BAB 2
TINJAUAN PUSTAKA

2.1 Tanaman

Tanaman adalah makhluk hidup yang tidak dapat berpindah tempat dan memproduksi makanannya sendiri. Sangat berbeda dengan hewan terutama manusia yang menggantungkan hidupnya dengan mahluk hidup lainnya, dengan bantuan sinar matahari makanan tanaman diproduksi sendiri menggunakan unsur-unsur anorganik yang terdapat di tempat sekitar mereka hidup. Proses ini disebut fotosintesis dan dilakukan oleh semua jenis tanaman. Sifat tanaman ini disebut autotrof, dan karena sifat inilah tanaman selalu ditempatkan di tempat pertama di setiap rantai makanan mahluk hidup [6].

2.1.1 Tanaman Obat

Tanaman obat adalah tanaman khusus yang berkhasiat sebagai obat. Biasanya, di lingkungan pedesaan setiap rumah memiliki tanaman-tanaman yang dapat digunakan sebagai bahan pengobatan herbal atau yang biasa dikenal sebagai "apotek hidup". Tanaman-tanaman obat yang biasa ada di perumahan anatara lain lidah buaya, kunyit, kencur, daun mangkokan, dan lain-lain. Tidak sedikit mereka yang menggunakan tanaman obat sebagai usaha keluarga. Jadi, selain untuk pengobatan dalam keluarga, juga dapat menambah penghasilan keluarga [6].

2.1.1.1 Penggolongan Tanaman Obat

Berdasarkan bahan yang dimanfaatkan untuk pengobatan, tanaman obat dapat digolongkan menjadi beberapa, yaitu sebagai berikut [6]:

1. Tanaman obat yang dapat diambil daunnya, misalnya daun salam, daun sirih, daun radu, dan lain-lain.
2. Tanaman obat yang diambil batangnya misalnya kayu manis, brotowali, pulasari, dan lain-lain.
3. Tanaman obat yang diambil buahnya, misalnya jeruk nipis, ketumbar, belimbing wuluh, dan lain-lain.
4. Tanaman obat yang diambil dari bijinya, misalnya kecubung, pinang, pala, dan lain-lain.

5. Tanaman obat yang diambil akarnya, misalnya pepaya, aren, pulai pencak, dan lain-lain.
6. Tanaman obat yang diambil umbi atau rimpangnya, misalnya kencur, jahe, bengle, dan lain-lain.

2.2 Game

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Dalam *game*, ada target-target yang ingin dicapai pemainnya. 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 [7].

Tujuan *game* adalah untuk menghibur, *game* tidak hanya disukai oleh anak-anak bahkan orang dewasa pun banyak yang memainkan *game*. *Game* sebenarnya penting untuk perkembangan otak, karena di dalam *game* terdapat berbagai masalah atau konflik yang menuntut kita untuk menyelesaikannya secara cepat dan tepat. Tetapi *game* juga bisa berdampak negatif jika penggunanya sudah kecanduan hingga lupa waktu dan akan mengganggu aktivitas lain.

2.2.1 Sejarah Singkat Game

Di antara jajaran nama tokoh penemuan penting, nama besar seorang penemu permainan komputer, atau lebih lazim kita menyebutnya dengan *game*, jarang sekali disebutkan. Namun bila penemuannya sampai sekarang masih digunakan dan bahkan sangat diminati dan dinikmati. Sebagian orang bahkan saat ini mengangap *game* sebagai sebuah kebutuhan yang harus terpenuhi.

Game pertama kali dibuat pada tahun 1966 oleh Ralph Baer bersama dengan timnya yang berjumlah 500 orang yang terdiri dari para insinyur dan teknisi. Permainan yang pertama sekali ditemukan hanya dapat dimainkan oleh komputer

seharga US$ 40.000. Dimana pengembangan dan pembuatan *game* tersebut didanai oleh Pentagon.

Pada tahun 1965, pihak militer datang kepada Bear dan meminta menciptakan simulasi komputer yang dapat membantu pasukan untuk belajar strategi dan mengukur kemampuan reflek. Proyek ini dikerjakan dengan tingkat keamanan yang sangat ketat di tengah situasi perang dingin. Setelah sebulan bekerja keras, Bear berhasil menampilkan dua titik putih yang berkejar-kejaran di layar. Hal ini membuat pihak militer merasa kagum dan memberikan dana yang lebih besar lagi sehingga ia dapat menyewa asisten lebih banyak. Tim ini kemudian berhasil membuat permainan antara papan dan bola yang pada akhir tahun 1966 dipresentasikan di depan para pejabat Pentagon. Namun sayang, ternyata mereka tak merasa tertarik dengan penemuan Bear. Hingga akhirnya ia berusaha memperoleh izin agar dapat memproduksi mesin permainan secara komersial.

Tahun 1970, Bill Enders yang tergabung dalam sebuah organisasi dengan nama Megnavox mencoba menyakinkan edukatif Megnavox untuk memberikan kesempatan pada Bear dan mesin permainannya. Hasilnya adalah munculnya video *game* komersial pertama, yaitu Magnavox Odyssey.

Tahun 1972, muncul nama baru dalam dunia *game*. Nolan Bushnel. Pada tanggal 27 Juni 1972, ia mendirikan perusahaan Atari, dan membuat *game* arcade Pong. Era baru dalam perkembangan dunia *game* terjadi pada tahun 1988, yang di dominasi oleh perusahaan di Jepang. Nintendo, yang awalnya hanya memproduksi mesin *photocopy*, beralih memproduksi *game*. Saat ini permainan komputer telah berkembang sedemikian pesatnya seiring dengan perkembangan hardware yang mendukung. Dengan program permainan yang lebih kompleks dan tampilan grafis 3D yang luar biasa [8].

2.2.2 Jenis Jenis Game

Berikut ini adalah jenis-jenis dari *game*:

Berdasarkan *platform* atau alat yang digunakan [7]:

1. *Arcade game*, yaitu *game* dengan mesin yang memang khusus didesain untuk jenis *game* tertentu dan memiliki fitur seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil dan lain-lain.

2. *PC game*, yaitu *game* yang dimainkan menggunakan *Personal Computer*.
3. *Console game*, yaitu *game* yang dimainkan menggunakan *console* tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.
4. *Handheld game*, yaitu *game* yang dimainkan di *console* khusus *game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.
5. *Mobile game*, yaitu *game* yang dapat dimainkan khusus untuk *mobile phone*, *smartphone* atau PDA.

Berdasarkan genre permainannya:

1. *Shooting* (tembak-tembakan) : *Shooting* merupakan salah satu *genre* dalam *game* yang biasanya mensimulasikan tembak-tembakan , atau hajar-hajaran bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya, *game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga *timing*, inti dari *game* jenis ini adalah tembak. Contoh : Medal of Honor, Counter Strike dan Fallout.



Gambar 2.1 Contoh Game Shooting Medal of Honor

2. *Fighting* (Pertarungan) : Jenis *game* ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan *skill* karakter yang dimainkan, 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. Contoh : Mortal Kombat dan Tekken.



Gambar 2.2 Contoh Game Fighting Mortal Kombat

3. *Adventure* (Petualangan) : Refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. *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. Contoh : Kings Quest dan Space Quest.



Gambar 2.3 Contoh Game Adventure Kings Quest

4. Simulasi, Konstruksi, Manajemen : *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. *Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh : The Sims, Flight Simulator dan Sim City.



Gambar 2.4 Contoh Game Simulasi The Sims

5. *Role Playing* : *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. Contoh : Final Fantasy, Dragon Age dan The Witchers.



Gambar 2.5 Contoh Role Playing Game Final Fantasy XIII

6. Strategi : Kebalikan dari *game* jenis *action* yang berjalan cepat dan perlu refleks secepat kilat, *game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. *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 pusat 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 melakukan *upgrade* pasukan atau teknologi. Contoh : Warcraft dan Age of Mythology.



Gambar 2.6 Contoh Game Strategi Age of Mythology

7. *Sport* (Olahraga) : Biasanya *game* genre ini permainannya diusahakan serealistik mungkin, dibuat semirip mungkin dengan aslinya walau kadang ada yang menambah unsur fiksi dalam *game* olahraga tersebut. Contoh : Winning Eleven dan NBA.



Gambar 2.7 Contoh Game Sport NBA

8. *Racing* (Balap) : *Racing Game* adalah *game* sejenis balapan yang memungkinkan kita untuk mengendalikan sebuah kendaraan untuk memenangkan sebuah kompetisi balap. Contoh : Need For Speed, Moto GP dan Grand Turismo.



Gambar 2.8 Contoh Game Racing Need For Speed

9. *Musical* : *Game* jenis ini adalah *game* yang memainkan suatu alat musik. Contoh: Guitar Hero dan Synthesia.



Gambar 2.9 Contoh Game Musical Guitar Hero

10. *Edugames* (Edukasi) : *Game* jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, baik itu untuk belajar 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. Contoh *edugames* : Bobi Bola, Dora the explorer, Petualangan Billy dan Tracy.



Gambar 2.10 Contoh Game Petualangan Billy dan Tracy

2.3 Game Edukasi

Educational game atau *game* edukasi adalah permainan yang dirancang dan dibuat untuk merangsang daya pikir termasuk meningkatkan konsentrasi dan memecahkan masalah. Definisi lain menyebutkan *game* edukasi adalah salah satu genre *game* yang digunakan untuk memberikan pengajaran atau menambah pengetahuan dan keterampilan melalui suatu media unik dan menarik [7].

Pemanfaatan *game* sebagai media pembelajaran belum lama dikembangkan di Indonesia. Padahal *game* menawarkan bentuk pembelajaran langsung dengan pola *learning by doing*. Pembelajaran yang dilakukan merupakan suatu konsekuensi dari sang pengguna *game* untuk dapat melalui tantangan yang ada dalam suatu permainan edukasi tersebut. Pembelajaran diperoleh dari faktor kegagalan yang telah dialami pengguna, sehingga mendorong pengguna untuk tidak mengulangi kegagalan di tahapan selanjutnya.

Selain itu pembelajaran yang dilakukan dalam sebuah *game* merupakan suatu konsekuensi dari sang pemain *game* untuk dapat melalui tantangan yang ada dalam suatu permainan *game* yang dijalankan. Dengan demikian maka *game*

menawarkan suatu bentuk media dan metode yang menakjubkan. *Game* mempunyai potensi yang sangat besar dalam membangun motivasi pada proses pembelajaran. Berbeda dengan pada penerapan metode konvensional, untuk menciptakan motivasi belajar sebesar motivasi dalam *game* [7].

2.4 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri dan digunakan oleh berbagai macam perangkat *mobile*. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk telepon seluler. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan *hardware*, *software*, 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. 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) [9].

2.4.1 Sejarah Android

Pada Juli 2000, Google bekerjasama dengan Android Inc., perusahaan yang berada di Palo Alto, California Amerika Serikat. Para pendiri Android Inc. bekerja pada Google, di antaranya Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Saat itu banyak yang menganggap fungsi Android Inc. hanyalah sebagai perangkat lunak pada telepon seluler.

Sejak saat itu muncul rumor bahwa Google hendak memasuki pasar telepon seluler. Di perusahaan Google, tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh *kernel* Linux. Hal ini menunjukkan indikasi bahwa Google sedang bersiap menghadapi persaingan dalam pasar telepon seluler. Versi android terbaru yaitu Android versi

3.0. juga sudah bergabung dengan beberapa smart mobile seperti Nokia, Sony Ericsson, dan lainnya [9].

2.4.2 Arsitektur Android

Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut [9]:

1. *Applications* dan *Widgets*

   Applications dan *Widgets* ini adalah *layer* dimana kita berhubungan dengan aplikasi saja dimana biasanya kita *download* aplikasi kemudian dengan kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semuanya aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Applications Frameworks*

   Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembangnya atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur alarm, dan menambahkan status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju *API framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*).

   Komponen-komponen yang termasuk di dalam *Applications Frameworks* adalah sebagai berikut :

   a. *Views.*
   b. *Content rovider.*
   c. *Resource Manager.*
   d. *Notification Manager.*
   e. *Activity Manager.*

3. *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 dan *security*.
g. *libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API‟s.

2.4.3 Versi Android

Berikut adalah berbagai versi Android sebagai berikut [7]:

1) Versi 1.1



Gambar 2.11 Android 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 dengan Gmail, dan pemberitahuan email.

2) Versi 1.5 (*Cupcake*)



Gambar 2.12 Android Cupcake

Pada pertengahan Mei 2009, Google kembali merilis 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 menonton 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, animasi layar, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

3) Versi 1.6 (Donut)



Gambar 2.13 Android Donut

Versi 1.6 dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding 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 yang dintegrasikan, Gestures, dan Text-to-speech engine, kemampuan dial kontak, teknologi *text to change speech*.

4) Versi 2.0/2.1 (*Eclair*)



Gambar 2.14 Android Eclair

Pada 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, 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. Dengan semakin berkembangnya dan semakin bertambahnya jumlah *handset* Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah Shazam, Backgrounds, dan WeatherBug. Sistem operasi Android dalam situs Internet juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh MySpace dan Facebook.

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



Gambar 2.15 Android Frozen Yogurt

Pada 20 Mei 2010, Android versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan *rendering* pada *browser*,

pemasangan aplikasi dalam SD *Card*, kemampuan WiFi Hotspot portabel, dan kemampuan *auto update* dalam aplikasi Android Market.

6) versi 2.3 (*Gingerbread*)



Gambar 2.16 Android 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, peningkatan fungsi *copy paste*, layar antar muka *(User Interface)* didesain ulang, dukungan format video VP8 dan WebM, efek audio baru *(reverb, equalization, headphone virtualization,* dan *bass boost)*, dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu.

7) versi 3.0/3.1 (*Honeycomb*)



Gambar 2.17 Android Honeycomb

Android *Honeycomb* dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi prosesor dan juga akselerasi perangkat keras *(hardware)* untuk grafis. Tablet pertama yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

8) Versi 4.0 (*Ice Cream Sandwich*)



Gambar 2.18 Android Ice Cream Sandwich

Android *Ice Cream Sandwich* yang dirilis pada 19 Oktober 2011 membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan data dan kontrol terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari *email* secara *offline*, dan berbagi informasi dengan menggunakan NFC.

9) Versi 4.1 (Jelly Bean)



Gambar 2.19 Android Jelly Bean

Android Jelly Bean yang diluncurkan pada acara Google I/O lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya meningkatkan input *keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat.

2.5 Multimedia

Multimedia adalah penggunaan komputer untuk menyajikan dan menggabungkan teks, suara, gambar, animasi dan video dengan alat bantu dan koneksi (*link*) sehingga pengguna dapat bernavigasi, berinteraksi, berkarya dan berkomunikasi. Multimedia sering digunakan dalam dunia hiburan. Selain dari

dunia hiburan, Multimedia juga diadopsi oleh dunia *game*. Multimedia juga dapat diartikan sebagai penggunaan beberapa media yang berbeda dalam menyampaikan informasi berbentuk teks, grafik, *audio*, animasi, dan video.

2.5.1 Elemen Multimedia

Komponen multimedia terbagi atas lima jenis yaitu [10]:

1. Teks

   Teks merupakan elemen multimedia yang menjadi dasar untuk menyampaikan informasi, karena teks adalah jenis data yang paling sederhana dan membutuhkan tempat penyimpanan yang paling kecil. Teks merupakan cara yang paling efektif dalam mengemukakan ide-ide kepada pengguna, sehingga penyampaian informasi akan lebih mudah dimengerti oleh masyarakat. Jenis-jenis teks seperti *Printed Text*, yaitu teks yang dihasilkan oleh *word processor* atau *word editor* dengan cara diketik yang nantinya dapat dicetak. *Scanned Text* yaitu teks yang dihasilkan melalui proses *scanning* tanpa pengetikan dan *Hypertext* yaitu jenis teks yang memberikan link ke suatu tempat atau meloncat ke topik tertentu.

2. Grafik

   Sangat bermanfaat untuk mengilustrasi informasi yang akan disampaikan terutama informasi yang tidak dapat dijelaskan dengan kata-kata. Jenis-jenis grafik seperti *bitmap* yaitu gambar yang disimpan dalam bentuk kumpulan pixel, yang berkaitan dengan titik-titik pada layar monitor. *Digitized picture* adalah gambar hasil rekaman video atau kamera yang dipindahkan ke komputer dan diubah ke dalam bentuk *bitmap*. *Hyperpictures*, sama seperti *hypertext* hanya saja dalam bentuk gambar.

3. *Audio*

   Multimedia tidak akan lengkap jika tanpa *audio* (suara). *Audio* bisa berupa percakapan, musik atau efek suara. Format dasar *audio* terdiri dari beberapa jenis :

   a. *Wave*

      Merupakan format file digital audio yang disimpan dalam bentuk digital dengan eksistensi WAV.

b. MIDI

MIDI (*Musical Instrument Digital Interface*) memberikan cara yang lebih efisien dalam merekam musik dibandingkan *wave*, kapasitas data yang dihasilkan juga jauh lebih kecil. MIDI disimpan dalam bentuk MID.

4. Video

Video menyediakan sumber yang kaya dan hidup untuk aplikasi multimedia. Dengan video dapat menerangkan hal-hal yang sulit digambarkan lewat kata-kata atau gambar diam dan dapat menggambarkan emosi dan psikologi manusia secara lebih jelas.

5. Animasi

Animasi adalah simulasi gerakan yang dihasilkan dengan menayangkan rentetan *frame* ke *layer*. *Frame* adalah satu gambar tunggal pada rentetan gambar yang membentuk animasi.

Animate adalah untuk membuat sesuatu hidup, sebagian orang mengira bahwa animasi itu sama dengan *motion* (gerakan), tetapi animasi mencakup semua yang mengandung efek visual sehingga animasi mencakup perubahan posisi terhadap waktu, bentuk, warna, struktur, tekstur dari sebuah objek, posisi kamera, pencahayaan, orientasi dan *focus* dan perubahan dalam teknik *rendering*.

2.6 Unified Modelling Language (UML)

Pemodelan (*modeling*) adalah proses merancang piranti lunak sebelum melakukan pengkodean (*coding*). Model piranti lunak dapat dianalogikan seperti pembuatan *blueprint* pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.

Dengan menggunakan model, diharapkan pengembangan perangkat lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability, robustness*, *security*, dan sebagainya.

Kesuksesan suatu pemodelan perangkat lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan *tool* yang digunakan.

Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya (proses) akan membuat proyek gagal. Dan pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan *tool* yang tepat [11].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/ semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*) [11].

2.6.1 Use Case Diagram

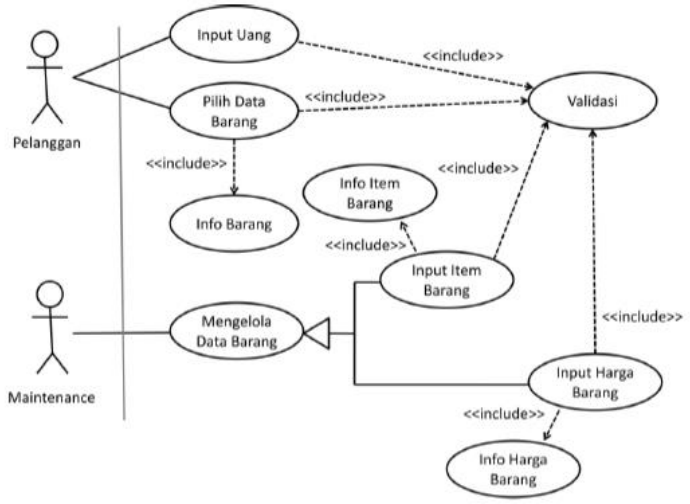
Diagram *use case* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem, yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Diagram *use case* dapat sangat membantu bila kita sedang menyusun kebutuhan sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat melakukan *include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang melakukan include dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain,

sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use* case lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain [11].



Gambar 2.20 Contoh Use Case Diagram

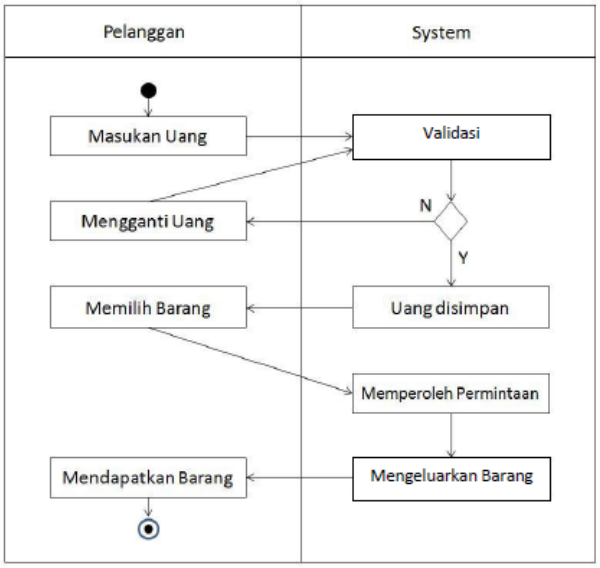
2.6.2 Activity Diagram

Activity diagram 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 sub sistem) 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.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan 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 [11].



Gambar 2.21 Contoh Activity Diagram

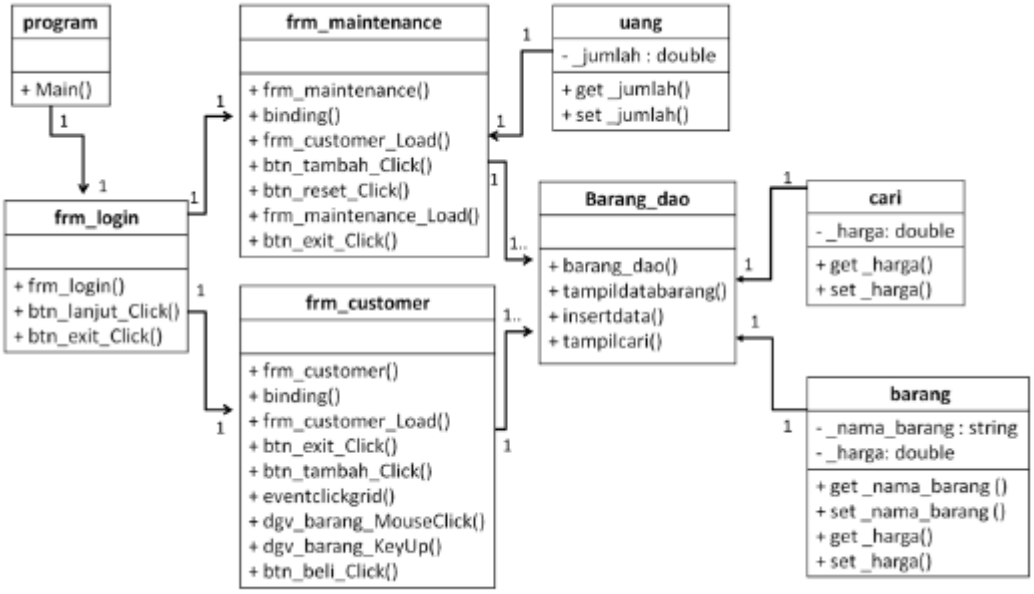
2.6.3 Class Diagram

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [11].

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung

diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.



Gambar 2.22 Contoh Class Diagram

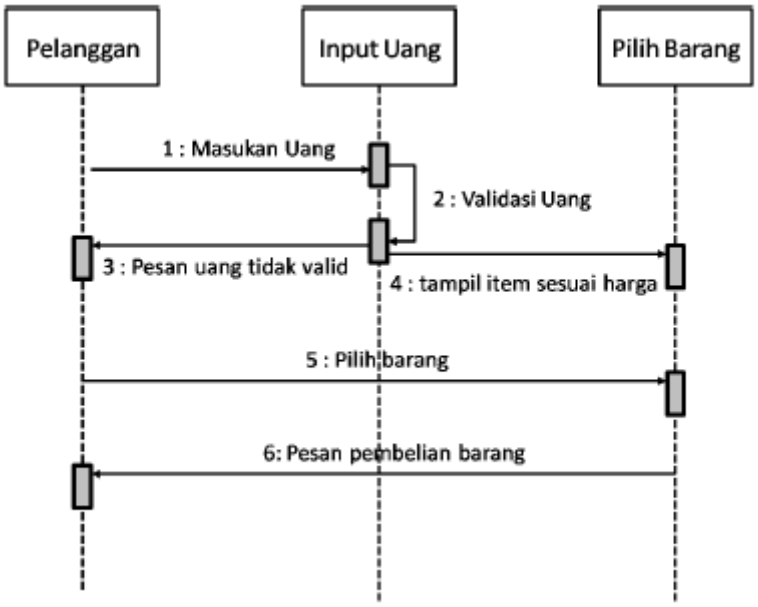
2.6.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* [11].

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity* [11].



Gambar 2.23 Contoh Sequence Diagram

2.7 Game Engine

Game Engine adalah sebuah perangkat lunak yang dirancang untuk membuat sebuah *game*. Sebuah *game engine* biasanya dibangun dengan mengenkapsulasi beberapa fungsi standar yang umum digunakan dalam pembuatan sebuah *game*. Misalnya, fungsi *rendering*, pemanggilan suara, *network*, atau pembuatan partikel untuk *special effect*. Sebagian besar *game engine* umumnya berupa *library* atau sekumpulan fungsi-fungsi yang penggunanya dipadukan dengan bahasa pemrograman.

Pada beberapa kasus *game engine* kadang pula disebut sebagai *middleware*. Hal ini dikarenakan karena *game engine* digunakan sebagai perantara antara bahasa pemrograman dengan format data dari berbagai perangkat lunak penghasil *asset*. Hal ini tentu saja memudahkan *developer* karena tidak perlu repot lagi memikirkan cara mengolah data yang dihasilkan dari berbagai perangkat lunak yang berbeda.

Sebuah *game engine* juga biasanya dilengkapi dengan fungsi-fungsi grafis atau komputasi dasar yang jarang diketahui oleh *developer*. *Game engine* membungkus itu semua untuk menjadikan proses pembuatan menjadi lebih cepat dan mudah. Beberapa *game engine* jenis ini diantaranya adalah TrueVision, Irrlicht dan Ogre3D [12].

2.7.1 Perkembangan Game Engine

Di awal tahun 2000, *game engine* mengalami perkembangan yang cukup signifikan. Beberapa *game engine* mulai dilengkapi dengan World Editor. Sehingga alih-alih menggunakan perangkat lunak 3D seperti 3Dmax atau Blender, pembuatan level atau 'dunia' *game* dapat dibuat melalui sebuah perangkat lunak tersendiri yang telah dirancang khusus untuk *game engine* tersebut.

Perkembangan ini bukannya tanpa masalah. Di satu sisi, *software* ini memudahkan para *developer* pemula yang membutuhkan kecepatan untuk melihat produk dari rancangan mereka. Tapi di sisi lain perangkat lunak ini bisa 'membatasi' kemampuan *developer* untuk berkreasi lebih jauh. Sehingga bagi sebagian *developer*, membangun 'dunia' *game* lewat *script* tetap puluhan kali lebih *powerfull.*

Terlepas dari kelebihan dan kekurangannya, World Editor sebagai bagian dari *game engine* tetap memiliki peran penting sebagai alat untuk mempercepat pembuatan sebuah level atau dunia. Selain itu, beberapa *game engine* bahkan telah membuat *Integrated Development Environment* (IDE) tersendiri untuk proses pengembangannya.

Dengan kata lain, *game engine* tersebut tidak lagi harus tergantung dengan *software development* seperti Delphi atau Ms. Visual C++ misalnya, karena *game engine* tersebut telah dilengkapi dengan *editor* dan *compiler* sendiri [12].

2.8 Tools yang digunakan

Tools yang digunakan dalam pembangunan game edukasi obat herbal berbasis android ini diantaranya adalah Adobe Photoshop dan Game Make Studio.

2.8.1 Adobe Photoshop

Photoshop adalah sebuah *image editor* atau program penyunting gambar yang berfungsi untuk membuat, menyunting, dan memodifikasi gambar-gambar digital yang terdapat di dalam komputer.

Photoshop merupakan program penyunting gambar yang paling hebat dan paling popular hingga saat ini. Kemampuan serta fasilitasnya yang lengkap membuatnya diminati oleh para seniman, profesional maupun pemula yang membutuhkan sebuah program gambar yang lengkap namun mudah dalam penggunaannya.

Photoshop saat ini digunakan untuk membuat gambar-gambar untuk keperluan seperti iklan, brosur, poster, seta berbagai macam *output* cetak lainnya. Bagi para penggemar photografi, photoshop juga dapat digunakan untuk me-*retouch* foto yang sudah rusak hingga memodifikasi foto menjadi lebih baik [13].

2.8.2 Game Maker Studio

Game maker adalah salah satu *game engine* yang bisa membantu menjadi penggembang *game* komputer tanpa menggunakan *coding* yang banyak. Game Maker merupakan suatu program yang bisa dikatakan cukup *simple* dan mudah untuk digunakan. Dengan program ini dapat menciptakan *game* jenis apa saja yang anda inginkan seperti: *Racing*, *Action*, *Fighting*, *RPG*, *Adventure*, *Puzzle*, *Tactic*, *Simulation*. Program ini lebih fokus pada *game* berjenis 2D tetapi program ini juga menyediakan fitur yang memungkinkan untuk membuat *game* jenis 3D [14].



Gambar 2.24 Logo Game Maker Studio

Game Maker dirancang untuk memungkinkan pengguna untuk dengan mudah mengembangkan permainan komputer tanpa harus belajar yang kompleks bahasa

pemrograman seperti C++ atau Java. Namun, adalah mungkin untuk membuat *add-ons* untuk *game* dibuat dengan Game Maker. Ekstensi tersebut, sebagaimana dikenal untuk pengguna Game Maker, memberikan pengguna seperti 3D maju melalui Port Rendering Mesin Ogre, suara yang lebih baik. Ekstensi untuk IDE utama adalah terbatas pada penciptaan *drag* dan *drop* naskah baru tindakan.

Untuk pengguna yang berpengalaman dengan pengalaman pemrograman komputer, Game Maker berisi *built-in* bahasa pemrograman *scripting* yang disebut GML (Game Maker Language), yang memungkinkan permainan yang lebih kompleks harus dibuat dengan program.

Game Maker awalnya dikembangkan oleh Prof Mark Overmars pada tahun 1999. Game Maker adalah sebuah program yang digunakan untuk membuat sebuah *game* yang khusus untuk PC. Game Maker menggunakan bahasa GML atau Game Maker Language. Namun, pengguna juga bisa menggunakan LIB atau *Libraries*. *Libraries* adalah *action-action* yang memang telah tersedia di Game Maker.



Gambar 2.25 Mark Overmars

Object-Oriented Programming (OOP) sudah menjadi pemrograman yang digunakan sekarang ini. Bahasa pemrograman seperti Java, C#, dan GML adalah salah satu dari OOP. Salah satu fitur dari sebuah OOP adalah bahwa memusatkan desain pada objek berdasarkan pada skenario dunia nyata. Hal ini menegaskan keadaan (*state*), behaviour dan interaksi dari objek. Game Maker merupakan salah satu dari OOP karena semua dipusatkan kedalam entitas, seperti *objects*, *sprites*, dan *room*. Dalam kenyataan, semua *resource* yang ada dalam Game Maker adalah sebuah objek karena mereka membawa informasi dan dapat menjadi *instance*.

Di dalam Game Maker terdapat baberapa *resource* yang dapat gunakan untuk membuat sebuah *game*:

1. *Sprites*

   Sprite adalah gambar dari semua objek dalam game. *Sprite* bisa terdiri dari satu gambar saja atau lebih dari satu gambar sehingga terlihat seperti gerakan animasi.

2. *Sound*

   Dengan *sound* dapat menyertakan suara latar atau efek suara sehingga *game* terasa lebih hidup.

3. *Background*

   Background adalah gambar besar yang digunakan untuk latar belakang sebuah *game* yang berlangsung.

4. *Path*

   Path adalah lintasan yang diikuti oleh suatu objek dalam *game*. dengan *Path* dapat menentukan kearah mana sebuah objek bergerak seperti pergerakan musuh dalam *game*.

5. *Script*

   Script dalam Game Maker tidak terlalu banyak hanya beberapa baris kode program. terdapat dua cara membuat program, yaitu dengan cara *drag-and-drop*, dan cara membuat *script* program. *Script* merupakan koleksi dari fungsi-fungsi *statement* program yang dapat dijalankan dari berbagai objek, atau berbagai *script* kode.

6. *Font*

   Font digunakan untuk menambahkan *font* yang akan di gunakan dalam *game*. *Font* biasanya digunakan dengan fungsi *text drawing*.

7. *Timelines*

   Dengan *timelines* dapat memberikan waktu pada suatu objek dalam melakukan sebuah *event* atau *action* pada *game* dengan waktu yang sudah ditentukan.

8. *Objects*

   Objek merupakan hal terpenting dalam game. Objek merupakan benda hidup yang bertindak dalam permainan. Sebuah objek diberikan pergerakan melalui

event yang harus direaksi oleh game, dan bagaimana seharusnya bereaksi. Sebuah objek biasanya diwakili oleh sebuah *sprite*.

9. *Rooms*

   Room adalah ruangan dimana *game* berjalan dan juga sebagai tempat untuk suatu objek untuk di letakan atau tempat suatu objek untuk bergerak

2.8.2.1 Game Maker Language

Game Maker Language atau disingkat GML adalah bahasa pemrograman yang *built-in* dalam IDE Game Maker. Dengan sedikit saja kita menguasai bahasa GML maka kita akan mampu membuat *game* yang professional. Ada beberapa perbedaan dalam tempat mengetikan *script* dalam GML. Pertama, ketika mendefinisikan *script*. Kedua, ketika menambahkan kode aksi kepada sebuah *event* [14].

2.9 Metode Pengujian Black-Box

Pada siklus hidup pengembangan perangkat lunak (*System Development Life Cycle: SDLC*) salah satu suatu proses yang harus dilakukan adalah proses pengujian pengujian (*testing*). Pengujian perangkat lunak adalah suatu teknik yang digunakan menguji apakah sebuah perangkat lunak yang dihasilkan telah memenuhi kebutuhan proses bisnis pengguna atau masih belum. Menurut Pressman, *testing* adalah proses eksekusi suatu program untuk menemukan kesalahan sebelum digunakan oleh pengguna akhir (*end-user*) [15].

Salah satu metode pengujian perangkat lunak adalah *Black-Box Testing. Black-box Testing* merupakan sebuah metode yang digunakan untuk menemukan kesalahan dan mendemonstrasikan fungsional aplikasi saat dioperasikan, apakah input diterima dengan benar dan *output* yang dihasilkan telah sesuai dengan yang diharapkan.

Fokus dari pengujian mengunakan metode *Black-Box* adalah pada pengujian fungsionalitas dan *output* dihasilkan aplikasi. Pengujian *black-box* didesain untuk mengungkap kesalahan pada persyaratan fungsional dengan mengabaikan mekanisme internal atau komponen dari suatu program.

Pengujian perangkat lunak mempunyai beberapa *level*, untuk pengujian menggunakan metode *Black Box*, terdapat enam *level* yaitu *Integration*, *Functional*, *System*, *Acceptance*, *Beta*, dan *Regression.* Salah satu dari pengujian *Black-Box* yang dapat dilakukan oleh seorang penguji independen adalah *Functional testing.* Basis uji dari *functional testing* ini adalah pada spesifikasi dari komponen perangkat lunak yang akan diuji. *Functional testing* memastikan bahwa semua kebutuhan-kebutuhan telah dipenuhi. Dengan demikian fungsinya adalah tugas-tugas yang didesain untuk dilaksanakan. *Functional testing* berkonsentrasi pada hasil dari proses, bukan bagaimana prosesnya terjadi [16].