BAB III
TEORI PENUNJANG

2.1 Laboratorium Hardware UNIKOM

Laboratorium *hardware* UNIKOM (Universitas Komputer Indonesia) pada awal mulanya merupakan sebuah laboratorium yang didirikan pada bulan oktober 2008 yang bertujuan untuk memfasilitasi lulusan-lulusan Universitas Komputer Indonesia khususnya fakultas teknik dibidang perangkat keras komputer.

Namun seiring berjalannya waktu dengan kegigihan dari para dosen dan antusiasme dari mahasiswa untuk mengikuti mata kuliah ini, maka UNIKOM pun mengeluarkan surat keputusan berupa menjadikan mata kuliah hardware komputer sebagai mata kuliah wajib di UNIKOM yang artinya tidak hanya fakultas teknik yang mendapat mata kuliah ini melainkan fakultas non-teknik pun harus mengambil mata kuliah ini. Tentunya pertimbangan tersebut diambil berdasarkan nama dari universitas sendiri yang berbasis komputer maka unikom mewajibkan seluruh mahasiswa/inya untuk dapat memahami komputer baik secara software maupun *hardware.* [12]

2.2 Computer Hardware

Computer Hardware (Perangkat Keras Komputer) adalah beberapa komponen fisik komputer yang bersifat dapat dilihat, diraba dan berbentuk nyata yang membentuk suatu kesatuan sistem PC (*Personal Computer*). Fungsi dari perangkat-perangkat ini sangat beragam, diantaranya sebagai alat *input*, alat *output,* perangkat pemroses data, dan juga sebagai alat tambahahan dengan beragam fungsi tambahan yang tidak selalu ada di semua sistem PC.

Biasanya perangkat-perangkat ini dirakit, sebagian besar di masukkan ke dalam casing komputer dan sebagian lain berada di luar casing. Berikut merupakan gambar macam-macam perangkat keras komputer dapat dilihat pada Gambar 2.1: [10]



Gambar 2.1 Macam-Macam Perangkat Keras Komputer

2.3 Augmented Reality

Augmented Reality (AR) adalah sebuah istilah untuk lingkungan yang menggabungkan dunia nyata dan dunia virtual yang dibuat oleh komputer sehingga batas antara keduanya menjadi sangat tipis. Menurut Ronald Azuma pada tahun 1997 mendenisikan *Augmented Reality* sebagai sistem yang memiliki karakteristik sebagai berikut:

1. Menggabungkan lingkungan nyata dan virtual
2. Berjalan secara interaktif dalam waktu nyata
3. Integrasi dalam tiga dimensi (3D)

Secara sederhana AR dapat didenisikan sebagai lingkungan nyata yang ditambahkan objek virtual. Penggabungan objek nyata dan virtual dimungkinkan dengan teknologi *display* yang sesuai, interaktivitas dimungkinkan melalui perangkat-perangkat input tertentu.

AR merupakan variasi dari *Virtual Environments* (VE), atau yang lebih dikenal dengan istilah *Virtual Reality* (VR). Teknologi VE membuat pengguna tergabung dalam sebuah lingkungan virtual secara keseluruhan. Ketika tergabung dalam lingkungan tersebut, pengguna tidak dapat melihat lingkungan nyata di sekitarnya. Sebaliknya, AR memungkinkan pengguna untuk melihat lingkungan nyata, dengan objek virtual yang ditambahkan atau tergabung dengan lingkungan

nyata. Tidak seperti VR yang sepenuhnya menggantikan lingkungan nyata, AR sekedar menambahkan atau melengkapi lingkungan nyata. [1]

Tujuan utama dari AR adalah untuk menciptakan lingkungan baru dengan menggabungkan interaktivitas lingkungan nyata dan virtual sehingga pengguna merasa bahwa lingkungan yang diciptakan adalah nyata. Dengan kata lain, pengguna merasa tidak ada perbedaan yang dirasakan antara AR dengan apa yang mereka lihat/rasakan di lingkungan nyata. Dengan bantuan teknologi AR (seperti visi komputasi dan pengenalan objek) lingkungan nyata disekitar kita akan dapat berinteraksi dalam bentuk digital (*virtual*). Informasi tentang objek dan lingkungan disekitar kita dapat ditambahkan kedalam sistem AR yang kemudian informasi tersebut ditampilkan diatas layer dunia nyata secara realtime seolah-olah informasi tersebut adalah nyata. Informasi yang ditampilkan oleh objek virtual membantu pengguna melaksanakan kegiatan-kegiatan dalam dunia nyata. AR banyak digunakan dalam bidang-bidang seperti kesehatan, militer, industri manufaktur dan juga telah diaplikasikan dalam perangkat-perangkat yang digunakan orang banyak, seperti pada telepon genggam. Berikut merupakan contoh dari penerapan teknologi *Augmented Reality* sederhana dapat dilihat pada Gambar 2.2. [1]



Gambar 2.2 Contoh Augmented Reality Sederhana

Augmented Reality dapat di klasifikasikan menjadi dua berdasarkan ada tidaknya penggunaan *marker* yaitu:

1. *MarkerAugmented Reality*

   Sebuah metode yang memanfaatkan marker yang biasanya berupa ilustrasi hitam dan putih berbentuk persegi atau lainnya dengan batas hitam tebal dan latar belakang putih. Melalui posisi yang dihadapkan pada sebuah kamera komputer atau *smartphone,* maka komputer atau *smartphone* tersebut akan melakukan proses menciptakan dunia virtual 2D atau 3D. Marker Based Tracking ini sudah lama dikembangkan sejak 1980 dan pada awal 1990 mulai dikembangkan untuk pengguna *Augmented Reality.*

2. *Markerless Augmented Reality*

   Salah satu metode Augmented Reality yang saat ini sedang berkembang adalah metode *Markerless Augmented Reality.* Dengan menggunakan metode ini maka pengguna tidak perlu menggunakan sebuah marker untuk menampilkan objek 3D atau yang lainnya. Sekalipun dengan markerless namun aplikasi tetap berjalan dengan melakukan pemindaian terhadap objek, namun ruang lingkup yang dipindai lebih luas disbanding dengan *Marker Based Tracking*.

   Saat ini Augmented Reality dikembangkan oleh perusahaan terbesar di dunia yaitu Total Imersion, yang telah membuat berbagai macam teknik Markerless Tracking sebagai teknologi andalan perusahaan, seperti *Image Target, Face Tracking, 3D Object Tracking, dan Motion Tracking.*

   a. *Image Target*

      Image Target adalah sebuah metode dimana marker yang dibaca oleh sistem AR adalah sebuah Gambar, baik Gambar hitam putih atau Gambar berwarna.

   b. *Face Tracking*

      Dengan menggunakan algoritma yang telah dikembangkan, komputer dapat mengenali wajah manusia secara umum dengan cara mengenali semua komponen wajah manusia kemudian akan mengabaikan objek-

objek lain di sekitarnya seperti pohon, rumah, dan benda-benda lainnya.

c. *3D Object Tracking*

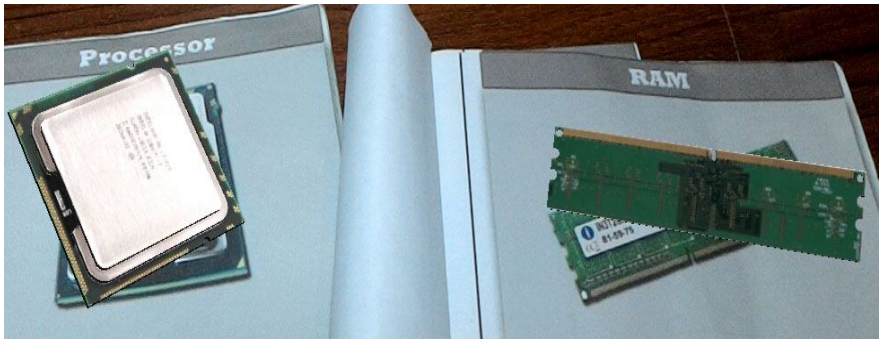
Berbeda dengan *Face tracking* yang hanya mengenali wajah manusia secara umum, teknik *3D Object Tracking* dapat mengenali semua bentuk benda yang ada disekitar, seperti mobil, meja, televisi, dan lain sebagainya.

d. *Motion Tracking*

Teknik ini merupakan teknik yang membuat komputer dapat menangkap gerakan. *Motion Tracking* telah mulai digunakan secara ekstensif untuk memproduksi film-film yang mencoba mensimulasikan gerakan. Contohnya pada film Avatar, dimana James Cameron menggunakan teknik ini untuk membuat film tersebut dangn menggunakannya secara *real-time.* [3]

2.3.1 Multi Marker

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



Gambar 2.3 Contoh Multi Marker

2.4 Aplikasi

Aplikasi adalah suatu penerapan dan pengimplementasian data dan permasalahan kedalam suatu media yang digunakan untuk penerapan permasalahan tersebut sehingga menjadi bentuk yang baru. Maka hanya bentuk dari tampilan data yang berubah sedangkan isi yang termuat dalam data tersebut tidak mengalami perubahan.

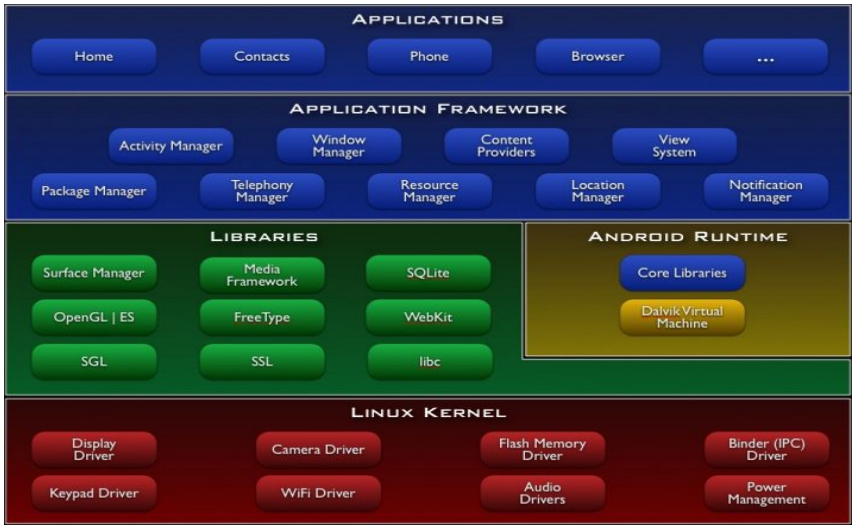
2.5 Android

Android adalah sistem operasi berbasis Linux yang dipergunakan sebagai pengelola sumber daya perangkat keras, baik untuk ponsel, *smartphone* dan juga PC tablet. Secara umum Android adalah platform yang terbuka (*Open Source*) bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai piranti bergerak.

Pada awalnya *Google Inc* membeli Android Inc sebagai pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah *Open 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, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. [3]

2.5.1 Arsitektur Android

Secara umum arsitektur android di bagi 5 layer elemen, setiap layer terdiri dari beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan pada Gambar 2.4: [3]



Gambar 2.4 Arsitektur Android

1. *ApplicationLayer*

Aplication Layer merupakan lapisan paling tampak pada pengguna ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam android runtime dengan menggunakan kelas dan service yang tersedia pada framework aplikasi.

2. *Aplication Framework Layer*

Merupakan kerangka aplikasi yang menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu, juga menyediakan abstraksi generic untuk mengakses perangkat, serta mengatur tampilan user interface dan sumber daya aplikasi.

3. *Libraries Layer*

Merupakan layer yang didalamnya terdapat fitur-fitur android, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya berjalan di atas karnel. *Layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta:

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries* untuk manajemen tampilan.
3. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
4. *Libraries* SQLite untuk mendukung *database.*
5. *Libraries* SSL dan WebKit terintegrasi dengan *web browser* dan *security*.

6. *Libraries* Live Webcore mencakup modern *web browser* dan *engine embeded web view*.
7. *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's.

4. *Android Run Time Layer*

Layer ini merupakan *layer* yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. *Dalvix Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi android. Di dalam android *run time* dibagi menjadi dua bagian, yaitu:

a. *Core Libraries*

Aplikasi android dibangun dalam bahasa java, sementara *Dalvik* sebagai virtual mesinnya bukan *virtual machine* java, sehingga diperlukan sebuah libraries yang berfungsi untuk menterjemahkan bahasa java/C yang ditangani oleh *Core Libraries*.

b. *Dolvik Virtual Machine*

Virtual mesin berbasis register yang di optimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat linux karnel untuk melakukan threading dan manajemen tingkat rendah.

5. *Linux Kernel Layer*

Merupakan *layer* dimana inti dari sistem operasi android itu berada. *Layer* ini berisi *file-file system* yang mengatur sistem *processing, memory, resource, drivers,* dan sistem-sistem operasi android lainnya. [3]

2.6 Tools Pendukung Pembuatan Aplikasi Android

Adapun tools pendukung dalam pembangunan aplikasi *Augmented Reality Computer Hardware* yaitu:

2.6.1 Java Development Kit (JDK)

JDK adalah *software development* kit merupakan seperangkat kode, *library*, form dan paket yang berguna untuk membuat aplikasi atau applet java. Di

dalam JDK terdapat Java Runtime Environment (JRE) yang akan memproses byte code yang dihasilkan dari interpenter java. Tugas dari interpenter java ini adalah untuk mengcompile source.java menjadi source.class yang selanjutnya source.class inilah yang akan dijalankan oleh JRE.

2.6.2 Software Development Kit (SDK)

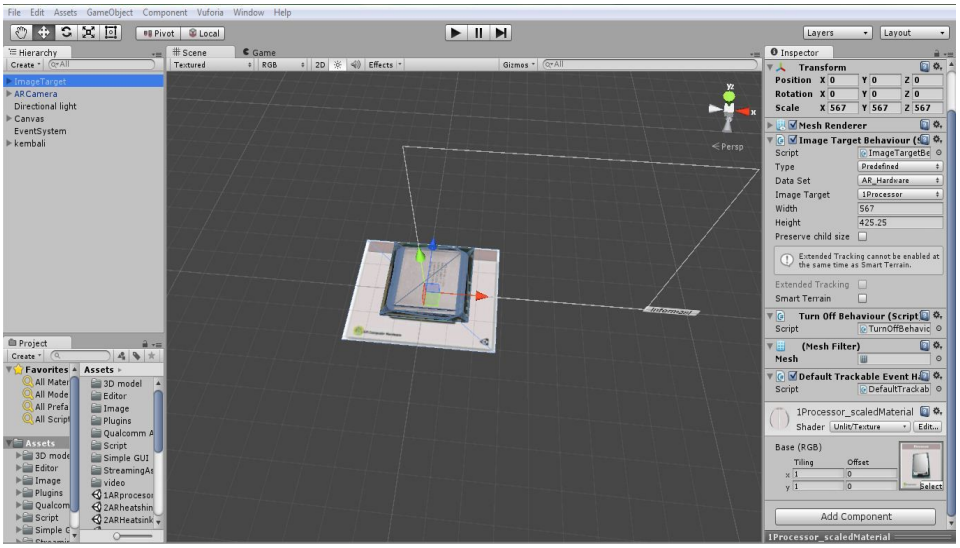
SDK adalah suatu kumpulan dari library dan toolkit seperti emulator untuk mengembangkan atau menciptakan aplikasi untuk suatu perangkat lunak, perangkat *computer*, system operasi atau platform. Di dalam SDK terdapat tools yang dibutuhkan dalam pengembangan android, diantaranya yaitu adb shell yang merupakan singkatan dari android development bridge yang dapat menjalankan terminal android seperti terminal pada system operasi linux, dan command yang terdapat dalam adb shell sendiri seperti command linux pada umumnya.

2.6.3 Unity 3D Engine

Unity Engine merupakan suatu software *game engine* yang terus berkembang saat ini. Penggunaan *engine* versi *free* masih dibatasi dengan beberapa fitur yang dikurangi atau bonus *modul/prefab* tertentu yang ditiadakan dan hanya tersedia untuk pengguna berbayar. *Unity Engine* dapat mengolah beberapa data seperti gambar tiga dimensi, suara, tekstur, dan lain sebagainya. Keunggulan dari *unity engine* ini yaitu, dapat menangani grafik dua dimensi dan tiga dimensi. Namun *engine* ini lebih konsentrasi pada pembuatan grafik tiga dimensi. Dari beberapa *game engine* yang sama-sama menangani grafik tiga dimensi, *Unity Engine* dapat mengani lebih banyak.

Beberapa diantaranya yaitu Windows, MacOS X, iOS, PS3, wii, Xbox 360, dan android yang lebih banyak daripada *game engine* seperti *Source Engine*, *Game Marker*, *Unigine*, *id Tech 3 Engine*, *Blender Game Engine*, *NeoEngine, Quake Engine, C4 Engine* atau *game engine* lain. *Unity Engine* memiliki kerangka kerja (*framework*) lengkap utnuk pengembangan professional. Sistem inti *engine* ini menggunakan beberapa pilihan bahasa pemrograman, diantaranya C#, javascrit maupun boo.

Unity3D editor menyediakan beberapa alat untuk memepermudah pengembangan yaitu *Unity Tree* dan *Terrain Creator* atau mempermudah pembuatan *vegetasi* dan *terrain* serta *MonoDevelop* untuk proses pemrograman. Tampilan dari software Unity3D *engine* dapat dilihat pada Gambar 2.5. [5]



Gambar 2.5 Tampilan Unity3d

2.6.4 Vuforia Augmented Reality

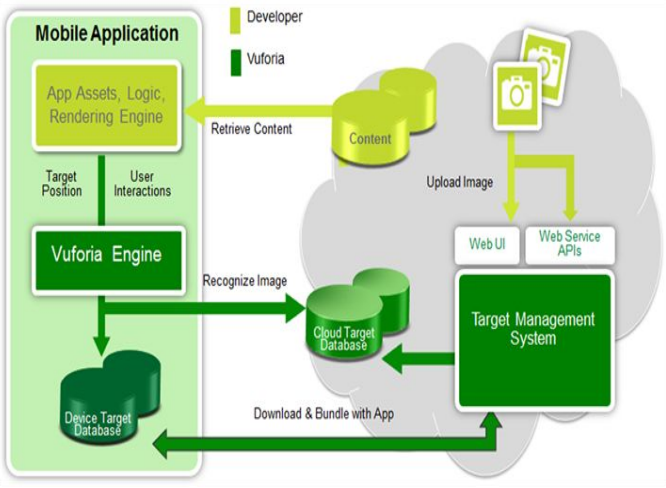
Vuforia merupakan *librarry* untuk *Augmented Reality* yang dikembangkan oleh Qualcomm, yang menggunakan sumber konsisten mengenai *computer vision* yang fokus pada *image recognition*. Vuforia mempunyai banyak fitur-fitur dan kemampuan, yang dapat membantu pengembangan untuk mewujudkan pemikiran mereka tanpa adanya batas secara teknikal.

Dengan *support* untuk sistem operas iOS, android, dan Unity3D, *platform* Vuforia mendukung para pengembang untuk membuat aplikasi yang dapat digunakan di hampir seluruh jenis *smartphone* dan *tablet*.

Dalam pengembangannya juga pengguna diberikan kebebasan untuk mendesain dan membuat aplikasi yang mempunyai kemampuan antara lain:

1. Teknologi *computer vision* tingkat tinggi yang mengijinkan *developer* untuk membuat efek khusus pada *mobile device*.
2. Terus-menerus mengenali *multiple image*.

3. *Tracking* dan detection tingkat lanjut.
4. Solusi pengaturan *database* gambar yang fleksibel. [4]



Gambar 2.6 Struktur Vuforia

Target pada Vuforia merupakan obyek pada dunia nyata yang dapat dideteksi oleh kamera, untuk menampilkan obyek virtual. Beberapa jenis target pada Vuforia adalah:

1. *Image Targets*, contoh: foto, papan permainan, halaman majalah, sampul buku, kemasan produk, poster, kartu ucapan.
2. *Frame markers*, tipe *frame* gambar 2D dengan patteren khusus yang dapat digunakan sebagai permainan.
3. *Multi-target*, contohnya kemasan produk atau produk yang berbentuk kotak ataupun persegi. Jenis ini dapat menampilkan gambar sederhana *Augmented Reality* 3D.
4. *Virtual buttons,* yang dapat membuat tombol sebagai daerah kotak sebagai sasaran Gambar. [4]

2.6.4.1 Arsitektur Vuforia

Vuforia SDK memerlukan beberapa komponen penting agar dapat bekerja dengan baik. Komponen-komponen tersebut antara lain:

a. Kamera

Kamera dibutuhkan untuk memastikan bahwa setiap *frame* ditangkap dan diteruskan secara efisian ke *tracker*. Para developer hanya tinggal memeberi tahu kamera kapan mereka mulai menangkap dan berhenti.

b. Image Converter

Mengkonfersi format kamera (misalnya YUV12 kedalam format yang dapat dideteksi oleh OpenGL (misalnya RGB565) untuk *tracking* (misalnya *luminance*).

c. Tracker

Mengandung algoritma *computer vision*untuk melaukan pelacakan objek nyata yang diambil dari kamera. Objek tersebut dievaluasi dan hasilnya akan disimpan dalam *statae object* yang akan digunakan oleh video *background renderer* dan dapat dikases dari *application code*.

d. Video Background Render

Me-*render* gambar dari kamera yang tersimpan di dalam *state object*. Performa dari *video background render* sangat bergantung pada *device* yang digunakan.

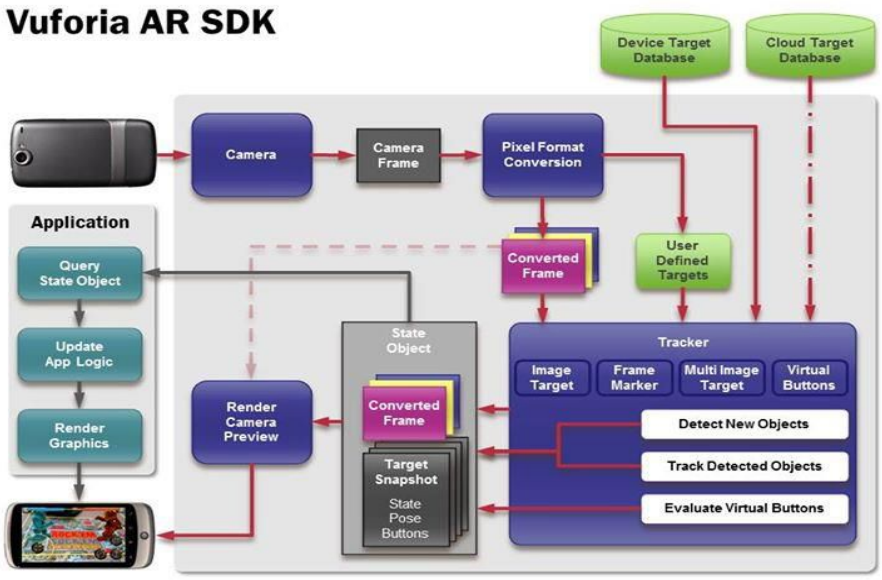
e. Application Code

Melibatkan analisasi dari semua komponen diatas. Selama objek yang dikehendaki diubah prosesnya, maka *application code* harus diubah berdasarkan lokasi objek virtual.

f. Target Resource

Dibuat menggunakan online *Target Management System*. Assets di unduh berisi sebuah konfigurasi xml config.xml yang memungkinkan developer untuk mengkonfigurasi beberapa fitur dalam trackable dan binary file yang berisi database trackable.

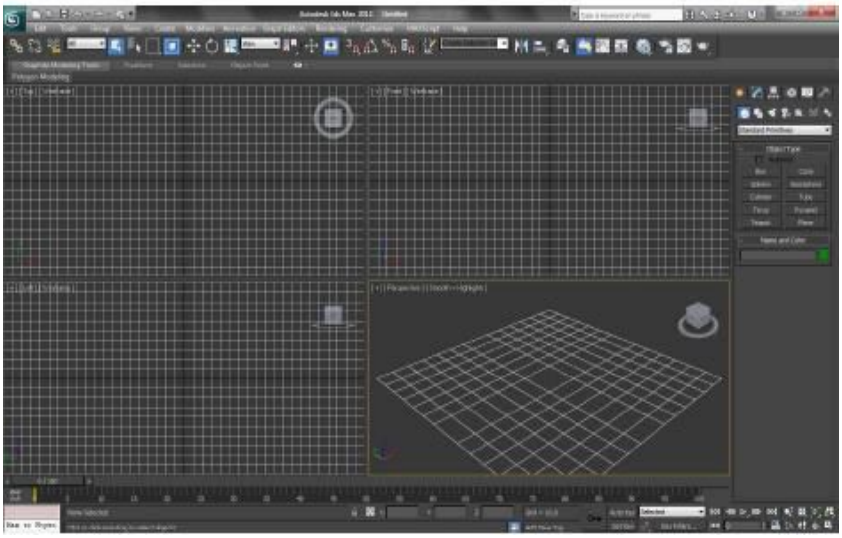
Berikut adalah gambaran dari diagaram alir data Vuforia, dapat dilihat pada Gambar 2.7. [4]



Gambar 2.7 Diagram Alir Data Vuforia

2.6.5 3DS Max

Autodesk 3DS Max adalah software visualisasi (modeling dan animasi) tiga dimensi yang popular dan serbaguna. Hasil yang dibuat di 3D Studio Max sering digunakan di televisi, media cetak, games, web dan lain-lain. Berikut merupakan tampilan dari software 3DS Max dapat dilihat pada Gambar 2.8.



Gambar 2.8 Tampilan 3DS Max

2.7 Metode FAST Corner Detection

FAST (*Feature Form Accelerated Segment Test*) *corner detection* adalah suatu algoritma yang dikembangkan oleh Edward Rosten, Reid Porter, and Tom Drummond. FAST *corner detection* ini dibuat dengan tujuan mempercepat waktu komputasi secara *real-time* dengan konsekuensi menurunkan tingkat akurasi pendeteksian sudut. [6]

FAST *corner detecteion* bekerja pada suatu citra sebagai berikut:

1. Tentukan sebuah titik *p* pada citra dengan posisi awal (x_p,y_p).
2. Tentukan keempat titik. Titik pertama (*n=1*) *terletak pada koordinat* (-3) titik kedua(*n*=2) terletak pada koordinat +3, titik ketiga terletak pada koordinat (*n*=3) terletak pada koordinat (-3), titik keempat (*n*=4) terletak pada koordinat (- 3).
3. Bandingkan intensitas titik pusat *p* dengan keempat titik disekitar. Jika terdapat paling sedikit 3 titik yang memenuhi syarat berikut, maka titik pusat *p*.

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \le I_{p\to x} \quad \text{(brighter)} \end{cases}$$

Keterangan:

Sp→x = Keputusan titik p sebagai sudut, nilai 1 menunjukan bahwa titik merupakan suatu sudut, dan nilai 0 menunjukan bahwa titik bukanlah sebuah sudut.

Ip = Nilai intensitas titik *p*

t = Batas ambang nilai intensitas yang ditoleransi

4. Ulangi proses sampai seluruh titik pada citra sudah dibandingkan intensitasnya.

2.8 Pemrograman Berorientasi Objek (Object Oriented)

Metode berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metode berorientasikan objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengolahan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas, analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek. [8]

Berikut beberapa contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek: [9]

1. Bahasa pemrograman Smalltalk
2. Bahasa pemrograman Eiffel
3. Bahasa pemrograman C++
4. Bahasa pemrograman (web) PHP
5. Bahasa pemrograman Java

2.8.1 Unifield Modelling Language (UML)

Unified Modeling Language (UML) adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. [9]

2.8.2 Sejarah UML

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya. [9]

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal

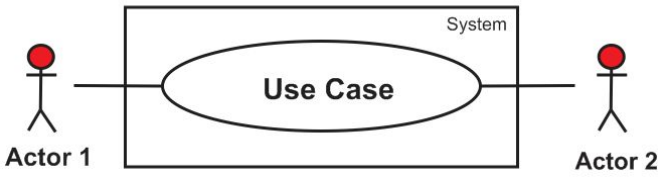
1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS. Secara aktual, penggunaan bahasa pemrograman berorientasi objek pada saat itu masih terbatas, namun telah banyak menarik perhatian di saat itu. Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek. Metode yang pertama diperkenalkan oleh Sally Shaler dan Stephen Mellor (Shaler-Mellor, 1988) dan Peter Coad dan Edward Yourdon (Coad-Yourdon, 1991), diikuti oleh Grady Booch (Booch, 1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani (Rumbaugh-Blaha-Premerlani-Eddy-Lorensen, 1991), dan masih banyak lagi. [9]

Buku terkenal yang juga berkembang selanjutnya adalah karangan Ivar Jacobson (Jacobson, 1992) yang menerangkan perbedaan pendekatan yang fokus pada *use case* dan proses pengembangan. Sekitar lima tahun kemudian muncul buku yang membahas mengenai metodologi berorientasi objek yang diikuti dengan buku-buku yang lainnya. Didalamnya juga membahas mengenai konsep, definisi, notasi, terminologi, dan proses mengenai metodologi berorientasi objek. [9]

Karena banyaknya metodologi-metodologi yang berkembang pesat itu, maka muncul ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep *Object Modeling Technique* (OMT) dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators* (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainya dimana James R. Rumbaigh, Grandy Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language.* [9]

2.8.3 Use Case Diagram

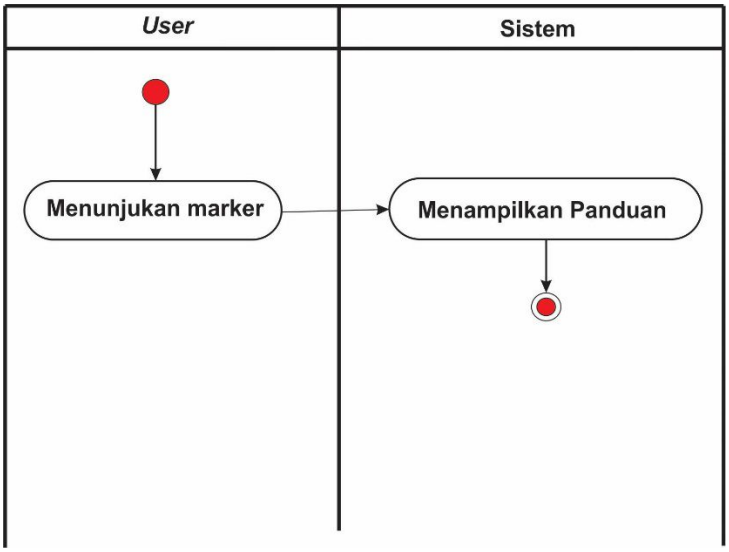
Use Case Diagram merupakan diagram yang dirancang untuk menunjukan fungsionalitas suatu sistem yang berinteraksi dengan dunia luar. *Use Case Diagram* juga digunakan untuk menjelaskan manfaat suatu sistem jika dilihat dari sudut pandang orang yang berada di luar sistem. Berikut merupakan contoh gambar *Use Case* diagram dapat dilihat pada Gambar 2.9. [9]



Gambar 2.9 Contoh Use Case Diagram

2.8.4 Activity Diagram

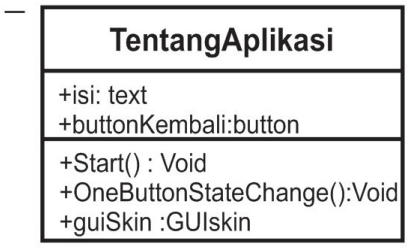
Diagram yang menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat digunakan untuk aktivitas lainnya seperti *Use Case*. Berikut merupakan contoh activity diagram dapat dilihat pada Gambar 2.10. [9]



Gambar 2.10 Contoh Gambar Activity Diagram

2.8.5 Class Diagram

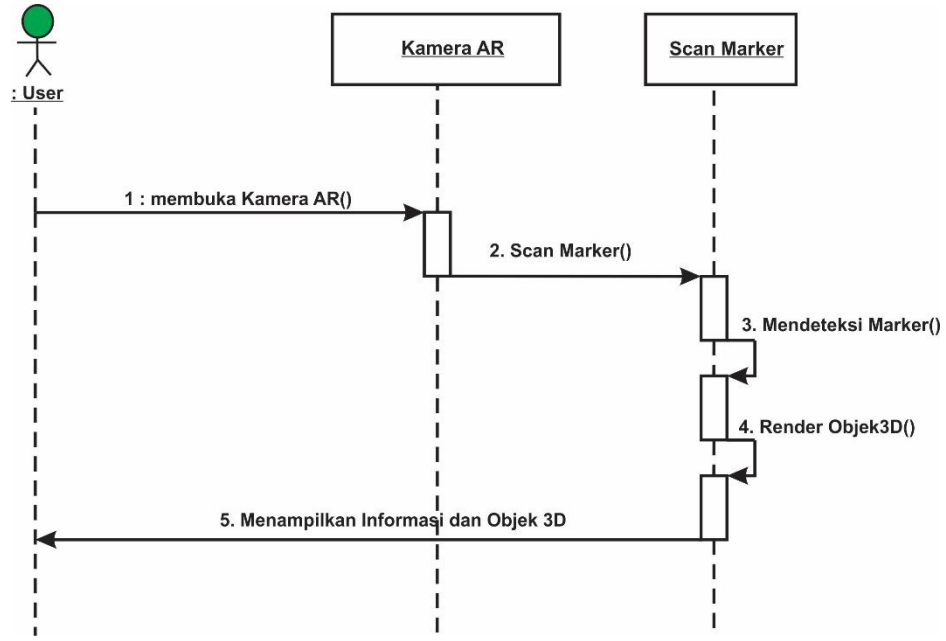
Diagram yang menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem. *Class Diagram* menunjukan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain suatu sistem. Berikut merupakan contoh class diagram dapat dilihat pada Gambar 2.11. [9]



Gambar 2.11 Contoh Class Diagram

2.8.6 Squence Diagram

Menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan *use case*. *Sequence diagram*, memperlihatkan tahap demi tahap yang seharusnya terjadi untuk menghasilkan sesuatu dalam *use case*. Berikut merupakan contoh gambar squence diagram dapat dilihat pada Gambar 2.12. [9]



Gambar 2.12 Contoh Squence Diagram

2.9 Alpha Testing

Didalam pengembangan software, pengujian pada umumnya sangat diperlukan sebelum perangkat lunak tersebut dipublikasikan. Pengujian yang di kenal dengan *alpha* testing sering dilakukan di bawah suatu debugger atau dengan *hardware-assisted* yang *debugging* untuk menangkap bugs dengan cepat. Teknik ini juga dikenal sebagai *white-box testing*.

Selanjutnya diserahkan staff pengujian untuk pemeriksaan tambahan di dalam lingkungan yang serupa. Teknik ini dikenal juga dengan *black-box testing* dan sering di sebut langkah lanjutan dari *white-box testing/alpha testing*.

Pengujian alpha merupakan pengujian fungsional yang digunakan untuk menguji sistem yang baru. Pengujian alpha berfokus pada persyaratan fungsional perangkat lunak.

2.10 Beta Testing

Pengujian *beta* merupakan pengujian langsung kepada pengguna untuk mencoba aplikasi yang baru. Pengujian *beta* diadakan dilingkungan *live* (sebenarnya), dalam pengujian *beta end user* mencatat kemudian menyampaikan pada pihak devloper.

Pengujian dilakukan pada satu atau lebih pelanggan oleh pemakai akhir perangkat lunak dalam lingkungan yg sebenarnya, pengembang biasanya tidak ada pada pengujian ini. Pelanggan merekan semua masalah (*real* atau *imajiner*) yg ditemui selama pengujian dan melaporkan pada pengembang pada interval waktu tertentu. Pengujian bertujuan untuk meningkatkan jumlah para pemakai di masa yang akan datang.

2.10.1 Pengujian Skala Likert

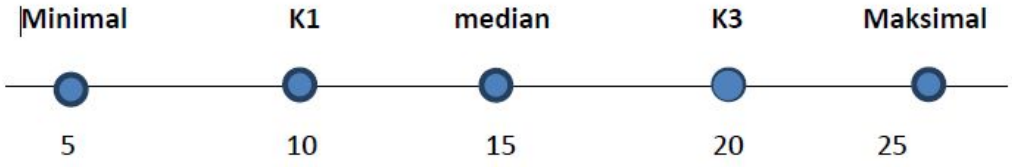
Skala *likert* adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. *Rensis likert* telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala *likert*. Skala *likert* ini merupakan skala yang dapat

dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena.

Dalam skala *likert* terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif dan sikap negatif yang berfungsi untuk mengukur sikap negative objek. Contoh skor pernyataan positif dimulai dari 1 untuk tidak setuju (STS), 2 untuk tidak setuju (TS), 3 untuk raguragu (R), 4 untuk setuju (S), dan 5 untuk sangat setuju (SS). Skor pernyataan negative dimulai dari 1 untuk sangat setuju (SS), 2 untuk setuju (S), 3 untuk raguragu (R), 4 untuk tidak setuju (TS), dan 5 untuk sangat tidak setuju (STS). Beberapa menghilangkan option "Ragu-ragu" dalam instrument untuk memudahkan dalam melihat angket yang responden isikan. Skala *likert* digunakan untuk mengukur kesetujuan dan ketidak setujuan seseorang terhadap sesuatu rencana program, pelaksanaan program ataupun tingkat keberhasilan suatu program.

Berikut adalah tahapan untuk mengetahui bagainana sikap tiap responden:

1. Menentukan skor maksimal, yaitu skor jawaban terbesar di kali banyak pertanyaan. 5 x 5 = 25
2. Menentukan skor minimal, yaitu skor jawaban terkecil dikali banyak pertanyaan. 1 x 5 = 5
3. Menentukan nilai median, yaitu hasil penjumlahan skor maksimal dengan skor minimal dibagi dua. (25+5) : 2 = 15
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan skor minimal dengan median dibagi dua. (5+15) : 2 = 10
5. Menentukan nilai kuartil 3, yaitu hasil penjumlahan skor maksimal dengan median dibagi dua. (25+15) : 2 = 20
6. Buatlah skala yang menggambarkan skor minimal, nilai kuartil 1, median, kuartil 3 dan skor maksimal.



Gambar 2.13 Contoh Skala Likert