**

*Marker* biasanya dengan warna hitam dan putih. Cara pembuatannya pun sederhana tetapi harus diperhatikan ketebalan *Marker* yang akan dibuat, ketebalan *Marker* jangan kurang dari 25 % dari panjang garis tepi agar pada saat proses deteksi *Marker* dapat lebih akurat. Nama Hiro pada gambar 2.15 merupakan pembeda saja. Sedangkan objek warna putih sebagai *background*, yang nantinya akan digunakan sebagai tempat objek yang akan di*render*.

Warna putih pada *Marker* menunjukan warna sebuah objek, sedangkan warna hitam menunjukan latar belakang.Intensitas warna pada suatu objek memiliki warna yang lebih rendah (gelap), sedangkan latar belakang mempunyai intensitas yang lebih tinggi (terang). Adapun beberapa aturan umum dalam pembuatan pola marker, yaitu seperti harus kotak berbingkai hitam dan ini adalah rahasia dari pelacakan sebuah *marker*, ukuran tidak lebih dari 631x634 pixel. Warna selain hitam putih masih bisa dikenali oleh system *Marker* membantu komputer dimana letak objek akan ditamplikan di display.




Gambar 2.8 Contoh Marker

Ukuran *Marker* yang digunakan dapat mempengaruhi penangkapan pola *Marker* oleh kamera. Semakin besar ukuran *Marker*maka semakin jauh jarak yang dapat ditangkap oleh kamera dalam pendeteksian*Marker*. Namun disinilah masalahnya, ketika *Marker* bergerak menjauhi kamera, jumlah *pixel* pada layar kamera menjadi lebih sedikit dan ini bisa mengakibatkan pendeteksian tidak akurat.

### 2.9.1 Markerless

Salah satu metode *Augmented Reality* yang saat ini sedang berkembang adalah metode "*MarkerlessAugmented Reality*", dengan metode ini pengguna tidak perlu lagi menggunakan sebuah *marker* untuk menampilkan elemen-elemen digital.

Dalam perancangannya, seolah-olah*markerless* menggabungkan objek virtual dengan objek nyata, dalam hal ini objek virtual berupa objek 2D atau 3D dan objek nyatanya berupa gambar dengan pola tertentu (*markerless*). Sistem *Augmented Reality Display* yang digunakan adalah teknik *spatial displaydengan screen display (bisa menggunakan monitor ataupun proyektor).*

Secara garis besarnya, dalam perancangan aplikasi ini ada tiga bagian utama yaitu sebagai berikut :

1. Inisialisasi
2. Tracking marker
3. Rendering Objek 3D

Adapun pengembangan *markerless*yangtelah di kembangkan oleh pengembang terbesar di dunia seperti :

1) ***Face Tracking***

Dengan menggunakan alogaritma yang mereka kembangkan, komputer dapat mengenali wajah manusia secara umum dengan cara mengenali posisi mata, hidung, dan mulut manusia, kemudian akan mengabaikan objek-objek lain di sekitarnya seperti pohon, rumah, dan benda-benda lainnya.



**Gambar 2.9Face Tracking**

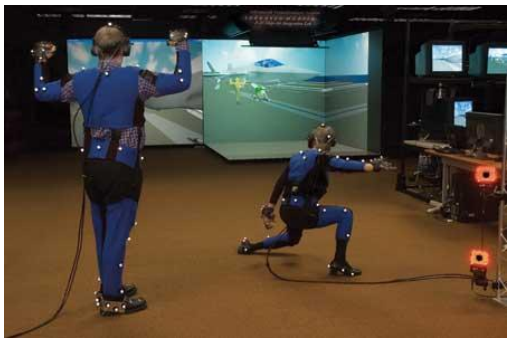
2) **3D *Object Tracking***

Berbeda dengan *Face Tracking* yang hanya mengenali wajah manusia secara umum, teknik 3D *Object Tracking* dapat mengenali semua bentuk benda yang ada disekitar, seperti mobil, meja, televisi, dan lain-lain.

3) ***Motion Tracking***

Pada teknik ini komputer dapat menangkap gerakan, *Motion Tracking* telah mulai digunakan secara ekstensif untuk memproduksi film-film yang mencoba mensimulasikan gerakan.

Contohnya pada film Avatar, di mana James Cameron menggunakan teknik ini untuk membuat film tersebut dan menggunakannya secara *realtime*.



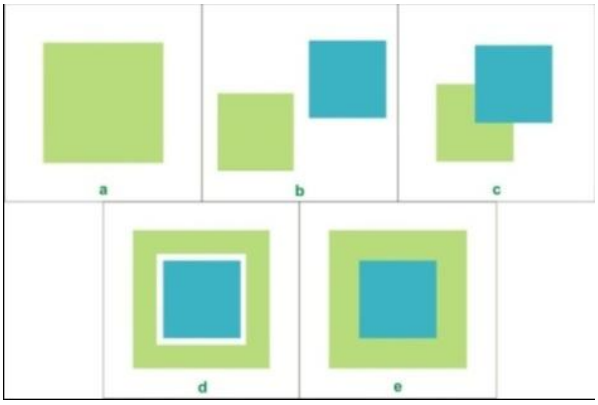
**Gambar 2.10Motion Tracking**

## 2.10 Multi Marker

*Multi marker* adalah merupakan sebuah metode perkembangan dari single marker, dimana kamera men-*Tracking*objek yang di tangkap lebih dari satu. Dalam implementasinya dapat di lakukan dengan menggunakan beberapa pendekatan metode yang dapat di lakukkan seperti pelabelan komponen serta *corner detection* sebagai pengenalan sudut dari beberapa bentuk *marker*.

### 2.10.1 Occlusion

*Occlusion*adalah hubungan antara suatu benda dengan benda lain jika kita lihat dari suatu sudut pandang. Hal ini tentunya mengurangi informasi antara objek dalam lingkungan 3D, karena jika dilihat dari sudut pandang makalingkungan 3D akan diproyeksi kepada suatu bidang sehingga seolah-olah menjadi lingkungan 2D. pengurangan di mensi ini menyebabkan informasi interaksi antar objek seperti keadaan bersinggungan, beirisan, atau beberapa jarak antar objek akan menjadi ambigu.




**Gambar 2.11 *Occlusion* yang terjadi karena interaksi antar objek.**

*Occlusion detection* adalah metode untuk mendeteksi ada tidaknya occlusion dalam penampilan objek 3D. Pada [Gun A, Mark, dan Gerard, 2004] secara sederhana *occlusion detection* hanya mendefinisikan keadaan dimana suatu marker tidak terdeteksi karena tertutup oleh benda lain. Sedangkan pada [Volkert, Stephen, Mark, 2004] menggunakan *occlusion detection* berdasarkan posisi koordinat 2D dari dua objek yang ada.

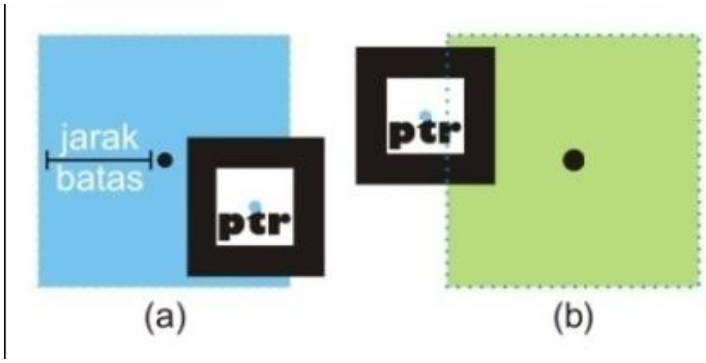
Jika ada n objek yang diwakili matrik O, maka akan dihasilkan matrik O1, O2, . ., On yang merupakan posisi proyeksi objek-1, objek-2, . . ., objek-n di layar. Deteksi occlusion akan dilakukan dengan pengecekan 2 objek missal dipilih objek-1 terhadap objek-2 maka akan dilakukan pengecekan syarat pertidaksamaan point clipping berikut [Donald, dan Baker, 1996] :

$$02x - batas \leq 01x \leq 02x + batas \qquad (5)$$

$$02y - batas \leq 01y \leq 02y + batas \qquad (6)$$

Hasil deteksi ini berupa nilai kebenaran yang merupakan dasar pendefinisian *event* dari interaksi *occlusion based* jikapertidaksamaan 5 dan 6 terpenuhi.Interaksi *occlusion based* adalah sebuah desain interaksi eksosentris [Hannah, Matthew, Rudi, dan Bruce, 2001]dimana dalam mendefinisikan *event* untuk menghasilkan aksinya menggunakan metode *occlusion detection* diatas.

Desaininteraksi yang menggunakan proyeksi 2D dari objek 3D ini mengurangi kompleksitas yang diperlukan dalam mendesaininteraksi dalam sistem AR lain yang menggunakan acuan bidang 3D.




**Gambar 2.12 (a) terjadi *event* (b) tidak terjadi *event***

Jika titik biru ditengah *marker* ptr adalah objek O1 dan titik hitam adalah objek O2 maka gambar (a) dikatakanterjadi *event* karena memenuhi pertidaksamaan 5 dan 6 yaitu koordinat O1(x,y) ada di dalam batas area O2. Sedangkangambar 3(b) tidak terjadi *event* karena hanya memenuhi pertidaksamaan 6 (nilai O1y ada dalam batas O2y) namun tidakmemenuhi pertidaksamaan 5.

## 2.11 Android

Androidadalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc.membeli Android Inc., pendatang baru yang membuat peranti lunakuntuk ponsel. Kemudian untuk mengembangkan Android, dibentuklahOpen Handset Alliance, konsorsium dari 34 perusahaan peranti keras,peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel,Motorola, Qualcomm, T-Mobile, dan Nvidia.Pada saat perilisan perdana Android, 5November 2007, Android bersama Open Handset Alliancemenyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode – kode Android dibawah lisensi Apache, sebuah lisensi

perangkat lunak dan standar terbuka perangkat seluler. Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Services (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution(OHD).

### 2.11.1 SejarahAndroid

Android pertama kali dikembangkan oleh perusahaan bernama AndroidInc. kemudian pada tahun 2005, Google mengakusisi perusahaan ini sehingga industri IT ketika itu beranggapan akan muncul istilah gPhone dengan langkah Google tersebut.

Pada tahun 2007, Google dan beberapa perusahaan yang tergabung dalam Open Handset Alliance (Intel, Nvidia, Texas Intrusment) mengembangkan sistem operasi Android dan resmi menjadi *open-source*.

Pada tahun 2008, AndroidSDK 1.0 diluncurkan dan phone G1 yang di produksi oleh HTC menggunakan sistem operasi tersebut.

Pada tahun 2009, versi terbaru dari sistem operasi Android diluncurkan mulai dari versi 1.5(*Cupcake*), versi 1.6(*Donut*), dan versi 2.0/2.1 (*Eclair*).Hal ini di dukung dengan lebih dari 20 gadget yang menggunakan versi tersebut.

Pada tahun 2010, Android menjadi sistem operasi Blackbarrydan menjadi sistem operasi terbaik pada platform smart-phone.Versi 2.2(*Froyo*) diluncurkan dan lebih dari 60 gadget menggunakannya.

Pada tahun 2011, versi 2.3 (*Gungerbread*) dan 3.0 (*Honeycomb*) berturut-turut diluncurkan.

### 2.11.2 Beberapa VersiAndroid

1. Android versi 1.1

   Pada 9 Maret 2009, Google merilis Android versi 1.1.Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm,voice

search(pencarian suara), pengiriman pesan denganGmail, dan pemberitahuan email.

2. Android versi 1.5 (*Cupcake*)

   Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (SoftwareDevelopment Kit) dengan versi 1.5 (Cupcake). Penambahan fitur dalam seluler versi ini yakni kemampuan merekam dan menonton *video* dengan modus kamera, mengunggah *video* ke Youtube dangambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP,kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

3. Android versi 1.6 (*Donut*)

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik disbanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN.Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, camcorder dan galeri yangdintegrasikan; CDMA / EVDO, 802.1x, VPN, Gestures, dan Text-to-speech engine; kemampuan dial kontak; teknologitext to changespeech(tidak tersedia pada semua ponsel; pengadaan resolusi VWGA.

4. Android versi 2.0/2.1 (*Eclair*)

   3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahanUI dengan browser baru dan dukungan HTML5, daftar kontak yangbaru, dukungan flashuntuk kamera 3,2 MP, digital Zoom, danBluetooth 2.1.Aplikasi terkenal yang diubah ke dalam sistem operasiAndroid adalah *Shazam*, *Backgrounds*, dan *WeatherBug*. Sistemoperasi Android dalam situs Internet juga dianggap penting untukmenciptakan aplikasi Android asli, contohnya oleh *MySpace* dan*Facebook*.

5. Android versi 2.2 (*Froyo: Frozen Yoghurt*)

   Pada 20 Mei 2010, Android versi 2.2 (*Froyo*) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnyaantara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi2 sampai 5 kali lebih cepat, intergrasi V8 *Java*Script engine yangdipakai Google Chrome yang mempercepat kemampuan renderingpada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFiHotspot portabel, dan kemampuan auto update dalam aplikasiAndroid Market.

6. Android versi 2.3 (*Gingerbread*)

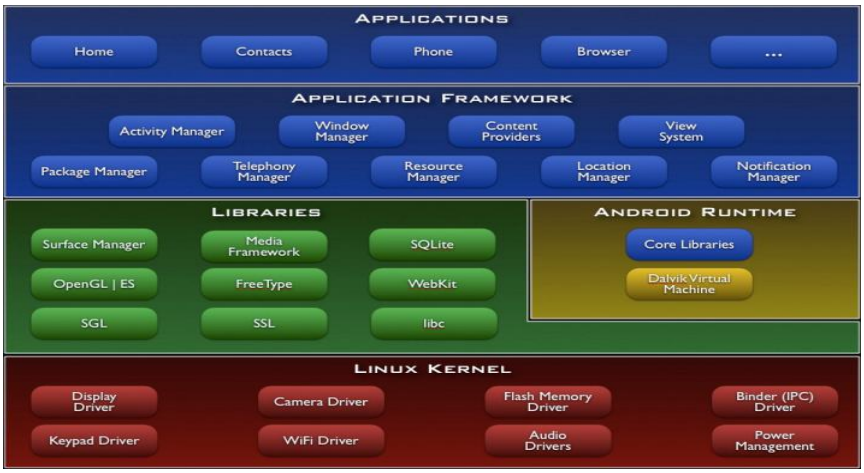
   Pada 6 Desember 2010, Android versi 2.3 (*Gingerbread*) diluncurkan. Perubahan-perubahan umum yang didapat dari Androidversi ini antara lain peningkatan kemampuan permainan (gaming),peningkatan fungsi *copy paste*, layar antar muka (*User Interface*)didesain ulang, dukungan format *video* VP8 dan WebM, efek audiobaru (*reverb, equalization, headphone virtualization*, dan *bass boost*),dukungan kemampuan *Near Field Communication* (NFC), dandukungan jumlah kamera yang lebih dari satu.

7. Android versi 3.0 (*Honeycomb*)

   Android*Honeycomb* dirancang khusus untuk tablet.Android versi ini mendukung ukuran layar yang lebih besar. *User nterface* pada *Honeycomb* juga berbeda karena sudah didesain untuktablet. *Honeycomb* juga mendukung multi prosesor dan jugaakselerasi perangkat keras (*hardware*) untuk grafis.

### 2.11.3 ArsitekturAndroid

Berikut ini menunjukkan komponen utama dari sistem operasi Android.




**Gambar 2.13Arsitektur Platform Android**

1. Linux Kernel

   Linux Kernal adalah layer dimana inti dari operating sistem dari Android itu berada. Berisi file – file system yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem –sistem operasi Android lainnya. Linux kernel yang digunakan Android adalah linux kernel release 2.6.

2. *Libraries*

   *Libraries* ini adalah layer dimana fitur – fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan diatas kernel, Layer ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta :

   a. *Libraries* media untuk pemutaran media audio dan *video*
   b. *Libraries* untuk manajemen tampilan
   c. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
   d. *Libraries* SQLite untuk dukungan *database*
   e. *Libraries* SSL dan Webkit terintegrasi dengan web browser dan security
   f. *Libraries* LiveWebcore mencakup modern web browser dengan engine embedded web view
   g. *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's.

3. Android*Runtime*

   Android terdiri dari satu *set core libraries* yang menyediakan sebagian besar fungsi yang sama dengan yang terdapat pada *core libraries* bahasa pemrograman *Java*. Setiap aplikasi menjalankan prosesnya sendiri dalam Android, dengan masing-masing instan dari mesin virtual Dalvik (Dalvik VM). Dalvik dirancang agar perangkat dapat menjalankan multiple VMs secara efisien. Mesin Virtual Dalvik mengeksekusi file dalam Dalvik executable (.dex), sebuah format yang dioptimalkan untuk memori yang kecil. Dalvik VM berbasis, berjalan, dan dikompilasi oleh compiler bahasa *Java* yang telah ditransformasikan ke dalam .dex format oleh tool "dx" yang telah disertakan. Dalvik VM bergantung pada kernel Linux untuk berfungsi seperti threading dan manajemen memori tingkat rendahnya.

4. *Framework* Aplikasi

   Pengembang memiliki akses penuh menuju API *framework* yang sama dengan yang digunakan oleh aplikasi inti. Arsitektur aplikasi dirancang agar komponen dapat digunakan kembali *(reuse)* dengan mudah. Setiap aplikasi dapat memanfaatkan kemampuan ini (sesuai dengan batasan keamanan yang didefinisikan oleh *framework*). Mekanisme yang sama memungkinkan komponen untuk diganti oleh pengguna. Semua aplikasi merupakan rangkaian set layanan dan sistem, termasuk:

   a. *Views* yang kaya dan *extensible* yang dapat digunakan untuk membangun aplikasi, termasuk *list,grids*, kotak teks, tombol, dan bahkan sebuah *embedded web*.
   b. *Content Provider* yang memungkinkan aplikasi untuk mengakses data (seperti dari daftar kontak telepon) atau membagi data tersebut.
   c. *Resource Manager,* yang menyediakan akses ke kode nonsumber lokal seperti string, gambar, dan tata letak file.
   d. *Notifications Manager* yang memungkinkan semua aplikasi menampilkan *alert* yang bisa dikostumisasi di dalam status bar.

e. *Activity Manager*yang mengelola siklus hidup aplikasi dan menyediakan navigasi umum *backstack*.

5. *Applications*

Android telah menyertakan aplikasi inti seperti email *client*, SMS, kalender, peta, *browser*, kontak, dan lainnya. Semua aplikasi tersebut ditulis dengan menggunakan bahasa pemrograman *Java*. Pada layer inilah *developer* atau kita menempatkan aplikasi yang dibuat. Yang istimewa adalah pada Android semua aplikasi baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan pada layer aplikasi dengan menggunakan *library* API yang sama. Ini berarti semua aplikasi yang dibuat untuk Android akan memiliki akses yang setara dalam mengakses seluruh kemampuan handset, tanpa membedakan apakah itu merupakan aplikasi inti atau aplikasi pihak ketiga. Dalam kata lain dengan platform Android ini, *Programmer* atau *Developer* secara penuh akan bisa mengkostumisasi perangkat Androidnya.

## 2.12 Java

*Java* adalah bahasa pemrograman yang dikembangkan oleh Sun Microsystems dan diterbitkan pada tahun 1995, *Java* merupakan bahasa pemrograman yang dapat di terapkan pada berbagai komputer maupun telepon genggam. Bahasa pemrograman *Java* pada awalnya lahir dari The Green Project.*Java* tidak boleh dianggap sebagai*JavaScript.JavaScript*adalah bahasa*scripting*yangdigunakan oleh*web browser*.

### 2.12.1 SejarahJava

Bahasa pemrograman *Java* pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak.Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta Sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang. Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas

1992 proyek ini ditutup dengan menghasilkan sebuah program *Java* Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (Star Seven).

Setelah era Star Seven selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek *The Green Project*. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto. Perusahaan baru ini bertambah maju, jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang.Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka.

Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer. Mereka menjadikan perambah *(browser*) Mosaic sebagai landasan awal untuk membuat perambah *Java* pertama yang dinamai *Web Runner*, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot *Java*.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber *Java* versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar San Jose Mercury News pada tanggal 23 Mei 1995. Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape. Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangankerja "bapak *Java*", James Gosling. Nama Oak ini tidak dipakai untuk versi release *Java* karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "*Java*".
Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman *Java* tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah *Java*).

### 2.12.2 KelebihanJava

1. *Multiplatform*. Kelebihan utama dari *Java* ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja.
2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di *Java* adalah Objek. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis *Java* secara cepat, tepat, mudah dan terorganisir.
3. Perpustakaan Kelas yang Lengkap, *Java* terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman *Java*) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya.
4. Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke *Java*.
5. Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).
6. *Java* dirancang untuk menciptakan program jaringan yang interaktif.

### 2.12.3 KekuranganJava

1. Tulis sekali, perbaiki di mana saja - Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.
2. Penggunaanmemoriyangbanyak.Penggunaanmemoriuntuk programberbasis*Java* jauhlebihbesar daripadabahasa tingkat tinggi generasisebelumnya seperti C/C++ dan Pascal(lebih spesifik lagi,Delphi dan *Object* Pascal). Biasanya inibukan merupakan masalahbagi pihak yangmenggunakan teknologi

terbaru(karena*trend*memoriterpasang makinmurah).

3. *Java* merupakan bahasa yang kompleks dan susah di pelajari, serta program yang di bangun oleh bahasa ini lebih lambat di banding program yang dibuat oleh bahasa lain seperti C dan C++.

## 2.13 Aplication Programming Interface (API)

Antarmuka pemrograman aplikasi (*Application Programming Interface*) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh pengembang saat membangung perangkat lunak untuk sistem operasi tertentu. API memungkinkan pengembang untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

## 2.14 Tools Implementasi

Tools implemtasi merupakan sebuah tools yang di gunakan sebagai media apa saja yang di butuhkan dalam membangun perangkat lunak. Androidadalah JDK sebagaiperangkatpembangun*Java*,EclipseIDEsebagai lingkunganpengembanganperangkatlunak

### 2.14.1 Eclipse

*Eclipse* adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform-independent*). Berikut ini adalah sifat dari *Eclipse*:

1. *Multi-platform*: Target sistem operasi *Eclipse* adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. *Mulit-language*: *Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya. *Eclipse* pada saat ini

merupakan salah satu IDE favorit dikarenakan gratis dan open *source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan plug-in.

### 2.14.2 Sejarah Eclipse

*Eclipse* awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM Visual Age for *Java* 4.0. Produk ini diluncurkan oleh IBM pada tanggal 5 November 2001, yang menginvestasikan sebanyak US$ 40 juta untuk pengembangannya. Semenjak itu konsursium *Eclipse Foundation* mengambil alih untuk pengembangan Eclipse lebih lanjut dan pengaturan organisasinya.

### 2.14.3 Arsitektur Eclipse

Sejak versi 3.0, *Eclipse* pada dasarnya merupakan sebuah kernel, yang mengangkat plug-in. Apa yang dapat digunakan di dalam *Eclipse* sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform (RCP)*. Berikut ini adalah komponen yang membentuk RCP:

1. *Core platform*
2. OSGi
3. SWT (*Standard Widget Toolkit*)
4. J*Face*
5. *Eclipse Workbench*

Secara standar *Eclipse* selalu dilengkapi dengan JDT (*Java Development Tools*), plug-in yang membuat *Eclipse* kompatibel untuk mengembangkan program *Java*, dan PDE (*Plug-in Development Environment*) untuk mengembangkan plug-in baru. *Eclipse* beserta plug-in-nya diimplementasikan dalam bahasa pemrograman *Java*. Konsep *Eclipse* adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, *Eclipse* tidak saja untuk mengembangkan program

*Java*, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal plug-in yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat plug-in CDT (C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, plug-in UML 2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat plug-in sesuai dengan keinginannya.

### 2.14.4 Software Development Kit (SDK)

*Software Development Kit* (*SDK*) adalah suatu kit atau *library* dari bahasa pemrograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya *SDK* terdiri dari kumpulan tools yang dibutuhkan. Misalnya bahasa pemrograman *Java*, mempunyai *SDK* yang berisi suatu *library* yang dapat digunakan untuk membuat suatu aplikasi berbasis *Java*.

### 2.14.5 JavaDevelopment Kit (JDK)

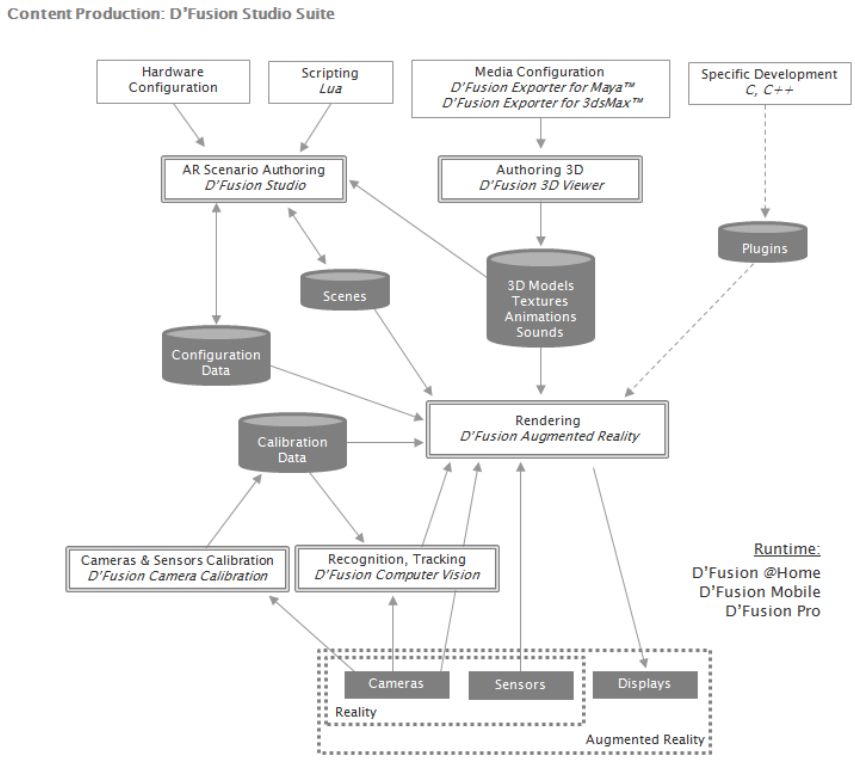
*Java Development Kit* (JDK) adalah sekumpulan perangkat lunak yang dapat digunakan untuk mengembangkan perangkat lunak yang berbasis *Java*, Sedangkan JRE adalah sebuah implementasi dari *Java Virtual Machine* yang benar-benar digunakan untuk menjalankan program *Java*. Biasanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber kompiler *Java*, *bundling, debuggers, development libraries* dan lain sebagainya. Perbedaan JDK dengan SDK (*Software Development Kit*) yaitu JDK adalah sebuah SDK tetapi sebuah SDK tidak harus menjadi sebuah JDK.

### 2.14.6 D’fusion Studio

*D'Fusion Augmented Reality* adalah mesin *Augmented Reality* yang mampu melakukan semua perhitungan yang diperlukan AR dan penanganan setiap skenario AR ketika memproses aplikasi. *D'Fusion StudioSuite* bergantung pada mesin *Fusion* D'AR. Semua fitur dari *D'Fusion* AR demikian diintegrasikan ke dalam paket *Fusion D'Studio Suite*:

1. Sebuah alat kalibrasi (*D'FusionCamera Calibration*) untuk mengkalibrasi kamera dan sensor.

2. Sebuah alat authoring tertanam di Autodesk Maya dan 3ds Max (D'Fusion Eksportir untuk Maya, D'Fusion Eksportir untuk 3ds Max dan D'Fusion 3D Viewer) untuk menghasilkan konten 3D, preview, dan ekspor untuk D'Fusion's *real-time* mesin.
3. Sebuah alat authoring (D'Fusion Studio) bertujuan untuk merancang skenario yang telah di buat, dan menentukan perilaku dan interaksi yang dapat dikontrol dengan menggunakan bahasa scripting Lua.
4. Sebuah mesin skenario (D'Fusion @ Home, D'Fusion Mobile) untuk mengelola secara real-time.
5. Sebuah mesin fisika (berdasarkan pada mesin Bullet) untuk meningkatkan interaksi alami dan rendering.
6. Sebuah mesin rendering (didasarkan pada mesin Ogre3d) untuk menggabungkan dengan halus objek yang dihasilkan komputer dengan dunia nyata melalui *video* streaming real-time.
7. Sebuah *debugger* untuk memastikan kelancaran proses pembangunan.
8. Sebuah kerangka kerja terbuka, untuk dengan mudah menambahkan sendiri plug-in dan sesuai dengan kebutuhan spesifik user.




**Gambar 2.14Konten Produk scenario**

*D'Fusion D'Computer Vision* modul mengekstrak informasi yang berguna secara*real-time*video stream seperti:

1. Pelacakan target 2D
2. 6D pelacakan, target 2D dan multiple3D
3. Deteksi pelacakan wajah.

*D'Fusion Computer Vision* dapat digunakan dalam dua cara:

1. Sebagai modul mandiri: *output* yang dikirim melalui jaringan (UDP) untuk perangkat lunak pihak ketiga yang membutuhkan informasi real-time pelacakan yang akurat (Brainstorm ®, Vizrt ®).
2. Sebagai sebuah plugin: output yang akan secara langsung dikirim ke modul Fusion D'*Augmented Reality* yang berjalan dengan Computer Vision Fusion D'ARsebagai sensor posisi standar.

Salah satu D'Komputer Visi Fusion poin terkuat adalah kemampuannya untuk mengenali dan melacak fitur alami menghindari penggunaan penanda standar relevan. Semua fitur D'Computer Fusion Vision telah terintegrasi ke D'Fusion Studio Suite dan dapat digunakan langsung melalui Studio D'Fusion.

### 2.14.7 D'Fusion Mobile

D'Fusion*Mobile* adalah pengembangan *toolkit* tersedia dengan D'Fusion ® *Studio Suite*.D'Fusion *Mobile* akan memungkinkan untuk membuat aplikasi *Augmented Reality*yang dapat digunakan pada platform mobile. D'Fusion ® *Studio Suite* juga mencakup D'Fusion *Studio*, perangkat lunak yang akan memungkinkan untuk merancang dan mengembangkan skenario realitas yang ditambahkan. AR D'Fusion Mobile menyediakan semua yang diperlukan untuk membuat aplikasi AR untuk *platform mobile*. *Platform* AR ini terdiri dari dua SDK:

1. D'Fusion Mobile SDK untuk platform iOS (iPhone, iPod touch dan perangkat iPad dengan kamera).
2. D'Fusion Mobile SDK untuk Android platform.

Untuk mengembangkan aplikasi mobile AR, akan memerlukan lingkungan pengembangan standar khusus untuk*Platform* yang ditargetkan pada sistem operasi mobile (misalnya, dapat menggunakan Xcode untuk iOS dan Android untuk Eclipse). Di samping platform-spesifik SDK di atas, D'Fusion Mobile juga termasuk alat tiPotectorAR2 (untuk menghasilkan Tombol Software Test untuk data spesifik) dan alat *PrepareFolderForUpdate* (untuk membuat data paket).

### 2.14.8 D'Fusion Mobile SDK ForAndroid 3.25

D'Fusion *Mobile* SDK untuk Android dirancang untuk pelacakanberkuantitas tinggi pada ponsel pintar atau tablet yang berjalan dengan sistem operasi Android. Algoritma pelacakan baru yang dioptimalkan dan telah dikembangkan untuk berjalan pada ponsel pintar maupun tablet. Selain itu terdapat mesin rendering 3D yang canggih dan layanan berbasis lokasi yang

tersedia untuk mengembangkan aplikasi *Augmented Reality immersive mobile*.Adapun paket yang di sediakan untuk fasilitas *library* (*LibTi*Android*AR2)* agar aplikasi lebih mudah untuk di jalankan dengan memanggil fungsi-fungsi yang terdapat didalamnya.

Dengan *Libraries*memungkinkan pengembang aplikasi Android untuk mengimplementasikan teknologi *Augmented Reaklity* agar dapat dengan mudah untuk dikembangkan, adapun beberapa *method* yang terkandung didalamkelas-kelas pada *Libraries tilib*Android*AR2*yang digunakan.

**Tabel 2.1 librari *Class***

| No | *Class* | *Method* |
|---|---|---|
| 1 | Camera | • applyCameraModel(bool iActive)<br>• setCameraModel()<br>• getCameraModel()<br>• isUsingCameraModel()<br>• loadCameraModel() |
| 2 | Scene | • render()<br>• setCurrentCamera()<br>• getCuurrent()<br>• setShadowColour()<br>• getShadowColoru() |
| 3 | *Video*Capture | • getVidCaptureId() |
| 4 | Vector3 | • setX()<br>• getX()<br>• setY()<br>• getY()<br>• setZ()<br>• getZ() |
| 5 | VidioTexture | • set*Video*Capture()<br>• get*Video*Capture() |
| 6 | Module | • createObject()<br>• getObjectByName()<br>• getObjectListByType()<br>• deleteObject()<br>• AttachObject() |
| 7 | Object3D | • getChild()<br>• setVisible()<br>• getVisible()<br>• getPos() |

| | | |
|---|---|---|
| | | • setPosition()<br>• setOrientation() |
| 8 | Sound | • setLoop()<br>• getLoop()<br>• play()<br>• stop()<br>• isPlaying() |
| 9 | MltPluginManager() | • startTracking()<br>• stopTracking()<br>• getTargetStatus()<br>• getTargetPos()<br>• getRecognizedKeyFrameindex() |
| 10 | quaternion | • set()<br>• get()<br>• setX()<br>• getX()<br>• setY()<br>• getY()<br>• setZ()<br>• getZ()<br>• setW()<br>• getW() |
| 11 | Binding global | • getCurrentScene()<br>• getMLTPluginManager() |
| 12 | Component Interface | • pullCommand()<br>• executeAppFunc() |
| 13 | Animation | • setLoop(bool loop)<br>• getLoop()<br>• Set Step(double step)<br>• getStep |
| 14 | viewport | • setCamera()<br>• getRenderTarget() |

Deskripsi *class- class* yang berkaitan :

1. *Class Camera*

**Table 2.2 Tabel *Class Camera***

| *Method* | *Input* | Deskripsi |
|---|---|---|
| ApplyCameraModel(bool iActive) | Bool iActive | Set apakah kamera mengikuti model yang di definisikan oleh setCameraModel |

| | | |
|---|---|---|
| setCameraModel(int Model,int maxx,int maxy, double pp_*video*x,double pp_vieoy, double focalx,double focal, double a1, double a2, double p1, double p2, bool dedistortion) | int Model,int maxx,int maxy, double pp_*video*x,double pp_vieoy, double focalx,double focal, double a1, double a2, double p1, double p2, bool dedistortion | Set model kamera |
| getCameraModel() | - | Kembalian nilai parameter model kamera |
| isUsingCameraModel() | - | Mengembalikan nilai apakah kamera mengikuti model yang didefinisikan oleh setCameraModel |
| loadCameraModel(string iFileName, bool iForceActivate) | string iFileName, bool iForceActivate | Set parameter distorsi kamera dari sebuah file |

Fungsi dokumentasi :

a. Camera::applyCameraModel( bool iActive )

   Set apakah kamera mengikuti model yang di definisikan oleh setCameraModel

   Parameter :

   iActive :benar jika kamera cocok dengan kamera CameraBase model, salah jika sebaliknya.

   Return :

   eOk

b. Bool Camera::isUsingCameraModel()

   Return : true jika camera mengikuti CameraBase model, false jika sebaliknya.

c. Camera::setCameraModel(int Model,int maxx,int maxy, double pp_*video*x,double pp_vieoy, double focalx,double focal, double a1, double a2, double p1, double p2, bool dedistortion )

Set camera model

Parameter :

model : camera Model

Maxx :kalibrasi lebar *video*

Maxy:kalibrasi*video* y

Pp_*video*x: tengah koordinat optik

Pp_vidioy: tengah koordinat optic

Focalex: focal pada axis x

Focaley: focal pada axis y

A1: a1 , a2:a2, p1: p1, p2: p2

Return :

eOk

d) Camera::getCameraModel()

Mengembalikan parameter model kamera.

Return:

: camera Model

:kalibrasi lebar *video*

:kalibrasi*video* y

Pp_*video*x: tengah koordinat optik

Pp_vidioy: tengah koordinat optic

Focalex: focal pada axis x

Focaley: focal pada axis y

: a1

:a2

: p1

: p2

e) Camera::loadCameraModel(string iFileName, bool iForceActivate)

Set parameter distorsi kamera dari sebuah file

Parameter:

iFileName: file penjelasan mengenai parameter distorsi .

iForceActivate: jika benar, model yang di ambil langsung dijalankan.

2. *Class Scene*

Tabel 2.3 Tabel *Class Scene*

| *Method* | *Input* | Deskripsi |
|---|---|---|
| Render() | - | Mengaktifkan rendering scenario utama |
| setCurrentCamera(Camera Cam) | Camera cam | Set cam sebagai kamera baru pada scenario utama |
| getCurrentCamera() | - | Mengambil scenario kamera yang memungkinkan ada beberapa kamera yang aktif dalam scenario terutama ketika menggunakan beberapa viewport atau target rendering. |
| setShadowColour(float r,float g, float b) | float r,float g, float b | Set warna yang digunakan untuk mengatur wilayah di dalam bayangan. |
| getShadowColour() | - | Mendapatkan nilai warna yang digunakan didalam wilayah bayangan. |
| Capture(VideoCapture capture) | VideoCApture Capture | Digunakan untuk Mengambil gmbar |

Fungsi dokumentasi :

a) Scene::getShadowColor( )

Mendapatkan warna yang digunakan untuk modulasi pada area bayangan.

Return ; red komponen dari komponen bayangan

blue komponen dari komponen bayangan

green komponen dari komponen bayangan

b) Scene::render()

Render scenario

Return: eOk jika berhasil

c) Scene:: setShadowShadowColour(float r,float g, float b)

Parameter :

r : komponen warna merah pada bayangan

g : komponen warna biru pada bayangan

b: komponen warna hijau pada bayangan

d) Scene::getCurrentCamera()

Mengambil scenario kamera dari beberapa kamera yang aktif dalam scenario, terutama ketika menggunakan beberapa viewports atau target render.

Maasih ada satu kamera utama yang merupakan kamera yang dipasangkan pada viewport utama.

Return :

Kamera scenario yang sedang digunakan.

e) Scene :: Capture(videoCapture Capture)

Digunakan untuk mengambil gambar.

Parameter :

Capture : videoCapture

3. *Class VideoCapture*

Table 2.4 Tabel *Class VideoCapture*

| *Method* | *input* | Deskripsi |
|---|---|---|
| getVidCapId() | - | Mendapatkan id yang digunakan oleh dideo capture objek. |

Fungsi dokumentasi :

a) *Video*Capture::getVidCapID()

Mendapatkan id yang digunakan untuk *video* capture objek

Return :

Capture id

4. *Class Vector3*

Table 2.5 Tabel *Class Vector3*

| *Method* | *input* | Deskripsi |
|---|---|---|
| setX(double x) | Double x | Set vector komponen x |
| getX() | - | Mendapatkan nilai komponen nilai vector x |
| setY(double y) | double y | Set vector komponen y |
| getY() | - | Mendapatkan nilai komponen nilai vector y |
| SetZ(double Z) | Double Z | Set vector komponen z |
| getZ() | - | Mendapatkan nilai komponen nilai vector z |

Fungsi dokumentasi :

a) Vector::getX()

Mendapatkan komponen vector x

Return : komponen vector X

b) Vector::getY()

Mendapatkan komponen vector Y

Return : komponen vector Y

c) Vector::getZ()

Mendapatkan komponen vector z

Return : komponen vector Z

d) Vector::setX(doublex)

Set quaternion komponenx

Parameter :

x : x komponen quaternion

e) setY(double y)

set quaternion komponen Y

parameter :

y :Y komponen quaternion.

f) setW(double w)

set quaternion komponen W

parameter :

w :W komponen quaternion.

g) setZ(double z)

set quaternion komponen Z

parameter :

z :Z komponen quaternion.

5. *Class VidioTexture*

Table 2.6 Tabel *Class VidioTexture*

| *Method* | *input* | Deskripsi |
|---|---|---|
| setVidioCapture(*video*Capture i*Video*CaptureObject) | *video*Capture i*Video*Capture Object | Mengambil tekstur menjadi *VideoCapture Objek* baru yang hanya akan selesai untuk diinisialisasi *Video*Texture objek, |
| get*Video*Capture() | - | Mendapatkan *VideoCapture Objek* yang teksturnya terpasang. |

Fungsi dokumentasi :

a) VieoTexture::set*Video*Capture()

Melampirkan tekstur kepada VodeoCapture baru yang hanya dapat dilakukan untuk menginisialisasikan *Video*Texture objek atau merubah *video* input penangkap baru jika objek capture kompatibel dengan yang sebeleumnya( dimensi yang sama sehingga tekstur tidak dibuat ulang).

Return : eOk jika berhasil

b) *Video*Capture::get*Video*Capture()

Mendapatkan *Video*Capture object untuk teksturnya dipasang.

6. *Class Module*

Table 2.7 Tabel *Class Module*

| *Method* | *input* | Deskripsi |
|---|---|---|
| createObject(int cid) | Int cid | Membuat sebuah objek dari class yang diberikan sebagai parameter dan menambahkannya ke modul. |
| getObjectByName(string name) | String name | Mendapatkan objek dengan nama objek. |
| getObjectListByType(int cid, bool includeChildClasses=false) | int cid, bool includeChildClasses=false | Mengembalikan daftar semua objek dari tipe yang diberikan |
| deketeObjct(object obj, bool freeResources=true) | object obj, bool freeResources=true | Delete sebuah objek |
| AttachObject(object obj) | object obj | Memasukan sebuah objek ke urutan module |

Fungsi dokumentasi :

a) Module::attachObject (object obj)

Tambahkan sebuah objek pada modul hirarki

Return:

obj: objek yang untuk dilampirkan.

b) Module::createObject(int cid)

Membuat sebuah objek dari klas yang diberikan sebagai parameter dan menambahkan terhadap modulnya.

Parameter :

cid : CLASSID dari objek yang akan dibuat

Return :

objek baru yang telah dibuat.

c) Module::deleteObject(object obj, bool freeResource=true)

Menghapus sebuah objek.

Parameter :

obj : pointer objek untuk dihapus

freeResource: jika di set true, sumber yang terhubung akan di*load.*

Return :

eOk jika berhasil, eInvalidParameter jika obj bukan sebuah objek yang terdapat pada modul.

d) Module::deleteObjectList(table list)

Menghapus daftar objek

Parameter :

List : table yang terhung dengan objek yang akan dihapus.

Return :

eOk jika berhasil.

e) Module::getObjectByName(string name)

Mendapatkan objek dengan nama.

Parameter :

Name : nama objek untuk dipanggil.

Return :

Nama objek yang diberikan. Jika nama objek tidak ditemukan , hasil objek harus di uji dengan isNull().

f) Modul::getObjectListByType(int cid, boolincludeChildClasses = false)

Mengembalikan daftar semua objek yang diberikan.

Parameter :

cid : Class ID dari objek untuk pengembalian (Class IDs)

includeChildClasses: menyertakan objek yang berasal dari pengkelasan.

7. *Class object3D*

Table 2.8 Tabel *Class Object3D*

| *Method* | *input* | Deskripsi |
|---|---|---|
| getChild(int index) | Int index | Mendapatkan kembali nilai child objek |
| setVisible(bool visible) | bool visible | Set agar objek terlihat |
| getVisible() | - | Mengembalikan nilai objek dapat terlihat. |
| GetPose(Vector3 position, quaternion orient object3D ref=Null) | Vector3 position, object3D ref=Null | Mendapatkan posisi objek dan orientasi yang di berikan |
| setPosition(Vector3 pos, object3D ref=NULL) | Vector3 pos, object3D ref=NULL | Set posisi baru untuk sebuah objek |
| setOrientation(Quaternion orient, object3D ref=NULL) | Quaternion orient, object3D ref=NULL | Set orientasi baru terhadap objek. |

Fungsi dokumentasi :

a) Object3D::getChild(int index)

Parameter :

Index : rank child untuk kembalian.

Return:

i-th child dari node, null jika index tidak terkait.

b) Object3D::getVisible()

Pengembalian langkah dari objek agar terlihat.

Return :

Visible : pemilihan nilai yang diisi dengan keadaan yang dapat terlihat.

c) Object3D:: getPose(Vector3 position, quaternion orient object3D ref=Null)

Mendapatkan posisi objek dan orientasi yang di berikan

Parameter:

Position ; (output)posisi objek

Orient : (output) orientasi objek

Ref : referensi objek (posisi, orientasi dan skala ) di posisi yang telah diperhitungkan, jika referensi Object3D adalah Null, posisi akan dikembalikan kepada koordinat dunia nyata.

d) Object3D::setPosition(Vector3 pos, object3D ref=NULL)

Set posisi baru untuk sebuah objek.

Parameter:

Pos : posisi baru untuk di set.

Ref : objek referensial(posisi, orientasi dan skala) dimana posisi diberikan. Jika referensial Object3D adalah NULL, pos dianggap sebagai posisi koordinat dunia nyata.

e) Object3D::setOrientation(Quaternion orient, object3D ref=NULL)

Set orientasi baru terhadap objek.

Parameter:

Orient : orientasi baru yang diberikan sebagai referensial.

Ref : objek referensial(posisi, orientasi dan skala) dimana posisi diberikan. Jika referensial Object3D adalah NULL, pos dianggap sebagai posisi koordinat dunia nyata.

8. *Class Sound*

Table 2.9 Tabel *Class Sound*

| *Method* | *input* | Deskripsi |
|---|---|---|
| setLoop( bool loop) | Bool loop | Spesifikasi jika suara harus di ulang |
| getLoop() | - | Mendapatkan nilai pengulangan objek |

| | | |
|---|---|---|
| | | suara |
| Play() | - | Memulai menjalankan suara. |
| Stop() | - | Menghentikan suara. |
| isPlaying() | - | Mengambil bagian jeda dari objhek suara. |

Fungsi dokumentasi :

a) Sound::play()

Memulai menjalankan suara

Perintah ini akan selalu membuat suara dimulai dari awal. Jika suara sudah berjalan, maka akan diulang kembali ke awal.

Return:

eOk jika berhasil.

eNotInitialize jika objek suara tidak benar-benar di inisialisasi ( tidak ada file suara yang dimuat sebagai contoh).

b) Sound::stop()

Menghentikan pemutaran suara.

Return :

eOk : jika berhasil

eNotInitialize jika objek suara tidak benar-benar di inisialisasi ( tidak ada file suara yang dimuat sebagai contoh).

eInvalidParameter jika suara tidak dapat dihentikan (suara telah berjalan atau dipaus).

c) Sound::isPlaying()

Mengambil bagian jeda dari objhek suara.

Return:

True jika suara benar-benar valid dan sedang dimainkan.False jika sebaliknya.

d) Sound::setLoop(bool loop)

Spesifikasi jika suara harus di ulang

Parameter :

Loop : suara akan di diulang jika parameter true.

Return :

eOk jika berhasil, eNotInitialize jika suara benar-benar tidak valid.

e) Sound::getLoop()

Mendapatkan nilai pengulangan objek suara

Return :

True jika suara di set pengulangan, false jikla sebaliknya.

9. *Class MLTPluginManager*

Table 2.10 Tabel *Class MLTPluginManager*

| *Method* | *input* | Deskripsi |
|---|---|---|
| Start tracking(string iFileName, int iVidCaptureIndex,Camera iCamera, int requiredMode) | string iFileName, int iVidCaptureIndex,Camera iCamera, int requiredMode | Memulai sebuah pelacakan MLT |
| Stop tracking(int TrackingIndex) | int TrackingIndex | Stop pelacakan MLT |
| getTargetStatus(int iTrackingIndex, int iTargetIndex) | int iTrackingIndex, int iTargetIndex | Mendapatkan status nilai pelacakan target . |
| getTargetPos(int iTrackingIndex,int itargetIndex, vector position, Quartenion orientation) | int iTrackingIndex,int itargetIndex, vector position, Quartenion orientation | Mendapatkan posisi sebuah target yang terlacak. |
| getRecognizedKeyFrameIndex(int iTrackingIndex,int iTargetIndex) | int iTrackingIndex,int iTargetIndex | Mendapatkan hasil untuk mengenali objek yang menjadi target *tracking*. |

Fungsi dokumentasi :

a) MLTPluginManager::startTracking(string iFileName, int iVidCaptureIndex,Camera iCamera, int requiredMode)

   Memulai Pelacakan MLT

   Parameter :

   iFileName : Pelacakan MLT file konfigurasi. Jalur harus di disandikan sebagai UTF-8 String.

   iVidCapIndex : indeks dari *video* Kaptur harus digunakan oleh MLT.

   iCamera : kamera yang dijadikan sebagai kamera yang akan digunakan oleh MLT.

   requiredMode (hanya iPhone dan Android) : tread mede yang akan digunakan oleh MLT, MLT_AUTOMATIC(default), MLT_SYNCRONEUS atau MLT_THREADED.

   Return:

   :eOk jika berhasil.

   Index pelacakan dibuat.

b) MltPluginManager::getTargetStatus(int iTrackingIndex, int iTargetIndex)

   Mendapatkan status pelacakan sebuah target.

   Parameter :

   iTrackingIndex : indeks pelacakan.

   iTargetIndex : indeks target.

   Return:

   :eOk jika berhasil.

   :status dari target (-1:eror, 0: tidak tejadi pelacakan atau dikenali, 1 objek terlacak)

c) MLTPluginManger::getRecognizedKeyFrameIndex(int iTrackingIndex,int iTargetIndex)

Mendapatkan pengenalan objek untuk dijadikan target.

Parameter :

iTrackingIndex: indeks pelacakan.

iTargetIndex: indeks target.

Return :

eOk jika berhasil.

:indeks objek dikenali.

d) MLTPluginManager::getTargetPos(int iTrackingIndex,int itargetIndex, vector position, Quartenion orientation)

Mendapatkan posisi dari target, metode ini mengembalikan data korespondensi kepada pemanggilan terakhir terhadap ticMLTPluginManager::updateTracking

Parameter :

iTracking : indeks pelacakan.

iTargetIndex : indeks target.

: (output) spesifikasi posisi target didalam kamera.

: (output) orientasi dari spesifikasi target didalam kamera.

10. *Class Quaternion*

Table 2.11 Tabel *Class Quaternion*

| *Method* | *input* | Deskripsi |
|---|---|---|
| set(double x, double y, double z,double w) | double x, double y, double z,double w | Set nilai quaternion |
| Get() | - | Mendapatkan nilai komponen quaternion |
| setX(double x) | Double x | Set nilai quaternion komponen x |
| getX() | - | Mendapatkan nilai quaternion komponen x |
| setY(double y) | Double y | Set nilai quaternion |

| | | |
|---|---|---|
| | | komponen y |
| getY() | - | Mendapatkan nilai quaternion komponen y |
| setZ(double z) | Double z | Set nilai quaternion komponen x |
| getZ() | - | Mendapatkan nilai quaternion komponen y |
| setW(double w) | Double w | Set nilai quaternion komponen w |
| getW() | - | Mendapatkan nilai quaternion komponen w |

Fungsi dokumentasi :

a) Quaternion::getW()

Mendapatkan komponen quaternion W

Return:

Komponen quaternion W

b) Quaternion::getW()

Mendapatkan komponen quaternion W

Return:

Komponen quaternion W

c) Quaternion::getX()

Mendapatkan komponen quaternion X

Return:

Komponen quaternion X

d) Quaternion::getY()

Mendapatkan komponen quaternion Y

Return:

Komponen quaternion Y

11. *ClassBinding global*

Tabel 2.12*Method* pada *Calss Binding global*

| *Method* | *input* | Fungsi |
|---|---|---|
| getCurrentScene() | - | Digunakan utnuk mengaktifkan scenario aplikasi. |
| getMLTPluginManager() | - | Digunakan untuk memanggil plugin MLT(CV). |

Fungsi dokumentasi :

a) Scene getCurrentScene()

Mengambil skrip yang sedang dijalankan.

Return:

Skrip yang yang sedang dijalankan.

b) MltPluginManager getPluginManager()

Mengembalikan instanc uniq dari MLT(CV) Manajer Plugin.

Return :

Instance dari MLT(CV) Manajer Plugin.

12. *Class ComponentInterface*

Table 3.13 Tabel *Class CompnentInterface*

| *Method* | *input* | Deskripsi |
|---|---|---|
| pullCommand() | - | Mengambil perintah dari beberapa kumpulan perintah. |
| executeAppFunc() | String funcName, string arg1, .., string argnn | Eksekusi fungsi aplikasi |

Fungsi dokumentasi :

a) ComponentInterface::pullComand()

Mengambil perintah dari *stack* perintah.

Return:

Benar dan jika table perintah di ambil.Salah jika sebaliknya.

Table kembalian jika benar terlihat seperti :

- Table[“CComandName”] nama dari perintah
- Table[“arg0”] argumen pertama dari sebuah perintah
- Table[“arg1”] argument kedua sebuah perintah.
- …….
- Table[“argn”] (n+1) argumen dari sebuah perintah

b) ComponentInterface::sexecuteAppFunc(String funcName, string arg1, ..,string argnn)

Eksekusi fungsi aplikasi

Parameter ;

funcName fungsi id dari aplikasi yang akan di eksekusi.

Argn nth argumen dari sebuah aplikasi fungsi, sebagai string.

Return :

Benar jika fungsi telah berhasil dieksekusi.

13. *Class Animation*

Table 3.14 Tabel *Class Animation*

| *Method* | *input* | Deskripsi |
|---|---|---|
| setLoop(bool loop) | Bool loop | Modifikasi state pengulangan |
| getLoop() | - | Mengembalikan status pengulangan. |
| Set Step(double step) | Double step | Set manual langkah dari animasi yang dapat digunakan untuk mengatur sebuah spesifikasi waktu dalam animasi manual, tanpa animasi benar-benar dijalankan. |

| | | |
|---|---|---|
| getStep | - | Mengembalikan nilai waktu dalam rentan jarak [0…1] dimana 1 berkoresponden terhadap panjang animasi. |

Fungsi dokumentasi :

a) Animation:getLoop()

Mengembalikan nilai status pengulangan.

Return:

Benar jika animasi sedang berulang.

b) Animation::getStep()

Mengembalikan nilai waktu dari animasi dalam jarak [0..1] dimana 1 korespondensi dengan panjang dari pengulangan animasi.

Jika pengulangan diset benar, langkah pada rentan [n..n+1] setelah n-th pengulangan, maka nilai dapat diindikasikan berapa banyak waktu animasi telah diulang.

Parameter ;

funcName fungsi id dari aplikasi yang akan di eksekusi.

Argn nth argumen dari sebuah aplikasi fungsi, sebagai string.

Return :

Mengembalikan nilai langkah pengulangan.

14. *Class viewport*

**Tabel 2.15 Tabel *Class viewport***

| *Method* | *input* | Fungsi |
|---|---|---|
| setCamera() | - | Digunakan utnuk kamera. |
| getRenderTarget() | - | Digunakan untuk mendapatkan target yang akan direnderer. |

Fungsi dokumentasi :

a) viewpotr:setCamera()

Digunakan untuk mendapatkan kamera untuk digunakkan..

b) getTargetRender()

Digunakan untuk mendapatkan *pointer*target yang akan direnderer.

### 2.14.9 Interface ForAndroid

#### 2.14.9.1 tiComponent declaration

Deklarasi antarmuka tiComponent untuk Android adalah sebagai berikut (perhatikan bahwa disediakan jar., Sehingga tidak ada "header" yang secara langsung terkena dan berikut ini hanya resume tiComponent kelas dan bagiannya):

Tabel 3.16 Tabel *Class tiComponent*

| *Method* | *input* | Deskripsi |
|---|---|---|
| setRendererType(final String rendererType) | String rendererType | Fungsi yang digunakan untuk set tipe rendering. |
| initialize(FrameLayout frameLayout) | FrameLayout frameLayout | Inisialisasi yang digunakan untuk mengatur framelayout pada saat keadaan awal. |
| onPause() | - | Berfungsi untuk set *scenario* agar dapat ditunda. |
| onResume() | - | Berfungsi untuk melanjutkan *scenario*. |
| terminate() | - | Berfungsi untuk menghentikan seluruh fungsi *scenario*. |
| isInitialized() | - | Berfungsi untuk mengembalikan nilai komponen apakah telah terinisialisasi. |
| loadScenario(String filename) | String filename | Berfungsi untuk mengambil data scenario Ar agar dapat |

| | | |
|---|---|---|
| | | berjalan serta mengembalikan nilai benar jika *scenario* telah di *load*. |
| pauseScenario()<br>playScenario()<br>isScenarioPaused() | - | Berfungsi untuk *play/pause scenario*. |
| enqueueCommand(String commandName, String [] args) | String commandName, String [] args | Memungkinkan untuk mengirim perintah dengan menggunakan argumen *script* Lua |
| registerCommunicationCallback(String functionId, Object object, String methodName) | String functionId, Object object, String methodName | Fungsi yang digunakan untuk mendaftarkan 'methodName' pada objek 'object' dengan menggunakan Id 'functionId'. |
| activateAutoFocusOnDownEvent(boolean activate) | boolean activate | Berfungsi untuk memfokuskan kamera pada layar. |
| doCameraFocusNow() | - | Berfungsi untuk melakukan fikus kamera. |
| getVersion() | - | Mengembalikan nilai versi dari SDK *mobile* komponen. |
| registerNotificationLicense(Object callbackObject, String callbackMethod) | Object callbackObject, String callbackMethod | Untuk mengembalikan nilai pesan lisensi yang terkait. |
| setCustomUserAppDataDirectory(String filepath) | String filepath | Menentukan UserAppDirectory secara kustom. |
| postRequest( String req, String[] args ) | String req, String[] args | Fungsi untuk pengaturan berbagai tambahan platform Android sebagai opsi khusus dari D'Fusion *mobilecomponent* |

### 2.14.10 Class FrameLayout

Faremelayout merupakan class yang berisikan fungsionalitas dalam menentukan layout yang digunakan serta mengatur berbagai fungsionalitas seperti kamera, ukuran dan posisi tataletak frame dalam penggunaannya.

```
Public class FrameLayout extends Activity{
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.frame_layout);
 }
}
```

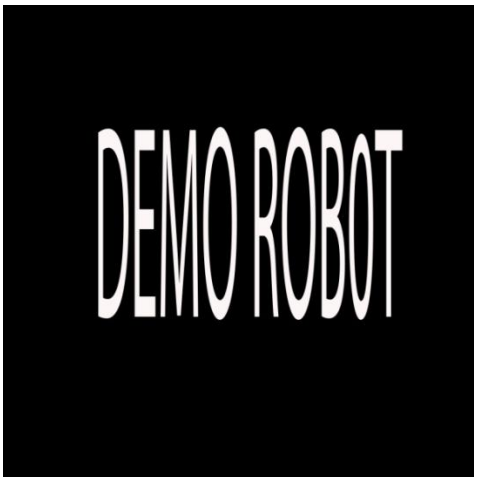
### 2.14.11 Perancangan AR

Dalam mengembangkan *Augmented Reality*dengan menggunakan *D'fusion Studio*ada beberapa material yang diperlukan agar semuanya dapat berjalan, dalam hal ini adapun material yang digunakan adalah :

1. File Gambar berupa file gambar .jpg.
2. File suara berupa file .ogg.
3. File objek 3D berupa file .scene.

#### 2.14.11.1 Perancangan Marker

Dalam tahapan ini marker yang merupakan inisialisasi awal agar fungsi *tracking* dapat dikenali, maka format file gambar harus di*convert* terlebih dahulu oleh *D'fusion Studio* , dalam tahapan ini mesin akan menyimpan file gambar *input*-an yang akan dijadikan objek marker menjadi sebuah file gambar yang akan memiliki file object.bin dimana didalamnya terdapat parameter yang digunakan untuk pencocokan pola oleh mesin *D'fusion Studio*.

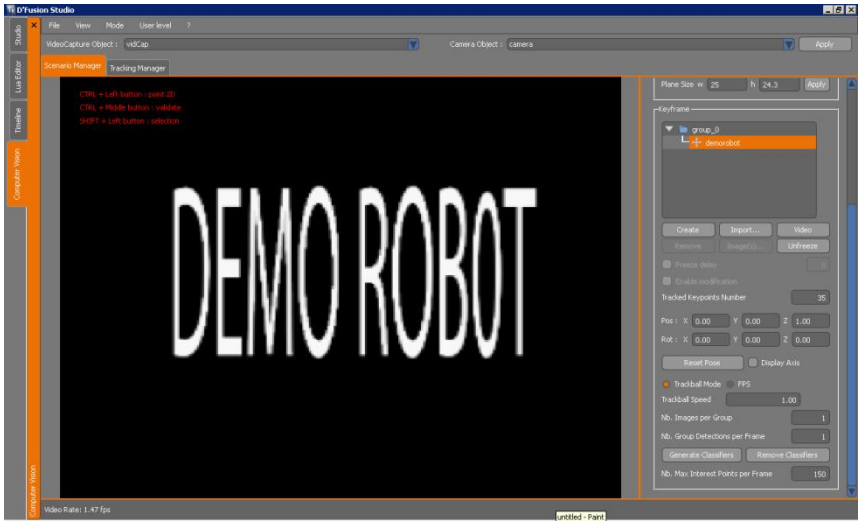



Gambar 2.15 Gambar *Input*-an Awal Marker 3D



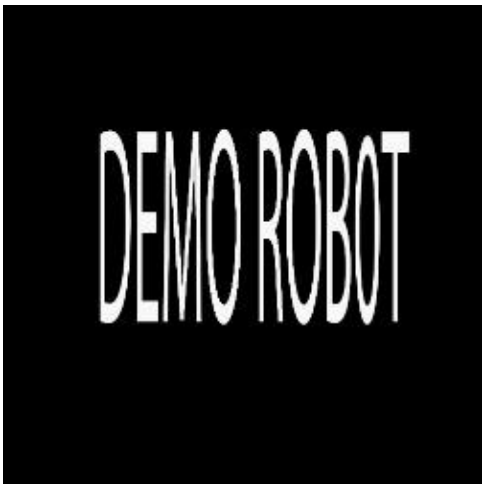

Gambar 2.16 Gambar *Input*-an Awal Marker Suara

Sebagai *input*-an awal Gambar 2.15 dan Gambar 2.16 memiliki ukuran 650x650 px yang berekstensikan file .jpg. Dalam tahapan perancangan marker, *D'fusion Studio* memiliki fungsionalitas berupa *Computer Vision* dimana fungsionalitas tersebut berfungsi agar file gambar biasa dapat dijadikan sebuah marker.




Gambar 2.17 Gambar *Computer Vision*

Setelah file gambar awal dijadikan marker, maka gambar tersebut akan menjadi sebuah gambar *grayscale* pada *database* dan memiliki file object.bin untuk pola marker yang telah disimpan tersebut didalamnya.




Gambar 2.18 Gambar Marker 3D setelah proses *generate*

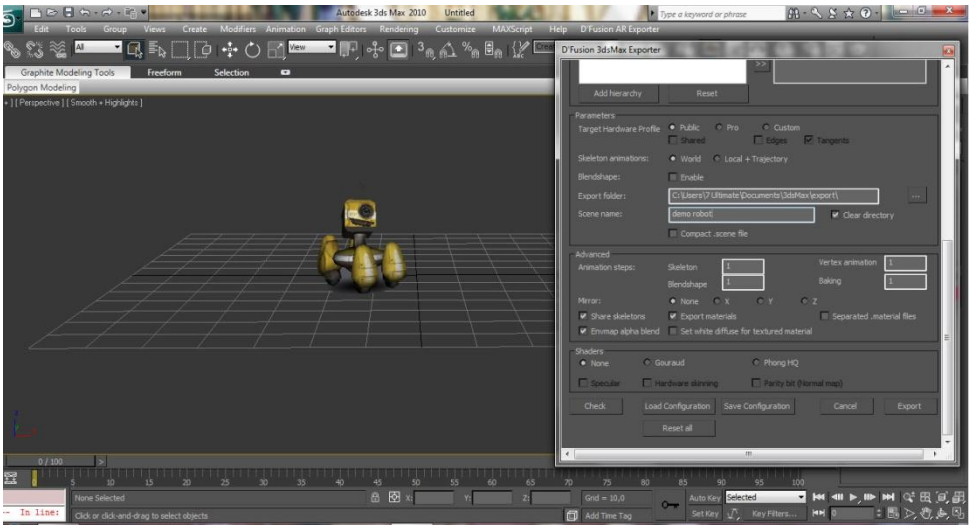



Gambar 2.19 Gambar Marker Suara setelah proses *generate*

Dari setiap marker yang telah dibuat, maka setiap marker tersebut akan disimpan pada *database* kemudian akan di *index*dari 0 samapi ke n sebanyak marker yang telah dirancang, fungsi *index* tersebut agar dapat dengan mudah menentukan objek marker mana yang sesuai pada tahap mencocokan pola yang digunakan pada LUA *script*. Ada juga terdapat file traker.xml yang berisikan parameter tekstur yang fungsinya digunakan untuk proses *tracking* objek gambar marker.

### 2.14.11.2 Perancangan Object 3D

Pada tahapan objek 3D ada beberapa tahapan yang dilakukkan agar objek 3D tersebut dapat digunakan pada aplikasi *Augmented Reality,* dikarenakan file yang digunakan harus berekstensikan format .scene. D'fusion sendiri telah menyediakan *eksporter* untuk beberapa aplikasi pembangun objek 3D seperti Maya dan 3Dsmax. Dalam perancangan objek 3D disini menggunakan *tools* 3DsMax sebagai *tools* pembangunnya, maka agar objek 3D tersebut dapat di *eksport* kedalam format .scene, 3DsMax tersebut harus memiliki *eksporter* D'fusion.



Gambar 2.20 Gambar Objek 3D

Setelah di-*eksport*, maka file format 3D tersebut akan secara langsung menjadi *object.scene.*

### 2.14.11.3 PerancanganSound

Dalam tahapan perancangan suara format file suara yang digunakan menggunakan format file .ogg. Dimana format tersebut memiliki kualitas yang baik jika digunakan dalam aplikasi *Augmented Reality* yang dibangun oleh *D'fusion Studio.*Output yang dihasilkan Dalam *tools* pembangun *D'fusion Studio* sendiri menyediakan 2 format yang dapat digunakan yaitu .ogg dan wav.

### 2.14.11.4 PerancanganFile Lua Script

Lua *Script* adalah merupakan sebuah bahasa pemrograman yang digunakan oleh AR D'fusion Studio. Didalam file tersebut terdapat perintah-perintah yang digunakan dan inisialisasi awal dari setiap objek untuk dapat digunakan dalam pencocokan pola.

```
gtrackingStatus = 0
gtrackingStatus2 = 0
gtrackingKeyFrameIndex = -1
local scene = getCurrentScene()
local camera = Camera(scene:getCurrentCamera())
local videoCapture_live = local videoCapture_live =
VideoCapture(scene:getObjectByName("vidCap"))
local trackingSuara =
Sound(scene:getObjectByName("soundRobot"))
local MLTPlugin = getMLTPluginManager()
local errorStatus = eOk
local trackingIndex = -1
local trackingRate = 0
local trackingPosition = Vector3()
local trackingOrientation = Quaternion()
local trackingObject =
Object3D(scene:getObjectByName("Tracking_Object"))
local trackingObjectIndex = 0
local trackingObjectIndex2 = 1
errorStatus, trackingIndex =
MLTPlugin:startTracking("tracking/tracker.xml",
videoCapture_live:getVidCapID(), camera)
if errorStatus == eOk then
repeat
errorStatus, gtrackingStatus =
MLTPlugin:getTargetStatus(trackingIndex, trackingObjectIndex)
errorStatus, gtrackingKeyFrameIndex =
MLTPlugin:getRecognizedKeyFrameIndex(trackingIndex,
trackingObjectIndex)
if (gtrackingStatus == 1) then
MLTPlugin:getTargetPos(trackingIndex, trackingObjectIndex,
trackingPosition, trackingOrientation)
trackingObject:setPosition(trackingPosition, camera)
trackingObject:setOrientation(trackingOrientation, camera)
if not trackingObject:getVisible() then
trackingObject:setVisible(true) end
elseif trackingObject:getVisible() then
trackingObject:setVisible(false) end
end
if (gtrackingStatus2 == 1 ) then
if not trackingSuara:isPlaying() then trackingSuara:play() end
elseif trackingSuara:isPlaying() then
trackingSuara:stop() endend
until coroutine.yield()
end
```

Didalam perintah lua *script* itu sendiri *library* yang menjalankan semua fungsi yang terdapat didalamnya. Ada beberapa parameter awal yang dijadikan sebagai inisialisasi awal agar semua perintah tersebut dapat berjalan, misalkan `gtrackingStatus=0` merupakan inisialisasi awal bahwa status dalam keadaan tidak men-*tracking*, ada dua tipe biner yang digunakan yaitu 0 dan 1 dimana 1 = *tracking*, `gtrackingKeyFrameIndex=-1` mendapatkan indeks

keyframedari objek yang ter-*tracking*, `localscene= getCurrentScene`() mengaktifkan skenario utama. `localcamera = Camera(scene:getCurrentCamera`()) mengaktifkan kamera utama pada skenario,`localvideoCapture_live=VideoCapture(scene:getObjectByName("vidCap"))` mendapatkan *video capture*yang akan digunakan dalam menangkap gambar,local `MLTPlugin=getMLTPluginManager()` fungsionalitas yang digunakan untuk *tracking*.`local errorStatus = eOk` inisialisasi status *error,* local `trackingIndex=-1` inisialisasi yang digunakan untuk memanggil tracker.xml, `local trackingPosition = Vector3()` fungsionalitas yang digunakan untuk posisi objek 3D, `local trackingOrientation=Quaternion()` fungsionalitas yang digunakan untuk 4 arah sumbu objek 3D, `local trackingObject = Object3D(scene:getObjectByName("Tracking_Object"),localtrackingSuara = Sound(scene:getObjectByName("soundRobot"))` fungsionalitas yang digunakan untuk mendapatkan bentuk 3D dan file suara, `local trackingObjectIndex=0` inisialisasi yang digunakan untuk inisialisasi nilai dari marker yang telah dibuat pada tahapan pembuatan marker dimana nilai 0 merupakan nilai indeks untuk marker pertama, `errorStatus,trackingIndex=MLTPlugin:startTracking("tracking/tracker.xml", videoCapture_live:getVidCapID(), camera)` merupakan fungsi yang digunakan untuk memulai *tracking*dengan memanggil bebrapa fungsi untuk dijalankan.

#### 2.14.11.5 PencocokanPola

Pencocokan pola merupakan suatu fungsi dimana setiap pola marker akan diperiksa oleh system aplikasi apakah pola marker tersebut ada pada *database*, jika ada maka system akan menyamakan pola tersebut kemudian menyamakan objek 3D mana atau suara yang sesuai dengan marker tersebut.

```
if (gtrackingStatus == 1) then
MLTPlugin:getTargetPos(trackingIndex, trackingObjectIndex,
trackingPosition,trackingOrientation)trackingObject:setPos
ition(trackingPosition, camera)
trackingObject:setOrientation(trackingOrientation, camera)

if not trackingObject:getVisible() then
trackingObject:setVisible(true) end

else

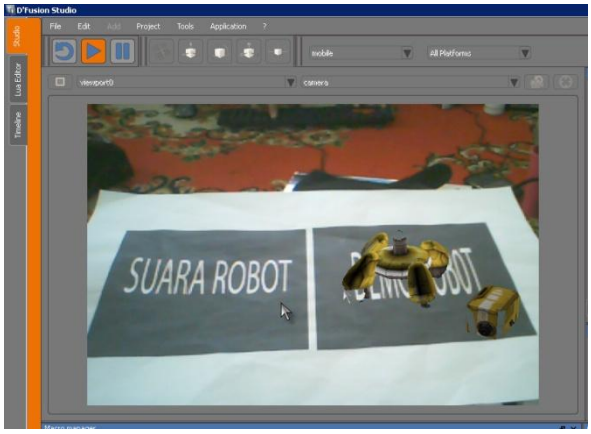
if trackingObject:getVisible() then
trackingObject:setVisible(false) end

end

if (gtrackingStatus2 == 1 ) then

if not trackingSuara:isPlaying() then trackingSuara:play()
end
else
if trackingSuara:isPlaying() then trackingSuara:stop() end
end
```

Fungsi di atas menunjukan bahwa jika status *tarcking* ada (`gtrackingStatus == 1`) maka set posisi objek 3D dimana objek marker dicari pada file tracker.xml yang digunakan (`trackingIndex = -1`) telah didapatkan,kemudian set posisi *relative*objek 3D tersebut untuk indeks marker 0 (`trackingObjectIndex`)melalui pandangan kamera yang digunakan (`trackingPosition, camera`) serta set orientasi objek tersebut pada posisi marker (`trackingOrientation, camera`). Jika objek 3D dalam keadaan tidak terlihat (`if not trackingObject:getVisible()`) maka set objek 3D menjadi ada (`trackingObject:setVisible(true)`) , jika tidak camera tidak dalam proses men-*tracking*, maka set objek 3D menjadi tidak terlihat (`trackingObject:setVisible(false)`).Untuk pernyataan kedua berisikan perintah bahwa jika status marker ada maka set suara jika suara sedang tidak aktif (`trackingSuara:isPlaying()`) menjadi aktif (`trackingSuara:play()`) jika tidak terjadi tracking maka diset kembali menjadi tidak aktif (`trackingSuara:stop()` ).




Gambar 3.21 Gambar Objek 3D dan suarapada status *tracking*

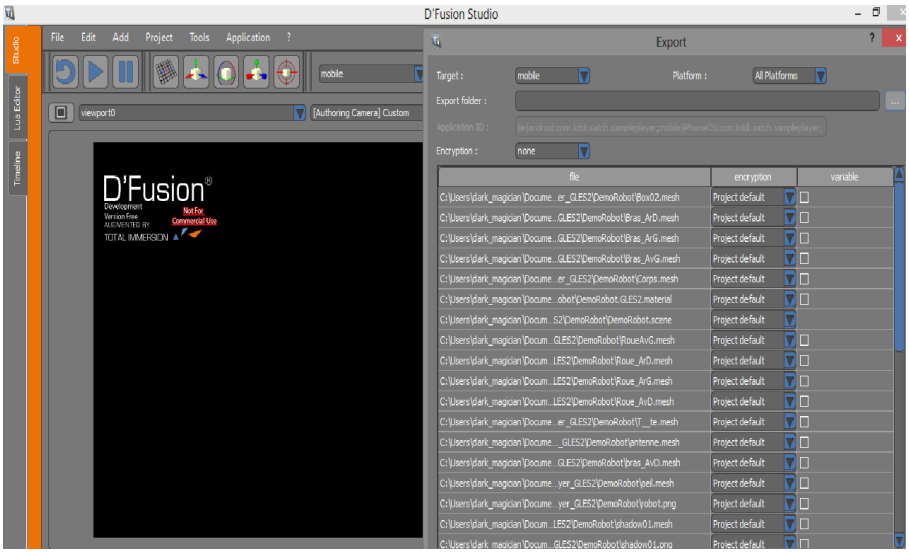
### 2.14.12 Flatform Mobile

*Flatform mobile*disini adalah merupakan penerapan teknologi *Augmented Reality* yang telah dirancang oleh *D'fusion Studio*agar dapat berjalan pada *flatfrom mobile.* Adapun material yang dibutuhkannya :

1. Scenario utama yang telah dieksport D'fusion Studio
2. D'fusion mobile SDK For Android 3.25

#### 2.14.12.1 Eksport Scenario

Setelah *scenario* telah selesai dirancang, maka agar dapat diaplikasikan dalam *flatform mobile* scenario tersebut harus di-*eksport* terlebih dahulu. D'fusion Studio memiliki fungsionalitas untuk mengekspor hasil *scenario* dengan beberapa *flatform* yang dapat digunakan yaitu untuk mobile dan juga desktop, pada tahapan ini pilih target yang digunakan agar dapat digunakan pada *flatform mobile*maka target yang digunakan adalah *mobile,* kemudian isikan Aplication Id dengan application Id yang sesuai dengan aplikasi yang akan dibangun oleh *tools*eclipse pada tahap untuk merancang pemanggilan fungsi *scenario* yang telah diekspor.




Gambar 2.22 Gambar *eksport* skenario

### 2.14.12.2 PerancanganMobile

Pada perancangan *mobile,tools*yang digunakan adalah menggunakan *eclipse.* Pada tahapan ini *scenario* yang telah diekspor akan disimpan pada sebuah direktori ‘asset’ dimana asset tersebut nantinya akan diakses langsung agar *scenario*agar dapat berjalan. Agar setiap fungsi-fungsi yang terdapat pada *scenario* dapat berjalan maka dibutukanlah SDK for Androi 3.25 yang berisikan *library*lib*Ti*Android*AR2*.so dimana seluruh fungsi tersebut ada didalamnya, didalam *eclipse*sendiri memiliki fasilitas fungsi *build patch*agar lib*Ti*Android*AR2.so*dapat digunakan. Terdapat beberapa fungsi-fungsi yang digunakan pada *class*utama untukmemanggilan *scenario* dimana fungsi-fungsi tersebut seperti :

1. Fungsi *onCreate()*

   Fungsi onCreate() digunakan untuk memanggil class baru sifatnya *privat* menjadi *public,* seperti *class*tiComponent dan FrameLayout.

2. Fungsi onStart()

   Fungsi onStart() digunakan untuk inisialisasi awal pada *class* tersebut yang fungsionalitasnya adalah untuk memanggil *scenario* utama yang berada pada *directory* ‘assets’ agar dapat berjalan.

3. Fungsi onResume()

Fungsi onResume() digunakan untuk me-*riload*ulang skenario.

4. Fungsi onPause()

   Fungsi onPause() digunakan untuk menghentikan *scenario* secara sesaat.

5. Fungsi onStop()

   Fungsi yang digunakan untuk menutup aktifitas *scenario*.

6. Fungsi onDestroy()

   Fungsi onDestroy() digunakan untuk menghentikan secara utuh semua fungsi yang ada.

Secara keseluruhan *library* yang menjalankan semua fungsi-fungsi yang ada baik *tracking*gambar sampai pencocokan pola kemudian *view* objek 3D dan mengeluarkan suara.




**Gambar 2.23 Gambar Aplikasi *mobile* AR**