BAB 2
LANDASAN TEORI

2.1 Permainan *(Game)*

Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, play dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan.

Game bertujuan untuk menghibur, *game* banyak disukai oleh anak – anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut untuk menyelesaikannya dengan cepat dan tepat. G*ame* juga bisa merugikan karena apabila sudah kecanduan *game* akan lupa waktu dan akan mengganggu kegiatan atau aktivitas yang sedang dilakukan [4].

Berdasarkan jenisnya *game* dibagi menjadi beberapa kategori diantaranya berdasarkan *platfrom* atau alat yang digunakan [4] :

1. Arcade *game*s, yaitu video *game* dengan mesin yang memang khusus di desain untuk jenis video *game*s tertentu dan memiliki fitur seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil dan lain-lain.
2. PC *Game*s, yaitu video *game* yang dimainkan menggunakan Personal Computers.
3. *Console game*s, yaitu video *game*s yang dimainkan menggunakan *Console* tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.
4. *Handheld game*s, yaitu yang dimainkan di *Console* khusus video *game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.
5. *Mobile game*s, yaitu yang dapat dimainkan atau khusus untuk *Mobile phone* atau PDA.

Berdasarkan *genre* permainanya[4] :

1. *Shooting*

   Shooting merupakan salah satu *genre* dalam *game* biasanya mensimulasikan tembak-tembakan , atau hajar-hajaran bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya, *video game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari *game* jenis ini adalah tembak.

2. *Fighting*

   Fighting (pertarungan) jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan *skill*karakter yang di mainkan, pengenalan karakter dan timing sangatlah penting, dan berbeda seperti *game* Aksi pada umumnya yang umumnya hanya melawan *Artificial Intellegence* atau melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya.

3. Petualangan

   Refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. *Video Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.termasuk didalamnya petualangan dengan teks atau sistem tunjuk dan klik.

4. Simulasi, Konstruksi dan manajemen

   Video Game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium *cloning*. *Video Game* jenis ini membuat pemain harus berpikir untuk

mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas.

5. *Role Playing*

   Video game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter dan lain-lain.

6. Strategi

   Kebalikan dari *video game* jenis action yang berjalan cepat dan perlu reflex secepat kilat, *video game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. *Video game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusal pelatihan tempur, tergantung dari tema ceritanya. Pemain *game* strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan *game* action. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumberdaya (uang, besi, kayu,minyak,dll), hingga ke pembelian dan pengupgrade-an pasukan atau teknologi.

7. Olahraga

   Biasanya *game genre* ini permainannya diusahakan serealistik mungkin di buat se mirip mungkin dengan aslinya walau kadang ada yang menambah unsur fiksi dalam *game* olahraga tersebut.

8. Edukasi

   Video Game jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, baik itu untuk belajr mengenal warna, mengenal huruf dan angka, matematika,

belajar bahasa asing dan lain-lain.*Developer* yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan desain visual ataupun animasinya.

9. *Casul Game*

Casual game memiliki aturan yang sangat sederhana atau teknik bermain dan tingkat yang sangat rendah strategi. *Casual game* juga tidak memerlukan komitmen waktu jangka panjang atau keahlian khusus untuk bermain, membuat mereka mudah untuk belajar dan bermain. Ada relatif produksi rendah dan biaya distribusi bagi produsen. Tujuan dari *casual game* adalah untuk menghibur, tetapi dengan komitmen yang jauh lebih rendah dibandingkan permainan video lainnya.

2.1.1 Permainan Tradisional

Permainan tradisional dapat diartikan sebagai satu kegiatan yang menyenangkan yang dilakukan menurut tradisi, sehingga menimbulkan rasa puas pada pelakunya.

Permainan tradisional adalah suatu jenis permainan yang ada pada satu daerah tertentu yang berdasarkan kepada kultur atau budaya daerah tersebut. Permainan tradisional biasanya dimainkan oleh orang-orang pada daerahnya tertentu dengan aturan dan konsep yang tradisional pada jaman dulu. Permainan tradisional kurang begitu dikenal oleh anak-anak pada jaman sekarang kebanyakan orang tua jaman dulu yang masih sangat tahu bagaimana memainkan permainan ini khususnya mengenang masa kecil mereka bermain permainan tradisional ini [4].

2.2 Sejarah Permainan Kelereng

Di setiap daerah di Negeri ini kelereng memiliki nama yang berbeda-beda. Orang Betawi menyebut kelereng dengan nama gundu. Orang Jawa neker, di Sunda kaleci, Palembang ekar, di Banjar kleker dan masih banyak lagi.

Sejak abad ke-12 di Prancis kelereng disebut dengan bille, artinya bola kecil. Lain halnya di Belanda, para Sinyo-Sinyo itu menyebutnya dengan knikkers. di Jawa, knikkers diserap menjadi nekker. Mengingat Belanda pernah numpang hidup di Indonesia.

Tahun 1694 di Inggris ada istilah marbles untuk menyebut kelereng. Marbles sendiri digunakan untuk menyebut kelereng terbuat dari marmer yang didatangkan dari Jerman. Namun, jauh sebelumnya, anak-anak di Inggris telah akrab menyebutnya dengan bowls atau knikkers. Kelereng populer di Inggris dan negara Eropa lain sejak abad ke-16 hingga 19. Setelah itu baru menyebar ke Amerika, bahan pembuatnya adalah tanah liat dan diproduksi besar-besaran.

Jauh pada peradaban Mesir kuno, tahun 3000 SM kelereng terbuat dari batu atau tanah liat. Kelereng tertua koleksi The British Museum di London berasal dari tahun 2000-1700 SM kelereng tersebut ditemukan di Kreta pada situs Minoan of Petsofa.

Pada masa Rowami permainan Kelereng juga sudah dimainkan secara luas, bahkan menjadi salah satu bagian dari festival Saturnalia yang diadakan saat menjelang perayaaan Natal. Saat itu semua orang saling memberikan sekantung biji-bijian yang berfungsi sebagai kelereng tanda persahabatan. Salah seorang penggemar kelereng adalah Octavian kelak menjadi Kaisar Agustus. Layaknya permainan, di Romawi saat itu juga mempunyai aturan-aturan resmi. Peraturan tersebut menjadi dasar permainan sekarang.

Teknologi pembuatan kelereng kaca ditemukan pada 1864 di Jerman. Kelereng yang semula satu warna, menjadi berwarna-warni mirip permen. Teknologi ini segera menyebar ke seluruh Eropa dan Amerika, namun akibat Perang Dunia II, pengiriman mesin pembuat kelereng itu sempat terhenti dan akhirnya masing-masing negara mengembangkannya sendiri [5].

2.2.1 Peraturan Permainan Kelereng

Kelereng adu/target disusun secara lurus di atas arena Jirak atau kalang yang berbentuk oval, jumlah kelereng berdasarkan perjanjian antara pemain, karena kelereng tersebut adalah kelereng taruhan. Tujuan utama pemain yaitu membidik

kelereng adu/target yang berada didalam arena jirak/kalang dan mengeluarkannya dengan menggunakan kelereng utama/kojo, kelereng adu/target yang keluar dari arena jirak/kalang menjadi milik pemain. Pemain yang paling banyak mengeluarkan kelereng adu/target dari arena jirak/kalang maka pemain tersebut menjadi pemenang, apabila kelereng utama/kojo yang digunakan pemain berhenti didalam arena jirak/kalang maka pemain dinyatakan kalah [5].

2.3 Algoritma Greedy Best-First Search

Algoritma yang termasuk kedalam kategori informed *search* adalah *Greedy Best First Search* yang dikenal juga dengan *Greedy Search.* Secara harfiah *greedy* artinya rakus atau tamak, sifat yang berkonotasi negative sesuai dengan arti tersebut, prinsip greedy adalah mengambil keputusan yang dianggap terbaik hanya untuk saat itu saja yang diharapkan dapat memberikan solusi terbaik secara keseluruhan. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. *Greedy Best First Search* seperti halnya algoritma yang menggunakan strategi *best-first search* lainnya mempunyai sebuah fungsi yang menjadi acuan kelayakan sebuah simpul yaitu fungsi evaluasi f(n). Pada *Greedy Best First Search* fungsi evaluasi tidak bergantung pada *cost* sebenarnya, tetapi hanya tergantung pada fungsi *heuristic* h(n) itu sendiri yang mengestimasikan arah yang benar, sehingga pencarian jalur dapat berlangsung dengan sangat cepat [6].

Greedy Best First Search ini Merupakan jenis *Best First Search* yang hanya mempertimbangkan biaya perkiraan (*estimated cost*) yaitu $f(n) = h(n)$ sedangkan harga sesungguhnya tidak digunakan sehingga solusi yang dihasilkan tidak optimal. Algoritma greedy ini membentuk solusi langkah per langkah (*step by step*). Pada setiap langkah, terdapat banyak pilihan yang perlu dieksplorasi oleh karena itu pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan.

Algoritma ini memerlukan fungsi *heuristik* yang akan mengestimasi seberapa baik dibangkitkannya setiap *node,* rumus untuk mendapatkan nilai *heuristik* adalah $h(n) = abs(n.x-goal.x) + abs(n.y-goal.y)$ [9]. Fungsi *heuristik*

yang digunakan dalam algoritma ini hanya memperhitungkan biaya perkiraan(*estimated cost*) untuk mencapai tujuan (*current state),* yang dinyatakan dengan *f(n) = h(n),*

dimana :

f(n) = fungsi evaluasi

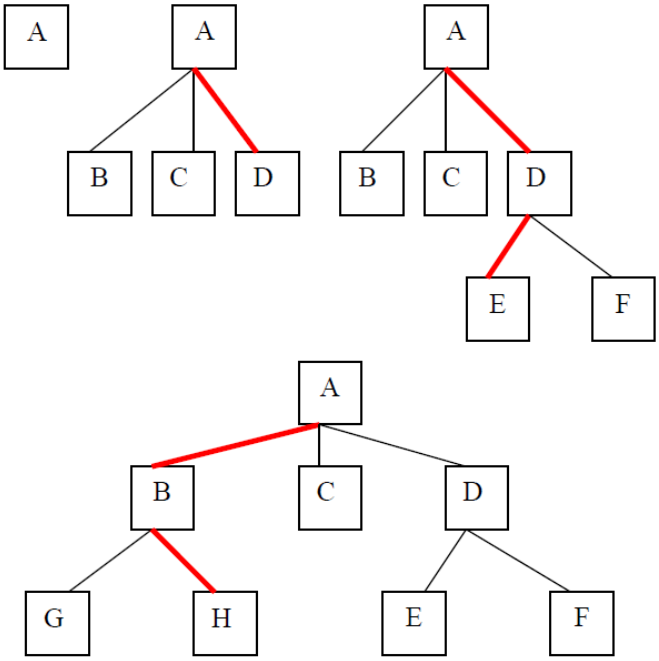
*h(*n) = fungsi *heuristik* biaya perkiraan(*estimated cost*) dari jalur terpendek dari *current state* ke *goal state.*

Algoritma *greedy best first search* ini tidak menjamin menemukan rute yang paling pendek, namun algoritma ini bekerja dengan sangat cepat menemukan tujuan (*goal*) dengan mengembangkan kemungkinan jalan yang mengarah ke tujuan. Masalah utama pada algoritma ini adalah cara kerjanya yang terus mencoba bergerak ke arah *goal*, meskipun jalan yang dilalauinya bukan jalan yang benar. Karena algoritma ini hanya memperhitungkan *estimated cost* untuk mencapai *goal state*, maka algoritma tersebut terus maju meskipun jalan yang telah dilaluinya sudah terlalu panjang [7].

Berikut beberapa istilah yang yang sering digunakan pada algoritma *greedy best first search*:

1. *Start node* adalah sebuah terminologi untuk posisi awal sebuah pencarian.
2. *Current Node* adalah simpul yang sedang dijalankan (yang sekarang). dalam algortima pencarian jalan terpendek.
3. Kandidat (*successor)* adalah simpul simpul yang berajasen dengan *current node*, dengan kata lain simpul-simpul yang akan diperiksa berikutnya.
4. Simpul (*node*) merupakan representasi dari area pencarian
5. *Open list* adalah tempat menyimpan data simpul yang mungkin diakses dari *starting node* maupun simpul yang sedang dijalankan.
6. *Closed list* adalah tempat menyimpan data simpul yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan.
7. *Goal node* yaitu simpul tujuan.
8. *Parent* adalah *current node* dari suksesor/kandidat.

Gambar dibawah ini akan mengilustrasikan langkah-langkah yang dilakukan oleh algoritma *greedy best first search*. Pertama kali, dibangkitkan dari *node* A. Kemudian semua suksesor A dibangkitkan, dan dicari harga yang minimal. Pada langkah 2, *node* D dibangkitkan, kemudian biaya akan dibandingkan dengan biaya *node* B dan C. Ternyata biaya *node* B paling kecil dibandingkan biaya *node* C, E, dan F. Sehingga B terpilih dan selanjutnya akan dibangkitkan semua suksesor B, demikian seterusnya sampai ditemukannya *node* tujuan.



Gambar 2.1 Langkah Algoritma *Greedy Best First Search*

2.4 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Pada mulanya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel.

Kemudian untuk mengembangkan Android, dibentuklah Open Hanset Alliance, konsorium dari 34 perusahaan peranti lunak dan telekomunikasi,

termasuk Google, HTC, Intel, Motorola, Qualcom, T-*Mobile* dan Nvidia. Pada saat pelirisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan standar terbuka pada perangkat seluler.

Di lain pihak, Google meliris 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 Service (GMS) dan kedua adalah yang benar – benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD) [8].

Google bersama dengan OHA merilis paket software SDK yang lengkap untuk mengembangkan aplikasi pada perangkat *Mobile*, yaitu : Sistem Operasi, Middleware dan Aplikasi utama untuk perangkat *Mobile*. Sebagai Programmer atau Developer kita bisa melakukan segalanya, mulai dari membuat aplikasi pengiriman SMS hanya dengan dua baris kode, hingga mengganti even pada Home Screen perangkat Android. Selain itu, bahkan dengan mudah kita bisa membuat dan mengkostumisasi Sistem Operasinya, atau mengganti semua aplikasi default dari google[8].

Semua aplikasi yang dibuat untuk android akan memiliki akses setara dalam mengakses seluruh kemampuan handset, tanpa membedakan apakah itu merupakan aplikasi inti atau aplikasi pihak ketiga. Dalam kata lain dengan *platform* android ini, Programer atau Developer secara penuh akan bisa mengkustomisas perangkat androidnya. Android built in pada Linux kernel (Open Linux Kernel), dengan sebuah mesin virtual yang telah didesain dan untuk mengoptimalkan penggunaan sumberdaya memori dan handware pada lingkungan perangkat *Mobile* (*Mobile Environment*). Dalvik adalah nama dari Android Virtual Mesin, yang merupakan interpreter virtual mesin yang akan mengeksekusi file dalam format Dalvik Executable (*.dex). Sebuah format yang dirancang untuk ruang penyimpanan yang efisien dan eksekusi memori yang terpetakan (memory-mappable execution) [8].

Dalvik Virtual Mesin (Dalvik VM) berbasis sergister dan dapat mengeksekusi kelasi (class) yang telah terkompilasi pada complier bahasa java, kemudian di transformasikan kedalam native format dengan menggunakan Tool “dx” yang telah di integrasi. Kita mungkin telah mengenal JavaVM (Java Virtual Mesin), yang saat ini bisa ditemukan pada setiap komputer desktop. Berbeda dengan DalvikVM, JavaVM berbasis stack (Stack-based Virtual Machine), DalvikVM memiliki keunggulan dengan menggunakan Registred Base, ini karena prosesore perangkat genggam telah dioptimalisasi untuk mengeksekusi berbasis register. Android saat ini tidak hanya berjalan pada handphone, beberapa vendor menananmkan Android pada tablet, Internet Tablet, E-Book Reader, Laptop dan gadget lainya. Adapun beberapa versi dari android yang telah diluncurkan diantaranya :

1. Android versi 1.1

   Pada 9 Maret 2009, Google meliris Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, voice search (pencarian suara), pengiriman pesan Gmail dan pemberitahuan email.

2. Android versi 1.5 (Cupcake)

   Pada pertengahan Mei 2009, Google kembali meliris telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit)* dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan meononton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke *headset* Bluetooth, naimasi layar dan *keyboard* pada layar yang dapat disesuaikan dengan system.

3. Android versi 1.6 (Donut)

   Donut (versi 1.6) diliris pada September dengan menampilkan proses pencarian yang lebih baik disbanding sebelumnya, penggunaan batrai indicator dan control applet VPN. Fitur lainya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus : kamera camcorder dan galeri yang diintegrasikan ; CDMA / EVDO, 802.1x, VPN,

Gestrures, dan *Text-to-speech* engine; kemampuan dial kontak; teknologi *text to change speech* (tidak tersedia pada semua ponsel; pengadaan resolusi VWGA.

4. Android versi 2.0/2.1 (Éclair)

   Pada 3 Desember 2009 kembali dilucurkan ponsel Android dengan versi 2.0/2.1 (Éclair), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, digital Zoom dan Bluetooth 2.1. Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan komptetisi aplikasi *Mobile* terbaik (*killer apps* - aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahap nya dipilih 50 aplikasi terbaik. Dengan semakin berkembangnya dan semakin bertambahnya jumlah handset Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka pada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah Shazam, Backgrounds dan WheaterBug. Sistem operasi Android dalam situs Internet juga dianggap penting untuk menciptakan 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 veris-veris sebelumnya antara lain dukungan Adobe Flash 1.1 , kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, integrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portable dan kemampuan auto update dalam aplikasi Android Market (Google Play).

6. Android versi 2.3 (Gingerbread)

   Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi 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 audio baru (*reverb*, *equalization*, *handphone virtualization* dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC) dan dukungan jumlah kamera yang lebih dari satu.

2.4.1 *Features*

Features yang tersedia pada *platform* Android saat ini antara lain :

a. Framework Aplikasi yang mendukung penggantian komponen dan *reusable*.
b. Mesin Virtual Dalvik berjalan diatas Linux kernel dan dioptimalkan untuk perangkat *Mobile*.
c. *Integrated browser* berdasarkan *open source engine WebKit*.
d. Grafis yang dioptimalkan dan didukung oleh library grafis 2D yang terkostumisasi, grafis 3D berdasarkan spesifikasi openGL ES 1,0 (Opsional akselarasi hardware).
e. SQLite untuk penyimpanan data.
f. Media support yang mendukung audio, video, dan gambar (MPGE, H.264, MP3, AAC, AMR, JPG, PNG, GIF) .
g. GSM Telephony (tergantung hardware).
h. Bluetooth, EDGE, 3G, dan WIFI (tergantung hardware).
i. Multi-touch : kemampuan layaknya handset modern yang dapat menggunakan dua jari atau lebih untuk berinteraksi dengan perangkat.
j. Lingkungan Development yang lengkap dan kaya termasuk perangkat emulator, tools untuk debugging, profil, dan kinerja memori, dan plugin untuk Eclips IDE.
k. Market Seperti kebanyakan Handphone yang memiliki tempat penjualan aplikasi, market pada android merupakan katalog aplikasi yang dapat di download dan di install pada Handphone melalui internet.

2.5 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:

a. Multi-*platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
b. 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.
c. 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 populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in* [8].

2.5.1 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. JFace
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 UML2 tersedia untuk membuat *Diagram* UML. Dengan menggunakan PDE setiap orang bisa membuat plug-in sesuai dengan keinginannya [8].

2.5.2 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama Simultaneous *Release*. Setiap versi peluncuran terdiri dari Eclipse *Platform* dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi eclipse yang telah diluncurkan yaitu :

1. Eclipse 3.0
2. Eclipse 3.1
3. Callisto
4. Europa
5. Ganymede
6. Galileo
7. Helios

2.6 Plugin *Android Development Tools* (*ADT*)

Android Development Tools (ADT) adalah plugin untuk *Eclipse Intergrated Development Environment* (IDE) yang dirancang untuk memberikan lingkungan yang terpadu di mana untuk membangun aplikasi Android. ADT memperluas kemampuan Eclipse untuk membiarkan para developer lebih cepat dalam membuat proyek baru Android, membuat aplikasi UI, menambahkan komponen berdasarkan Android Framework API, debug aplikasi dalam pengunaan Android SDK, dan membuat file APK untuk mendistribusikan aplikasi.

Mengembangkan aplikasi di Eclipse dengan ADT sangat dianjurkan dan merupakan cara tercepat untuk memulai membuat aplikasi android, karena banyak kemudahan-kemudahan sebagai tools yang terintegrasi seperti, custom XML editor, dan debug panel ouput. Selain itu ADT memberikan dorongan luar biasa dalam mengembangkan aplikasi Android [8].

2.7 Android *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 [8].

2.8 *Java Development Kit* (*JDK*)

JDK (Java Development Kit) adalah sebuah perangkat peralatan yang digunakan untuk membangun perangkat lunak dengan menggunakan bahasa pemrograman Java. JDK berjalan diatas sebuah virtual machine yang dinamakan JVM (Java Virtual Machine). Dokumentasi JDK berisi spesifikasi API, deskripsi fitur, panduan pengembang, referensi halaman untuk perkakas JDK dan utilitas, demo, dan link ke informasi terkait. Dokumentasi ini juga tersedia dalam bundel yand dapat diunduh dan diinstal pada komputer. Perbedaan JDK dengan SDK (Software Development Kit) yaitu JDK adalah sebuah SDK tetapi sebuah SDK tidak harus menjadi sebuah JDK [8].

2.9 *Object Oriented Programing* (OOP)

Object Oriented Programing atau dikenal dengan pemograman berorientasi *object* merupakan paradigma pemograman yang berorientasikan kepada objek. Semua data dan fungsi dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek [9].

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah tidak melihat bagaimana cara menyelesaikan masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap dalam sebuah departemen yang mememiliki manajer, sekrtaris , petugas administrasi data dan lainnya. Misalkan manajer tersebut ingin memperoleh data dari bagian administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manajer tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manajer bisa mendapatkan data tersebut melaui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri [9]

Pemograman orientasi objek menekankan konsep berikut :

1. *Class* – kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemograman berorientasi objek. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programer sekalipun terkait dalam sebuah domain permasalahan yang ada dan ide yang terdapat dalam sebuah *class* sebaiknya (*relative*) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

2. Objek (*Object*) – membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Abstraksi (*Abstract*) – Kemampuan sebuah program untuk melewati aspek informasi yang diprosesnya, yaitu kemampuan untuk memfokuskan pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaaanya dan berkomunikasi dengan objek lainya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

4. Engkapsulasi (*Encapsulation*) – Memastikan pengguna sebuah objek tidak dapat mengganti keadaaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaaanya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainya dapa berinteraksi dengannya. Objek lainya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. Polimorfime (*Polimorfism*) melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengriman pesan tergantung kepada objek tertentu dimana pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melaui penggunaan fungsi kelas-pertama.

6. Inheritas (*inheritance*) – Mengatur polismorfisme dan engkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada – objek-objek ini dapat membagi (memperluas) perilaku mereka tanpa harus mengimplementasikan ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memlikik inheritas).

2.10 Adobe Photoshoap

Adobe Photoshop, atau biasa disebut Photoshop, adalah perangkat lunak editor citra buatan Adobe Systems yang dikhususkan untuk pengeditan foto/gambar dan pembuatan efek. Perangkat lunak ini banyak digunakan oleh fotografer digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (market leader) untuk perangkat lunak pengolah gambar/foto, dan, bersama Adobe Acrobat, dianggap sebagai produk terbaik yang pernah diproduksi oleh Adobe Systems. Versi kedelapan aplikasi ini disebut dengan nama Photoshop CS (Creative Suite), versi sembilan disebut Adobe Photoshop CS2, versi sepuluh disebut Adobe Photoshop CS3 , versi kesebelas adalah Adobe Photoshop CS4 , versi keduabelas adalah Adobe Photoshop CS5 , dan versi yang terakhir (ketigabelas) adalah Adobe Photoshop CS6 [10].

Photoshop tersedia untuk Microsoft Windows, Mac OS X, dan Mac OS; versi 9 ke atas juga dapat digunakan oleh sistem operasi lain seperti Linux dengan bantuan perangkat lunak tertentu seperti *CrossOve*r [10].

2.11 Adobe Illustrator

Adobe Illustrator adalah produk pedamping dari Adobe Photoshop. Jika Photoshop ditujukan untuk memanipulasi foto digital dan gaya foto realistik ilustrasi komputer, sedangkan Illustrator ditujukan untuk kreasi logo, huruf dan desain grafis. Aplikasi ini banyak digunakan oleh desainer profesional, mulai desainer iklan, majalah, cover, banner sampai desainer web. Adobe Illustrator mendukung format RGB (Red, Green, Blue) dan CMYK (Cyan, Magenta, Yellow, Key - Black) yang biasa digunakan untuk proses pencetakan. Adobe Illustrator pertama kali dibuat pada tahun 1986, sebagai aplikasi pembuat desain grafis, huruf, dan logo. Sebenarnnya Adobe Illustrator diciptakan dengan tujuan

menggabungkan persamaan matematika untuk menciptakan garis halus dan melengkung serta bentuk objek yang dikenal sebagai kurva Bezier. Untuk menekankan gagasan yang mampu memproduksi dan mengedit dan menyimpan garis lengkung dan beralur, Adobe memilih gambar "The Birth of Venus" dari Sandro Botticelli untuk menjadi cover pemasaran mereka. Selama bertahun-tahun dan beberapa versi, Adobe menambahkan fitur-fitur baru dan perbaikan [11].

2.12 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.

UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object* Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM [12].

UML merupakan suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi.UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson.Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi.Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem [12].

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep

dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C [12].

2.12.1 *Diagram* UML

UML menyediakan 10 macam *Diagram* untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram* untuk memodelkan proses bisnis.
2. Conceptual *Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. Sequence *Diagram* untuk memodelkan pengiriman pesan (message) antar *object*s.
4. *Collaboration Diagram* untuk memodelkan interaksi antar *objects*.
5. *State Diagram* untuk memodelkan perilaku *object*s di dalam sistem.
6. *Activity Diagram* untuk memodelkan perilaku *Use Case* dan *objects* di dalam *system.*
7. *Class Diagram* untuk memodelkan struktur kelas.
8. *Object Diagram* untuk memodelkan struktur *object*.
9. *Component Diagram* untuk memodelkan komponen *object*.
10. Deployment *Diagram* untuk memodelkan distribusi aplikasi.

2.12.1.1 Usecase *Diagram*

Use case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, *create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Sebuah *use case* dapat menginklusi fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang diinklusi akan dipanggil setiap kali *use*

case yang menginklusi dieksekusi secara normal. Sebuah *use case* dapat diinklusi oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat mengekstensi *use case* lain dengan prilakunyanya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain [12].

2.12.1.2 Activity *Diagram*

Activity *Diagram*s menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity *Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Activity *Diagram* merupakan state *Diagram* khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity *Diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Decision digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (fork dan join) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. Activity *Diagram* dapat dibagi menjadi beberapa *object* swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [12].

2.12.1.3 Sequence *Diagram*

Sequence *Diagram* menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case: interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi [12].

2.12.1.4 Class *Diagram*

Class *Diagram* merupakan *Diagram* yang selalu ada di permodelan sistem berorientasi objek.Class *Diagram* menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Sebuah kelas mewakili sekelompok hal yang memiliki keadaan umum dan perilaku. Anda dapat menganggap kelas sebagai *blue print* untuk sebuah objek dalam suatu sistem berorientasi objek. Dalam UML, kelas adalah semacam classifier. Misalnya, Volkswagen, Toyota, Ford dan semua mobil, sehingga Anda dapat mewakili mereka menggunakan kelas bernama Mobil. Setiap jenis tertentu mobil adalah sebuah instance dari kelas, atau benda. Sebuah kelas dapat mewakili sebuah konsep yang nyata dan konkret, seperti faktur, dan bisa sesuatu yang abstrak, seperti dokumen atau kendaraan (sebagai lawan faktur, atau sepeda motor lebih dari 1000 cc), atau mungkin mewakili konsep tidak berwujud seperti strategi investasi berisiko tinggi [12].