BAB II

LANDASAN TEORI

II.1. Perancangan

Perancangan adalah penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari berapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi Perancangan sistem dapat dirancang dalam bentuk bagan alir sistem (*system flowchart*), yang merupakan alat bentuk grafik yang dapat digunakan untuk menunjukan urutan-urutan proses dari sistem.(Muhammad Fadlan, 2014:151)

II.2. Aplikasi

Aplikasi adalah penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan. Aplikasi dapat diartikan juga sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi *software* yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu: (Muhammad Fadlan, 2014:151)

1. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
2. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

II.3. Sejarah *Game*

Game atau permainan dapat diklasifikasikan menjadi dua bagian besar *game* fisik dan *game* elektronik. *Game* fisik mungkin sudah sering kita lakukan dalam kehidupan sehari-hari sewaktu masih anak-anak. Seperti lompat tali, petak umpet dan sebagainya. Dan *game* elektronik merupakan fenomena yang sangat menarik saat ini. Bahkan dapat dikatakan bahwa hampir semua kalangan menyukai *game* elektronik. *Electronic Game* atau selanjutnya dapat disebut sebagai *Video Game* pertama sekali ditemukan oleh *Thomas T. Goldsmith Jr* dan *Estle Ray Mann*. Penemuan ini dipatenkan pada januari 1974. Yang mendasari perkembangannya saat ini adalah ketika mereka menemukan *Cathode-Ray* sebuah tabung *vacum* yang digunakan sebaai mendia untuk membuat simulasi kecepatan tembakan dan arah tembakan sebuah roket. Pada februari 1951, *Christopher Strachey* memulai pengembangannya ke arah pemrograman yang mulai menggunakan memori di mana aplikasinya diterapkan untuk kebutuhan para pilot. Dan penemuan baru terus berkembang hingga tahun 1959. Namun era perkembangan konsol *game* di mulai setelah masa ini. (*Ivan C. Sibero ,2009:9*)

II.3.1. Generasi Perkembangan *Game*

II.3.1.1 Generasi Pertama (197201977)

Generasi pertama ini diawali dari ide *Ralph Baer* seorang teknisi televisi. Dia memulai membuat *game* dua pemain sederhana yang bernama *chese* dimana 2 buah titik saling berkejaran. Generasi pertama ini merupakan awal interaktif *game.* Perkembangan penemuan *Baer* ini membuat beberapa penemuan lain mulai ikut bergabung dengannya sehingga ditemukan sebuah mesin baru yang bernama

Magnavox dan mulai dirilis pada tahun 1972. (*Ivan C. Sibero, 2009:10*)

II.3.1.2 Generasi Kedua (1976-1983)

Generasi kedua ini sering juga disebut generasi konsol *bit*. Diamana pemrograman *video game* dibuat lebih *advance. Farichild Channel F* merupakan sebuah produk yang dirilis sekitar tahun 1976. Tahun 1977 atari memulai konsol baru dengan dasar *CPU*. Merena menamakan produk ini *Video Computer System* (*VCS*) yang dikenal dengan *ATARI 2600*. Pada masa itu, *ATARI* cukup dikenal masyarakat dibandingkan dengan produk lainnya. *Magnavox* malah mulai dengan konsol *base CPU* pada tahun 1978 dengan produknya *Odyssey 2*. Di Amerika dan Kanada, *Philips Electronics*, sebuah perusahaan yang cukup dikenal hingga saat ini, mulai memproduksi *Philips G7000* dan *Interton VC 4000*. Namun begitu banyak konsol baru yang muncul pada saat itu, *ATARI* tetap masih menjadi magnet *game* bagi masyarakat dengan *arcade game* nya yang cukup dikenal *Space Invaders*. Walaupun dengan keterbatasan *hardware* memori yang sangat minim sekitar 2-31 kb, namun perkembangannya cukup pesat. (*Ivan C. Sibero, 2009:11*)

II.3.1.3 Generasi Ketiga (1983-1992)

Jika sebelumnya perkembangan *game* banyak dilakukan di Eropa dan Amerika, pada generasi ini Jepang juga mulai merilis konsol barunya, *Famicom*, dan berubah nama menjadi *Nintendo*. *Nintendo Entertainment System (NES)* mendominasi pasarnya hingga Amerika Utara, selain *NES*, *Sega Master System* juga berkembang pesat di Eropa dan Brasil. Dan *Atari* pun membuat produk

barunya *ATARI 7800*. Pada akhir generasi ini, *Nintendo* memproduksi *Game Boy* yang cukup fenomenal hingga bertahan sampai 15 tahun lebih. (*Ivan C. Sibero, 2009:12*)

II.3.1.4 Gerenasi Keempat (1987-1996)

Sejarah generasi ini lebih dikenal dengan generasi 16 bit dimana pengembangannya dimulai sekitar Oktober 1987. Generasi awal *PC* yang dikembangkan *Nippon Electric Company's* (*NEC*). Produknya yang cukup dikenal di Amerika yaitu *TurboGrafx-16*. Walaupun *NEC* mulai dengan konsol barunya *Nintendo* dan *Sega* masih tetap menjadi konsol terbaik bagi para penggila *game.* (*Ivan C. Sibero, 2009:13*)

II.3.1.5 Generasi Kelima (1993 – 2002)

Konsol 64 bit mulai muncul pada era ini. Tidak banyak perubahan dari generasi sebelumnya. Pengembang konsol generasi keempat tetap eksis dengan produk terbarunya. Grafis *3D* mulai menjadi alasan untuk perang konsol ini. *Nintendo* dengan *Super Nintendo* dan *Sega* dengan *Sega Saturn*. Para pengembang *game* lebih menyukai membuat *game* untuk konsol *Nintendo* dan produk *Sony* yang cukup terkenal yaitu *Playstation. (Ivan C. Sibero, 2009:14)*

II.3.1.6 Generasi Keenam (1996-2006)

Generasi 128 bit. Teknologi semakin maju membuat beberapa konsol baru mulai muncul dan semuanya memiliki pasar sendiri. Namun para pabrikan konsol generasi sebelumnya sampai sekarang masih ikut dalam perang konsol ini. *PC* konsol masih menjadi lirikan para penggila *game.* Dengan kemajuan teknologi ini

membuat era ini perkembangan *game PC* lebih cepat dibanding generasi sebelumnya. Kartu grafis yang mulai memiliki kecepatan tinggi sehingga membuat para *gamer* dimanjakan. Para pengembang *game PC* juga mulai bertumbuh. Pada generasi ini, beberapa produk mutakhir mulai bermunculan seperti *Sony Playstation 2* dari *Sony*, *Sega's Dreamcast*, *Xbox* dari *Microsoft*, dan *Nintendo* dengan produk *GameCube*-nya. (*Ivan C. Sibero, 2009:15*)

II.3.1.7 Generasi Ketujuh (2004)

Pada masa ini beberapa pengembang konsol mulai fokus dengan pengembangan saja. Mulai tahun 2004 hanya beberapa perusahaan konsol yang masih bertahan dengan penembangannya seperti *Nintendo*, *Sony*, dan *Microsoft*. Diluar itu, perusahaan konsol mulai membuat konsol yang lebih menarik dan dengan fitur baru dan *game-game* barunya yang menarik. Beberapa konsol baru yang ada dipasaran saat ini masih dibuat oleh perusahaan konsol generasi sebelumnya, seperti *Nintendo Wii* oleh *Nintendo*, *Xbox 360* dari *Mricosoft* dan *Playstation 3* dari *Sony*. Perkembangan konsol ini juga mempengaruhi tumbuhya para pengembang *Software game. (Ivan C. Sibero, 2009: 17)*

II.3.2 *Personal Computer (PC) Game*

Perkembangan *PC game* pada masa perang konsol masih sangat lambat karena *PC* lebih sering digunakan untuk peralatan kantor.Namun perkembangan *hardware* komputer membuat para *game developer* melirik *PC* sebagai konsol *game*. Perlu diingat perkembangan *game PC* cukup baik pada era 80-an. *Game Digger* menjadi salah satu *game* yang cukup laris. Ingin bermain dengan tampilan

yang realistis dengan *genre* yang berbeda, ini mungkin jawaban bagi mereka yang mulai membuat *game PC*. (*Ivan C. Sibero, 2009:18*)

II.3.2.1 *Genre Game*

Genre game adalah klasifikasi *game* yang didasari interaksi pemainnya. Visualisasi juga menjadi ukuran klasifikasi *genre* ini. Namun untuk beberapa kasus pengembang *game* membuat kompilasi antar berbaai *genre* ini. Tentu saja variasi format *game* lebih banyak. (*Ivan C. Sibero, 2009:18*)

A. *Action*

Genre ini mungkin gaya permainan yang paling diminati para *player*. Dibutuhkan kecermatan reaksi waktu dan gerak. *Genre* ini memiliki banyak rintangan di dalamnya. Jenis *game* ini seperti menembak, perkelahian dan banyak lagi termasuk dalam *genre* ini. Berikut adalah beberapa diantaranya : (*Ivan C. Sibero, 2009:18*)

1. *Ball And Paddle*

Jenis *game* ini merupakan jenis *game* yang pertama kali dibuat. Bola dan penangkisnya menjadi alat dalam permainan ini. *Game Ping Pong* adalah salah satu contoh *genre game* ini. Memang terlihat sederhana, namun dapat dikembangkan dengan visualisasi yang lebih menarik, misal objek bola diganti menjadi buah-buahan atau lainnya. Contoh *game genre* ini adalah *Arkanoid* dan *Block Buster*. (*Ivan C. Sibero, 2009: 18*)

2. *Beat'em up, hack and slash*

Jenis *game* ini menekankan pada reaksi pemain. Berpetualang sambil bertempur merupakan aksi dari *genre* ini. Aksi pukul-tendangan, penggunaan pedang dan beberapa alat lain menjadi hiburan tersendiri buat pemain. Lawan pemain bisa berupa apa saja dan biasanya cukup banyak. Dilihat dari aksi yang cukup banyak ini dibutuhkan kontrol yang cukup banyak juga. *Double Dragon* merupakan *game* dari *genre Beat'em up, hack and slahs* yang cukup terkenal pada tahun 1987. (*Ivan C. Sibero, 2009: 19*)

3. *Fighting*

Jenis ini sekilas hampir mirip, namun perbedaan terlihat dari pertempurannya. Di sini pemain berhadapan satu lawan satu dengan musuh yang memiliki beragam keahlian. Karena karakter lawan digerakkan oleh komputer maka keahlian bertarung mereka menggunakan kontrol yang begitu banyak sampai harus dihafal. *Street Fighter* adalah *game* yang cukup populer dari *genre* ini yang sampai sekarang masih banyak dimainkan. (*Ivan C. Sibero, 2009:19*)

4. *Maze*

Genre ini sudah tidak asing lagi bagi para *gamer*, Beberapa pengembang *game* menyatakan bahwa *genre* ini cukup bertahan dan menjadi dasar utuk pengembangan beberapa *game*. Di awali dengan *game Pacman* dan *Digger* (*game PC* pertama), *game* ini cukup mudah untuk dimainkan, dan cukup menarik. Reaksi gerakan pemain harus cepat agar terhindar dari musuh menjadi dasar dari *game* ini. (*Ivan C. Sibero, 2009:20*)

5. *Pinball*

Jenis *game* ini merupakan aplikasi dari permainan *pinball table*. Kontrol *game* ini cukup sederhana, berupa dua buah lengan pemukul bola. Bola yang memantul memberikan nilai yang berbeda pada tiap zona. Mungkin sedikit sulit mencapai nilai yang tinggi karena bola yang dipukul tidak dapat dikontrol gerakannya. *(Ivan C. Sibero, 2009:20)*

II.4. *Game* Tebak Gambar Dan *Puzzle*

Tebak gambar adalah permainan asah otak ringan, kumpulan gambar disusun sedemikian rupa sehingga bisa menimbulkan sebuah kosakata baru yang diadaptasi dari istilah sehari-hari, ungkapan unik dan lucu, ataupun berupa isu dan peristiwa yang sedang terjadi. *Puzzle* yang merupakan permainan asah otak yang menantang ketrampilan pemainnya, sepertinya tidak pernah kehilangan popularitasnya dan tidak pernah termakan usia. *Puzzle* merupakan salah satu jenis *game* yang cukup memeras otak untuk menyelesaikannya.(*Ashari, 2014:1)*

II.5. Android

Android adalah sebuah sistem operasi untuk perangkat mobile bebasis linux yang mencakup sistem operasi, middleware, dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi. Antarmuka penguna android didasarkan pada manipulasi langsung, mengunakan masukan sentuh yang serupa dengan tindakan didunia nyata, seperti menggesek, mengetuk, mencubit, dan membalikkan cubitan untuk memenipulasi obyek pada layar. Android memiliki sejumlah besar komunitas pengembang

aplikasai (apps) yang memperluas fungsionalisasi perangkat, ummumnya ditulis dalam versi bahasa pemrograman Java yang telah di-*custom.(wahana komputer,2014:1-2).*

Android adalah istilah dalam bahasa Inggris yang berarti “Robot yang menyerupai manusia”. Logo “Android” sendiri, dicerminkan seperti sebuah robot berwarna hijau, yang mengacu kepada arti kata Android.

Gambar II.1. Logo Android

Sumber:(Beginning Android Programming with ADT Bundle: 2014: 2-4)

Android adalah sistem operasi yang bersifat open source (sumber terbuka). Disebut open source karena source code (kode sumber) dari sistem operasi Android dapat dilihat, di-download, dan dimodifikasi secara bebas. Paradigma open source ini memudahkan pengembangan teknologi Android, karena semua pihak yang tertarik dapat memberikan kontribusi, baik pada pengembangan sistem operasi maupun aplikasi.*(Alfa dan eva, 2014: 2-4).*

II.5.1 Versi dan Jenis-Jenis Android

Pengembangan android dimulai dengan berdirinya Android,Inc. pada Oktober 2003 dengan tujuan membuat *Mobile Device* yang lebih *Smart* untuk menyaingi Symbian dan Windows Mobile yang populer saat itu(Iphone dan Blackberry). Pada tahun 2005,Android Inc. Diakusisi oleh Google. Pengembangan terus dilanjutkan sampai Android versi beta diluncurkan pada tanggal 5 November 2007, bersamaan dengan berdirinya OHA (Open Handset Alliance). Sampai saat ini tanggal 5 November diperingati sebagai hari jadi Android. Seminggu setelahnya, pada tanggal 12 November 2007 Android SDK (Software Development Kit) diluncurkan, sehingga pengguna dapat membuat dan mengembangkan aplikasi Android mereka sendiri.

Android telah dikembangkan dan diupdate beberapa kali sejak rilis pertamanya. Tabel dibawah ini memperlihatkan versi Android semenjak pertama kali dirilis

Tabel II.1. Versi Android

Versi	Nama	Tanggal Rilis	Catatan
1.1	-	9 Februari 2009	-
1.5	Cupcake	30 April 2009	Mulai pakai kode nama
1.6	Donut	15 September 2009	-
2.0 / 2.1	Éclair	26 Oktober 2009	-
2.2	Froyo	20 Mei 2010	-
2.3	Gingerbread	6 Desember 2010	Masih banyak digunakan di smartphone jenis lama
3.0	Honeycomb	22 Februari 2011(3.0) 10	Hanya untuk tablet

		Mei 2011 (3.1) 15 Juli 2011 (3.2)	
4.0	Ice Cream Sandwich	19 Oktober 2011	*Smartphone* dan Tablet
4.1- 4.3	Jelly Bean	9 Juli 2012 (4.1) 13 November 2012(4.2) 24 Juli 2013 (4.3)	Update untuk memperbaiki dan menambah fitur-fitur ICS
4.4	Kit Kat	-	Diperkenalkan November 2013

(Sumber : Beginning Android Programming with ADT Bundle: 2014: 7)

II.5.2. Tool Pengembangan Android

1.Eclipse

Sejak berkurangnya standarisasi pengujian aplikasi pada android, membangun aplikasi pada *operating system* tersebut menjadi lebih mudah dan fleksibel dibandingkan sebelumnya. Sehingga semakin banyak programmer baru yang lahir dengan beragam kreatifitas dalam mengembangkan aplikasi. Dalam mengembangkan sebuah aolikasi sendiri, dibutuhkan software pendukung, Begitu pula dalam mengembangkan aplikasi pada android, software yang diperlukan seperti Eclipse sebagai IDE(*Integrated Development Environment)*atau program komputer dengan beberapa fasilitas yang diperlukan dalam mengembangkan perangkat lunak.*(Alfa dan Eva, 2014;12)*

2. *Android* SDK (*Software Development Kit*)

Android SDK adalah *tools* API *(Aplication Programming Interface)* yang diperlukan untuk mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java*. Beberapa fitur-fitur *Android* yang paling penting

adalah mesin *Virtual Dalvik* yang dioptimalkan untuk perangkat *mobile*, *integrated browser* berdasarkan *engine open source WebKit*, Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi *opengl* ES 1.0 (Opsional akselerasi perangkat keras), kemudian *SQLite* untuk penyimpanan data (*database*). Fitur-fitur *android* lainnya termasuk media yang mendukung *audio*, *video*, dan gambar, juga ada *fitur bluetooth*, EDGE, 3G dan WiFi, dengan fitur kamera, GPS, dan kompas. Selanjutnya fitur yang juga turut disediakan adalah lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*. (*Alicia Sinsuw ; 2013 : 2*)

3.AVD *(Android Virtual Device)*

Android Virtual Device merupakan emulator untuk menjalankan aplikasi *android*, yang tampilannya dapat dilihat pada gambar 1. Setiap AVD terdiri dari sebuah profil perangkat keras yang dapat mengatur pilihan untuk menentukan *fitur hardware emulator*. Misalnya, menentukan apakah menggunakan perangkat kamera, apakah menggunakan *keyboard* QWERTY fisik atau tidak, berapa banyak memori internal, dan lain-lain. AVD juga memiliki sebuah pemetaan versi *Android*, maksudnya kita menentukan versi dari *platform Android* akan berjalan pada emulator. Pilihan lain dari AVD, misalnya menentukan *skin* yang kita ingin gunakan pada emulator, yang memungkinkan untuk menentukan dimensi layar, tampilan, dan sebagainya. Kita juga dapat menentukan SD *Card virtual* untuk digunakan dengan di emulator. (*Alicia Sinsuw ; 2013 : 2*)

Gambar II.2. Tampilan AVD

Sumber : (*Alicia Sinsuw ; 2013 : 3*)

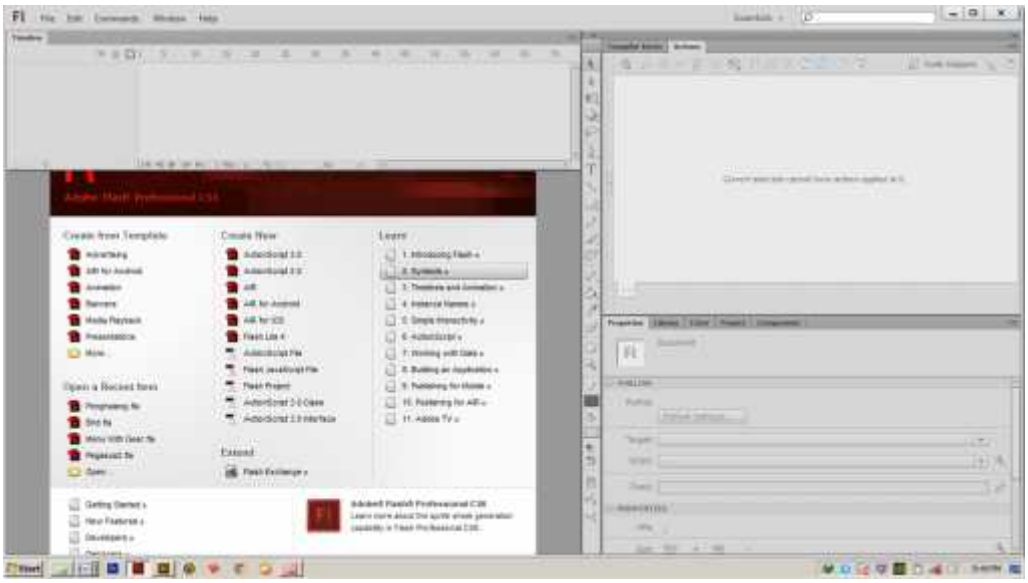
II.6. *Adobe Flash*

Adobe Flash (dahulu bernama *Macromedia Flash*) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan *Adobe System*. *Adobe flash* digunakan untuk membuat gambar vector maupun animasi gambar tersebut. Berkas yang dihasilkan dari perangkat lunak ini mempunyai file *extension* .swf dan dapat diputar di penjelajah web yang telah dipasangi *Adobe Flash Player*. *Flash* menggunakan bahasa pemrograman bernama *Action Script* yang muncul pertama kalinya pada *Flash* 5.

Adapun langkah-langkah untuk menjalankan Adobe Flash Proffesional CS 6 adalah sebagai berikut :

1. Klik tombol *Start* > *Program* > *Adobe Collection* > *Adobe Flash Proffesional CS 6.*
2. Selanjutnya akan ditampilkan dialog startup Adobe Flash Proffesional CS 6.

Tampilan *start Adobe Flash Proffesional CS 6* dapat dilihat pada gambar berikut ini :



Gambar II.3. Tampilan *start Adobe Flash Proffesional CS 6*

Sumber : Adobe Flash Proffesional CS 6

Sebelum tahun 2005, Flash dirilis oleh Adobe. Flash 1.0 diluncurkan pada tahun 1996 setelah Adobe membeli program animasi vector bernama Future Splash. Versi terakhir yang diluncurkan di pasaran dengan menggunakan nama 'Adobe' adalah Adobe Flash 8. Pada tanggal 3 Desember 2005 Adobe Systems mengakuisisi Macromedia dan seluruh produknya, sehingga nama Macromedia Flash berubah menjadi Adobe Flash, Dibawah ini Gambar II.2 tampilan Adobe Flash. Adobe Flash merupakan sebuah program yang didesain khusus oleh Adobe dan program aplikasi standar authoring tool professional yang digunakan untuk membuat animasi dan bitmap yang sangat menarik untuk keperluan pembangunan situs web yang interaktif dan dinamis. Flash didesain dengan kemampuan untuk membuat animasi 2 dimensi yang handal dan ringan sehingga flash banyak

digunakan untuk membangun dan memberikan efekanimasi pada website, CD Interaktifdan yang lainnya. Selain itu aplikasi ini juga dapat digunakan untuk membuat animasi logo, movie, game, pembuatan navigasi pada situs web, tombol animasi, banner, menu interaktif, interaktif form isian, e-card, screen saver dan pembuatan aplikasi-aplikasi web lainnya. Dalam Flash, terdapat teknik-teknik membuat animasi, fasilitas action script, filter, custom easing dan dapat memasukkan video lengkap dengan fasilitas playback FLV. Keunggulan yang dimiliki oleh Adobe Flash ini adalah ia mampu diberikan sedikit code pemograman baik yang berjalan sendiri untuk mengatur animasi yang ada didalamnya atau digunakan untuk berkomunikasi dengan program lain sepeprti HTML, PHP, dan Database dengan pendekatan XML, dapat dikolaborasikan dengan web. *(Kristo Radion, 2012:3-4).*

II.6.1 *Timeline Dan Stage*

Animasi yang dibuat di Flash diorganisasikan dengan *timeline* (representasi grafik yang terdiri dari kumpulan frame). Animasi dapat dibuat pada single frame pada suatu waktu, dengan menambahkan *key frames* pada *timeline* secara sekuensial. *Layer* dapat dipergunakan untuk mengorganisasikan elemen *frame* (layer background, layer tanaman, layer awan, layer…). Flash *interface* berisi vector drawing tool, host of palletes (colour mixing, alignment, applying transformations, setting typographics options dan lainnya).*(Kristo Radion, 2012:12)*

II.6.2 Symbol dan Tweening

Objek dapat disimpan pada library dalam bentuk khusus, yang dinamakan *symbol*, sehingga dapat dipergunakan ulang. Beberapa *instance symbol* dapat ditempatkan pada stage. *Symbol* dapat ditransformasi (ukuran, orientasi). Tween motion dapat dibuat dengan beberapa cara. Hasil tweening dapat dilihat pada timeline berupa tanda panah pada awal dan akhir keyframe yang dipilih. Tiga macam symbol di dalam Flash : *(kristo Radioan, 2012:14)*

1. *Graphic symbol*. *Simply reusable vector objects*. Dipergunakan untuk *motion tweening*.
2. *Button symbol*. Dipergunakan untuk membuat bagian interaktif.
3. *Movie clip symbol*. Animasi yang dapat ditambahkan ke dalam movie utama.

II.7 Action Script

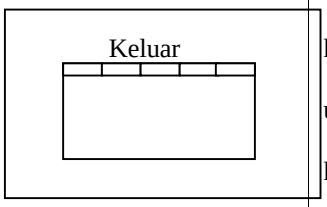
*Action Script*adalah bahasa pemrograman yang di pakai oleh software Flash untuk mengendalikan object-object ataupun movie yang terdapat dalam Flash. Sebenarnya Flash juga bisa tidak menggunakan ActionScript dalam pemakaiannya, tapi kalau menginginkan adanya interaktifitas yang lebih komplek maka ActionScript ini dibutuhkan *(Kristo Radion, 2012:6).*

II.8 *Story Board*

Storyboard adalah penggambaran jalan cerita sesuai dengan isi cerita dan berisi pengambilan sudut gambar, pengisian suara, serta efek-efek khusus.*(Fahri dan Imam, 2013:5)*

Tabel II.2. Contoh *Storyboard*

Scene	Visual	Isi	Keterangan	Audio
1	Aplikasi multimedia pembelajaran *aritmatika* komputer pada materi konversi bilangan	Intro adalah halaman awal yang pertama kali muncul saat program dijalankan. Di halaman ini terdapat judul dari plikasi yaitu media pembelajaran	Merupakan menu Intro untuk enuju kemenu utama di dalam aplikasi	mp3
2	Keluar Halama Home	Pada halaman menuutama terdapat judul aplikasi, gambar, tombol menu utama dimana user dapat menekan tombol menu utama untuk menampilkan Tombol-tombol seperti tombol kompetensi untuk	Tampilan awal menu utama yang berisi tombol Petunjuk untuk menuju ke Petunjuk, tombol Kompetensi untuk menuju Kompetensi, tombol Materi untuk menuju	mp3

3		Pada halaman kompetensi dimana user dapat melihat isi kompetensi berupa teks tentang standar kompetensi yang ingin dicapai dalam mata kuliah organisasi dan arsitektur komputer	Tampilan Menu Kompetensi ini adalah menampilkan teks yang menjelaskan tentang tujuan pembelajaran Aritmatika komputer dan	mp3

(Sumber : Fahri, Imam Riadi ; 2013:5)

II.9 UML *(Unified Modelling Language)*

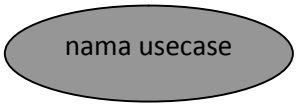
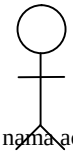
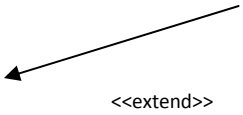
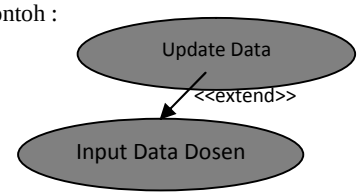
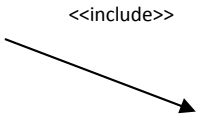
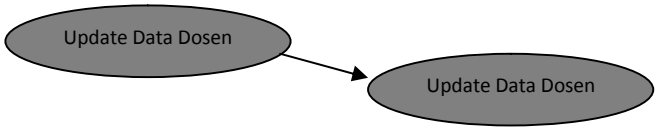
Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis 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. 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*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.*(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram Use Case yang terdapat di dalam UML. Adapun maksud dari Use Case Diagram diterangkan dibawah ini.

1. Use Case 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, 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. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*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 meng-*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. *(Yuni Sugiarti ; 2013 : 41)*

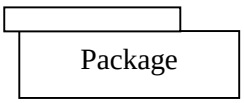
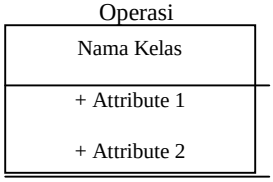
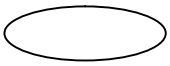
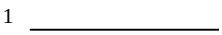
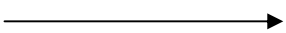
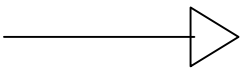
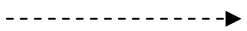
Simbol	Deskripsi
Usecase 	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase namause case.
Aktor 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan dengan menggunakan kata benda diawal frase nama aktor.
Assosiasi / Association 	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
Exetnd 	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjuk pada use case yang dituju. Contoh : 
Include 	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsi atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

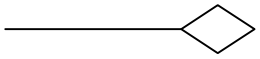
Gambar II.4. ***Use Case*** **Diagram**

Gambar II.5. *Sumber : (Yuni Sugiarti ; 2013 ; 42)*

2. *Class Diagram*

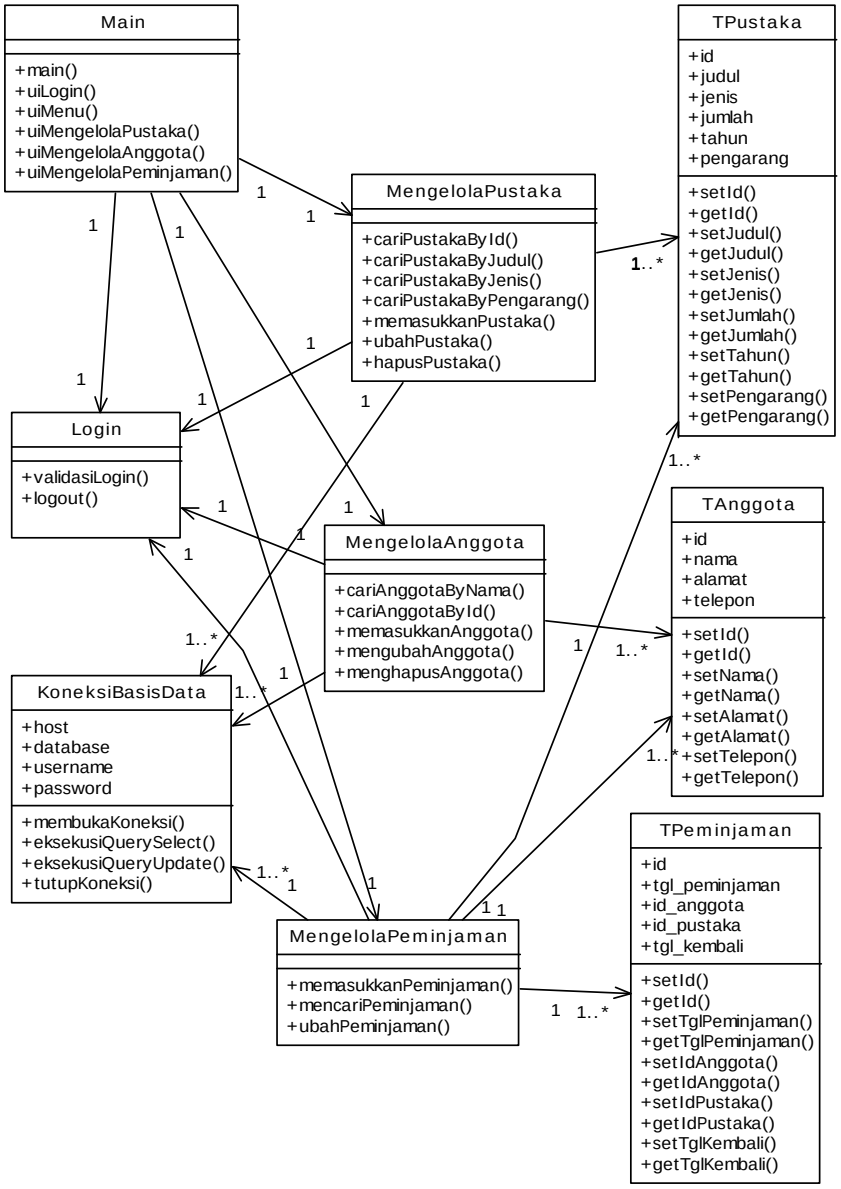
Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka/Interface  interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity.
Asosiasi Berarah/directed assosiasi 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity.
Generasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum – khusus)
Kebergantungan/Defedency 	Relasi antar kelas dengan makna kebergantungan antar kelas

Agregasi 	Relasi antar kelas dengan makna semua bagian (whole-part)

Gambar II.6. *Class* Diagram

Sumber : (Yuni Sugiarti ; 2013:59)



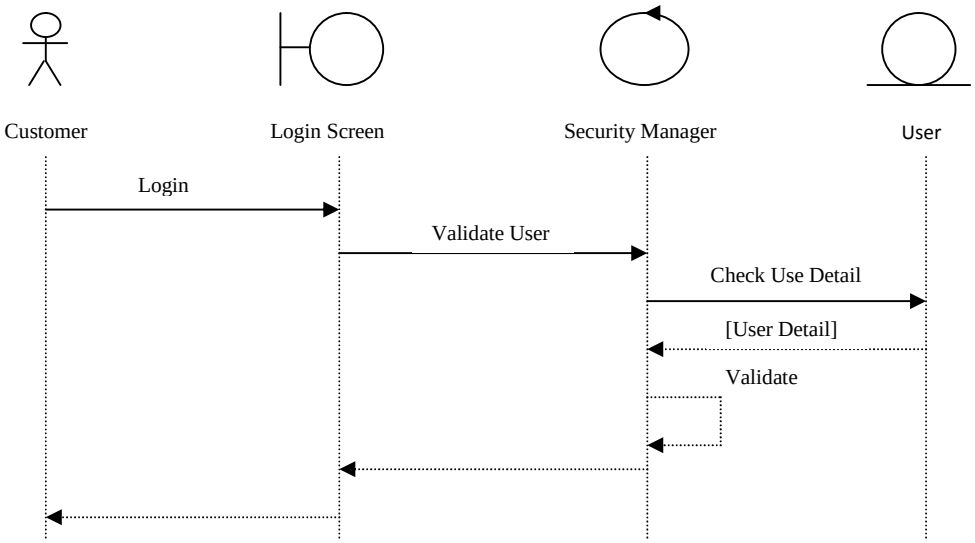
Gambar II.7. Contoh *Class* Diagram

Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.8. Contoh *Sequence* Diagram

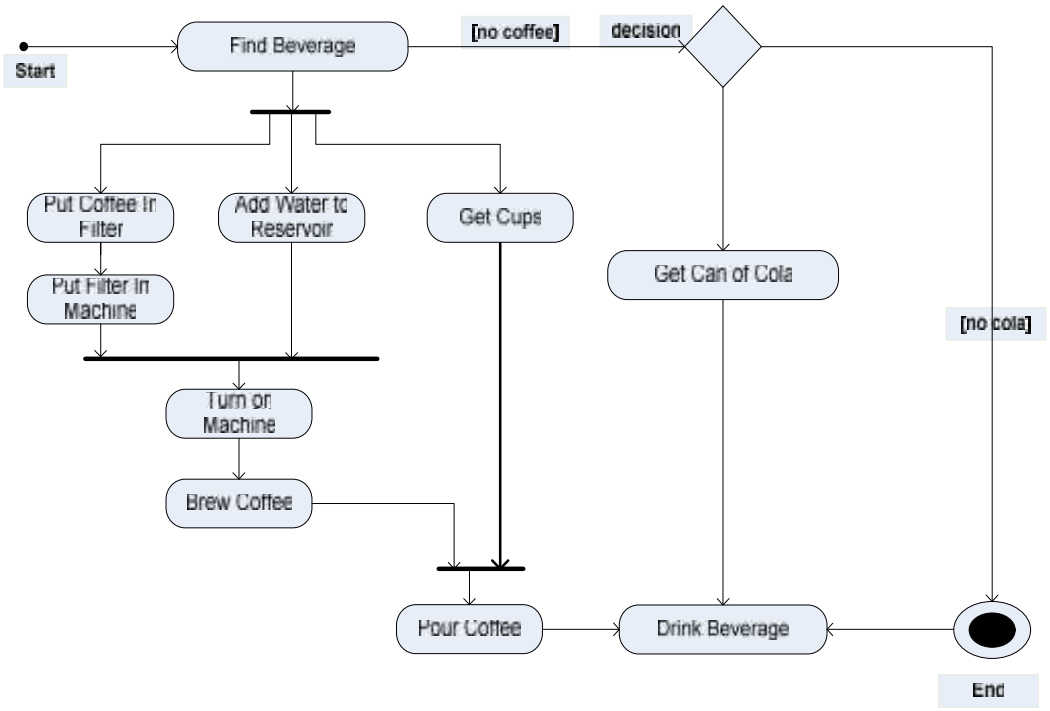
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *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 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. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas.



Gambar II.9. *Activity* Diagram

Sumber : (Yuni Sugiarti ; 2013 : 76)