BAB II

LANDASAN TEORI

Sejarah *Game*

Menurut *John von Neumann* dan *Oskar Morgenstern* tahun 1944, *game* adalah : "Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi." (*Nelly Indriani Widiastuti1, Irwan Setiawan*, 42 Volume. I Nomor. 2, Bulan Oktober 2012 - ISSN :2089-9033).

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*)

I.1.1. Generasi Perkembangan *Game*

I.1.1.1 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*)

I.1.1.2 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*)

I.1.1.3 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*)

I.1.1.4 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*)

I.1.1.5 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)*

I.1.1.6 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*)

I.1.1.7 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)*

I.1.2 *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*)

I.1.3. *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*)

I.1.3.1. *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. (*Ivan C. Sibero, 2009, 18*)

I.1.3.2. *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*)

I.1.3.3. *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*)

I.1.3.4. *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*)

I.1.3.5 *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*)

I.1.3.6 *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.2. Perancangan

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

(Syifatun Nafisah,2003,2).

II.3. Aplikasi

Aplikasi adalah suatu program yang siap untuk digunakan untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Selain itu aplikasi juga mempunyai fungsi sebagai pelayan kebutuhan beberapa aktivitas yang dilakukan oleh manusia seperti sistem untuk software jual-beli, permainan atau game online, pelayanan mayarakat dan hampir semua proses yang dilakukan oleh manusia dapat dibantu dengan menggunakan suatu aplikasi. Beberapa aplikasi jika digabungkan akan menjadi satu paket atau sering juga disebut dengan *aplication suite*, dimana aplikasi tersebut memiliki posisi antar muka yang mempunyai kesamaan sehingga dapat dengan mudah digunakan atau dipelajari penggunaan tiap aplikasi tersebut. *(Syifatun Nafisah ,2003, 2).*

II.4. Flying Pegasus

Pegasus adalah kuda putih bersayap yang tercipta dari darah Medusa setelah Perseus memenggal kepala Medusa. Satu-satunya manusia yang bisa menunggangi Pegasus adalah Bellerofon. Bellerofon mampu menjinakkan Pegasus setelah memperoleh tali kekang emas dari Dewi Athena. Bellerofon menunggangi Pegasus

dalam berbagai petualangannya, misalnya ketika membunuh Khimaira, mengalahkan kaum Solimi, dan memerangi bangsa Amazon. Suatu ketika Bellerofon mencoba menunggangi Pegasus untuk menuju Olimpus, kediaman para dewa. Para dewa menghukum Bellerofon dengan mengirim serangga untuk menyengat Pegasus sehingga Pegasus mengamuk dan menjatuhkan Bellerofon. Setelah itu Pegasus tinggal di kandang kuda di Olimpus dan berperan sebagai pembawa petir Zeus.
(http://id.wikibooks.org/wiki/Mitologi_Yunani/Makhluk_Mitologi/Pegasus)

II.5. Algoritma

Algoritma adalah perintah kepada komputer yang sistematis, rinci serta memenuhi kaidah logika tertentu. Algoritma berbeda dengan bahasa pemrograman, meskipun kadang-kadang sintaknya mirip, sebab algoritma mendeskripsikan langkah-langkah yang harus dikerjakan komputer (lebih berkaitan dengan logika/pola pikir manusia untuk memecahkan masalah tertentu). Sedang bahasa pemrograman adalah bahasa yang digunakan untuk mengkomunikasikan rincian langkah tersebut pada komputer(biasanya menggunakan bahasa pemrograman).

Algoritma bersifat universal, artinya bahasa pemrograman apapun dapat digunakan untuk mengkomunikasikan perintah-perintah manusia kepada komputer pada dasarnya memiliki algoritma yang sama. Sebab itulah seseorang yang sudah memahami dan menguasai algoritma pemecahan masalah biasanya dengan relatif mudah dapat berganti-ganti bahasa pemrograman sesuai dengan keistimewaan serta fungsi utama bahasa pemrograman yang digunakannya. Misalnya, jika akan memerintahkan komputer untuk melakukan penghitungan-penghitungan saintifik maka mungkin kita

akan menggunakan bahasa FORTRAN yang memang dirancang untuk itu. Jika kita ingin agar komputer melakukan manipulasi terhadap basis data (tambah, hapus, ubah, cari), mungkin kita akan menggunakan Borland Delphi atau Visual BASIC. Apa pun bahasa pemrograman yang kita gunakan, algoritmanya adalah sama.

Sebagai pengetahuan saja, kadang ada orang bertanya dari mana asal kata algoritma? Kata algoritma ini tidak muncul dalam kamus bahasa Inggris hingga akhir tahun 1957. Sebelum 1957,orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka Arab. Kita akan dinamakan *algorist* jika kita menggunakan angka-angka Arab (0,1,2,…..,9) di saat menghitung, para ahli sejarah matematika menemukan asal mula kata *algorism* ini berasal dari Al-Khuwarizmi (orang Eropa melafalkannya sebagai *algorism*), seorang penulis Arab yang bernama lengkap Abu Ja'far Muhammad Ibn Musa Al-Khuwarizmi. Al Khuwarizmi menulis buku yang berjudul Kitab Al Jabar Wal Muqabala yang artinya Buku Pemugaran dan Pengurangan. Kata *algorism* berubah menjadi *algorithm* karena kata algorism sering dikelirukan dengan *aritmethic*, sehingga akhirnya *–sm* berubah menjadi *–thm*. Lebih lanjut kata *algorithm* berubah menjadi algoritma setelah pemrograman komputer mulai marak di Indonesia.

(*Adi Nugroho*, 2009:22).

II.6. *Linear Congruential Generator* (LCG)

Linear Congruential Generator (LCG) merupakan jenis PRNG yang banyak digunakan dalam aplikasi komputer modern. LCG ditemukan oleh D.H Lehmer. LCM memanfaatkan model linier untuk membangkitkan bilangan acak. Bilangan acak yang

dibangkitkan oleh komputer merupakan bilangan acak semu, karena pembangkitannya menggunakan operasi-operasi aritmatika. Banyak algoritma atau metode yang dapat digunakan untuk membangkitkan bilangan acak. Di dalam buku teks statistik klasik, angka-angka acak diciptakan dengan mengambil bola yang dinomori ke luar dari suatu kotak yang berisi sejumlah bola bernomor yang diketahui jumlahnya. Jika jumlah bola yang ada didalam kotak jumlahnya sedikit, maka hal tersebut masih mungkin dilakukan. Tetapi jika jumlah bolanya dalam jumlah yang sangat besar, maka hal tersebut akan susah dan tidak efektif dilakukan. Terlepas dari hal itu, ada isu yang lain yang meragukan bahwa mengambil bola bernomor dari suatu kotak besar merupakan suatu proses acak dengan kesempatan sama untuk semua bola. Karena hal tersebut dan pertimbangan lainnya, maka dilakukan komputerisasi generator bilangan *random*/acak. Sesungguhnya, bahasa pemrograman tingkat tinggi menawarkan sedikitnya satu format dari generator bilangan *random*. Pembuatan angkaangka yang acak bukan merupakan hal yang mudah, karena komputer adalah suatu mesin *deterministik*. Karena itulah mustahil untuk membuat angka-angka acak/bilangan yang benar-benar *random* tanpa adanya perangkat keras tambahan.

True *random number* secara definisi tidak dapat terprediksi. TRNG dilakukan dengan melakukan sampling entropi sumber dari alam dan memprosesnya melalui komputer. Misalnya adalah bilangan *random* yang dihasilkan oleh Random.org dan Laverand.sgi.com. Random.org menggunakan *atmospheric noise* dari radio dan Lavarand.sgi.com menggunakan Lava Lite® lamps sebagai entropi sumber. Entropi sumber yang lain yang cukup bagus adalah radioaktivitas yang juga digunakan oleh Fourmilab di Swiss untuk membangkitkan *true random number*. *Pseudorandom*

Number Generator (PNRG) atau dalam bahasa indonesia Pembangkit bilangan acak semu adalah sebuah algoritma yang membangkitkan sebuah deret bilangan yang tidak benar-benar acak. Keluaran dari pembangkit bilangan acak semu hanya mendekati beberapa dari sifat-sifat yang dimiliki bilangan acak. Walaupun bilangan yang benar-benar acak hanya dapat dibangkitkan oleh perangkat keras pembangkit bilangan acak, bukannya oleh perangkat lunak komputer, akan tetapi bilangan acak semu banyak digunakan dalam beberapa seperti untuk simulasi dalam ilmu fisika, matematika, biologi dan sebagainya, dan juga merupakan hal yang sangat penting dalam dunia kriptografi.

Linear Congruent Method (LCM) merupakan metode pembangkitkan bilangan acak yang banyak digunakan dalam program komputer. LCM memanfaatkan model linier untuk membangkitkan bilangan acak yang didefinisikan dengan:

x(n+1) = (a xn + c) mod m ..... (01)

Dimana : xn = adalah bil. acak ke n

a dan c adalah konstanta LCM

m adalah batas maksimum bilangan aca

Ciri khas dari LCM adalah terjadi pengulangan pada periode waktu tertentu atau setelah sekian kali pembangkitan, hal ini adalah salah satu sifat dari metode ini, dan *pseudo random generator* pada umumnya.Penentuan konstanta LCM (a, c dan m) sangat menentukan baik tidaknya bilangan acak yang diperoleh dalam arti memperoleh bilangan acak yang seakan-akan tidak terjadi pengulangan. Dapat dilihat dari beberapa contoh seperti di bawah ini :

Pada contoh kasus game deal or no deal kali ini yaitu pengacakan pada nilai koper yang terdiri dari 1 sampai 26 koper. Dimana, setiap pemain tidak akan terjadi pengulangan saat memainkan game tersebut. Nilai yang akan dibangkitkan yaitu 1 sampai 26 koper, dengan ketentuan variabel a dan c nilai konstanta telah ditentukan.

Rumus :

$X(n+1) = (a.xn + c) \bmod m$

$a = 4$

$c = 7$

$x0 = 3$

$m = 27$

Penyelesaian :

$X(0) = 3$

$X(1) = (4\ (3) + 7) \bmod 27 = 19$

$X(2) = (4\ (19) + 7) \bmod 27 = 2$

$X(3) = (4\ (2) + 7) \bmod 27 = 15$

$X(4) = (4\ (15) + 7) \bmod 27 = 13$

$X(5) = (4\ (13) + 7) \bmod 27 = 5$

$X(6) = (4\ (5) + 7) \bmod 27 = 0$

$X(7) = (4\ (0) + 7) \bmod 27 = 7$

$X(8) = (4\ (7) + 7) \bmod 27 = 8$

$X(9) = (4\ (8) + 7) \bmod 27 = 12$

$X(10) = (4\ (12) + 7) \bmod 27 = 1$

$X(11) = (4\ (1) + 7) \bmod 27 = 11$

X(12) = (4 (11) + 7) mod 27 = 24

X(13) = (4 (24) + 7) mod 27 = 22

X(14) = (4 (22) + 7) mod 27 = 14

X(15) = (4 (14) + 7) mod 27 = 9

X(16) = (4 (9) + 7) mod 27 = 16

X(17) = (4 (10) + 7) mod 27 = 17

X(18) = (4 (17) + 7) mod 27 = 21

X(19) = (4 (21) + 7) mod 27 = 10

X(20) = (4 (10) + 7) mod 27 = 20

X(21) = (4 (20) + 7) mod 27 = 6

X(22) = (4 (6) + 7) mod 27 = 4

X(23) = (4 (4) + 7) mod 27 = 23

X(24) = (4 (23) + 7) mod 27 = 18

X(25) = (4 (18) + 7) mod 27 = 25

X(26) = (4 (25) + 7) mod 27 = 26

Bilangan acak yang dibangkitkan 1 sampai 26 tidak terlihat pengulangan secara periodik. (*Darma Perwira Hasibuan*, 2013, Volume : IV, Nomor: 1, ISSN : 2301-9425).

II.7. *Adobe Flash*

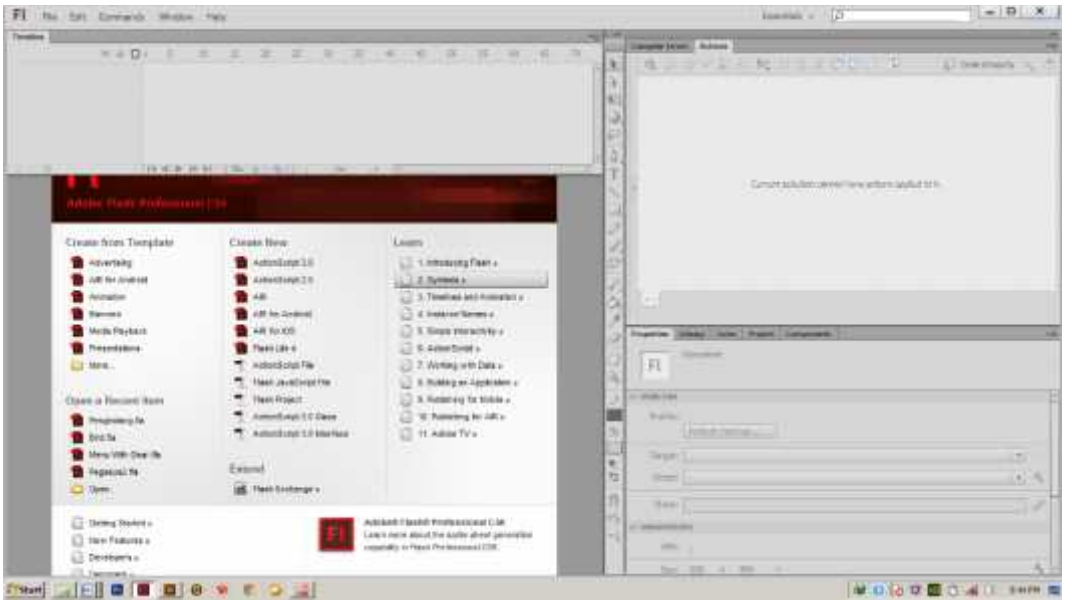
Adobe Flash (dahulu bernama *Macromedia Flash*) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan *Adobe System*. *Adobe flash* digunakan untuk membuat gambar vector maupun animasi gambar tersebut. Berkas

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 Dreamweaver.

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



Gambar II.1 Tampilan *start Dreamweaver*

Sumber : Adobe Dreamweaver CS6

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, S.ST, Ease Game Programming Usung Flash and Action Script 3.0, Penerbit Andi, Yogyakarta).*

Berikut adalah contoh action script pada flash :

```
public function Burung(side:String, speed:Number, altitude:Number) {
if (side == "left") {
this.x = -170.0; // start to the left
dx = speed; // fly left to right
this.scaleX = -1; // reverse
} else if (side == "right") {
this.x = 760.0; // start to the right
dx = -speed; // fly right to left
this.scaleX = 1; // not reverse
}
//this.y = altitude; // vertical position
this.y = randomRange(808.5, 213.7);
// choose a random bird
if (PoorBird.level == 1)
{
this.gotoAndStop(Math.ceil(Math.random()*2));
}
if (PoorBird.level == 2)
{
this.gotoAndStop(Math.ceil(Math.random()*3));
}
```

```
if (PoorBird.level == 3)

{

this.gotoAndStop(Math.ceil(Math.random()*4));

}

// set up animation

addEventListener(Event.ENTER_FRAME,moveBird);

lastTime = getTimer();

}
```

II.7.2 *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, S.ST, Ease Game Programming Usung Flash and Action Script 3.0, Penerbit Andi, Yogyakarta)

II.7.3 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.

Motion tweening?Gerakan gambar ditentukan terlebih dahulu dengan membuat motion path. Shape tweening? Dikenal dengan nama morphing. Perubahan bentuk suatu objek menjadi bentuk baru. Tiga macam symbol di dalam Flash :

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.

(Kristo Radion, S.ST, Ease Game Programming Usung Flash and Action Script 3.0, Penerbit Andi, Yogyakarta)

II.8 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, S.ST, Easy Game Programming Usung Flash and Action Script 3.0, Penerbit Andi, Yogyakarta).*

II.9 UML *(Unified Modelling Language)*

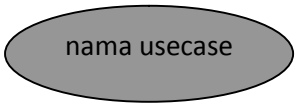
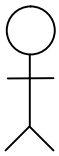
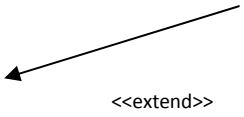
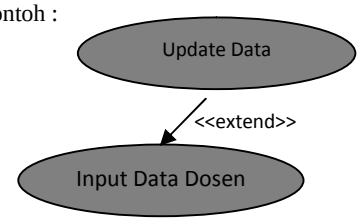
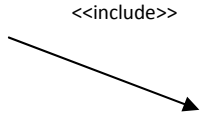
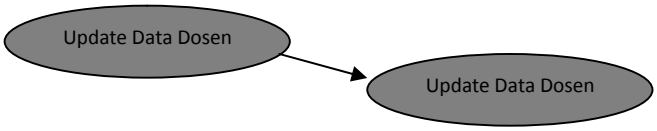
Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis 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

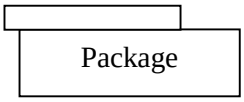
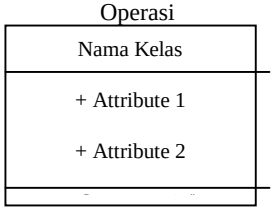
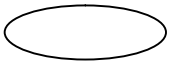
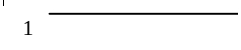
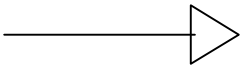
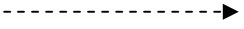
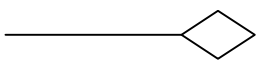
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login 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  nama actor	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 : 

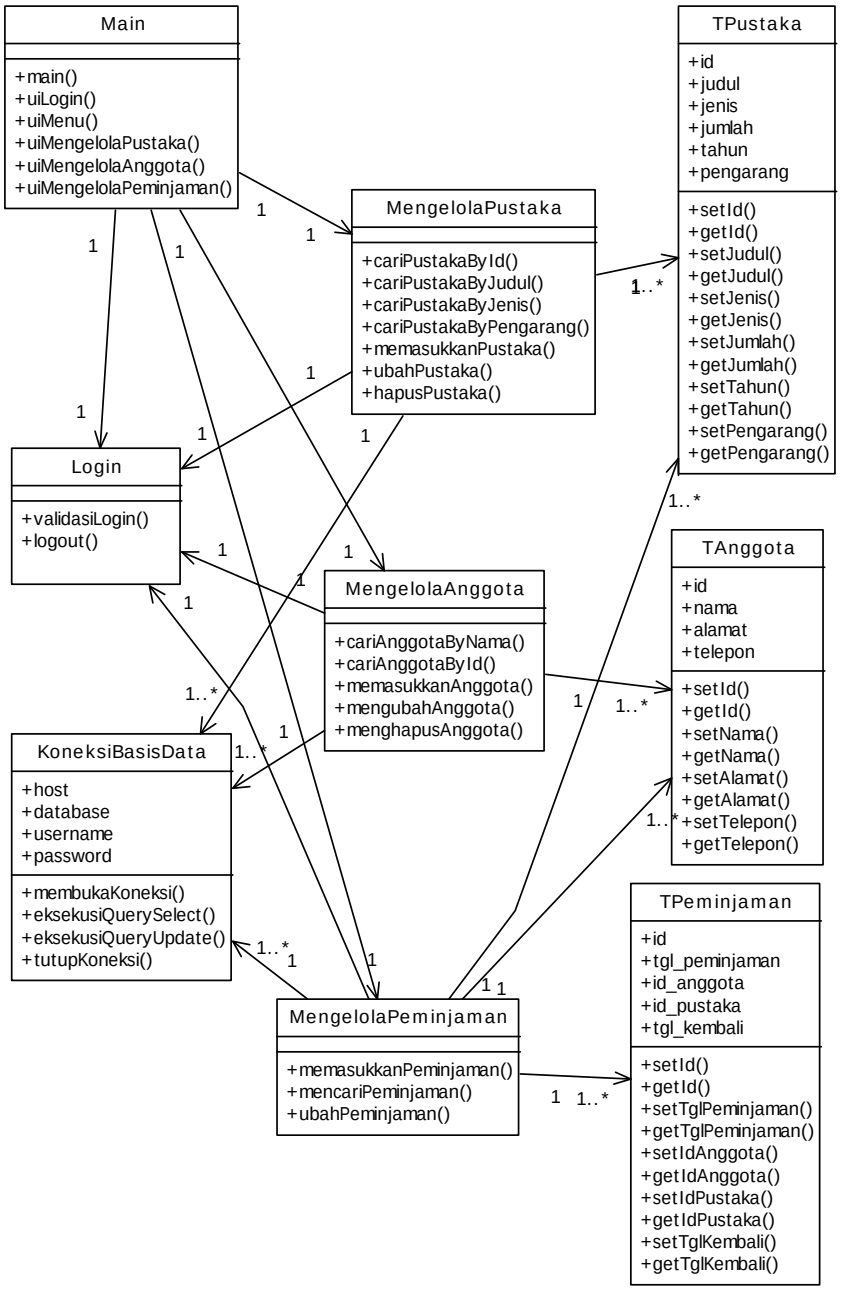
Gambar II.2. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

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 system
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.3. *Class* Diagram
Sumber : (Yuni Sugiarti *; 2013 : 59)*

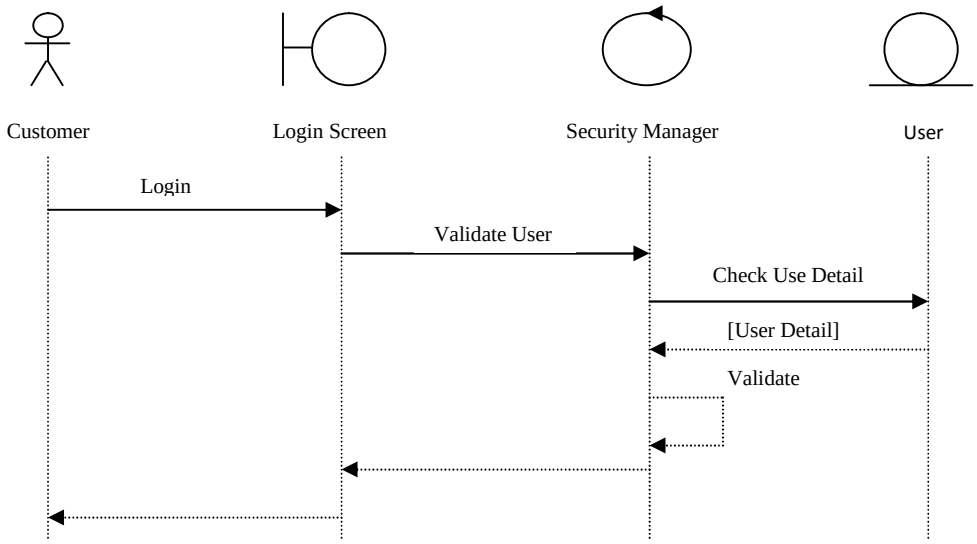


Gambar II.4. Contoh *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

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.5. Contoh *Sequence* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

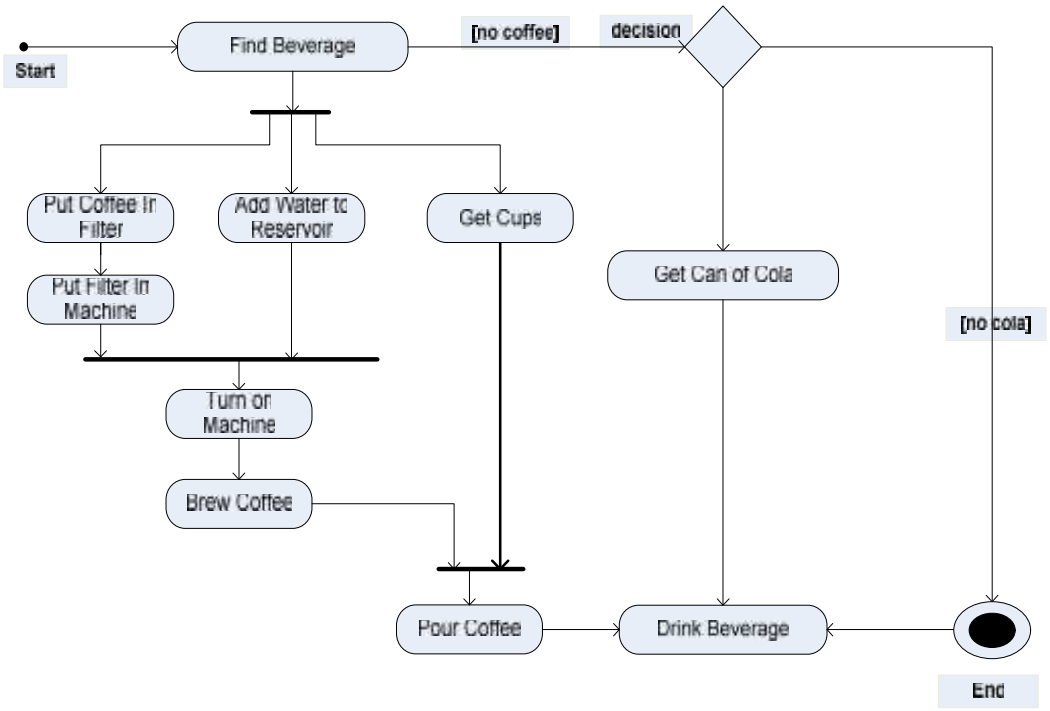
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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. *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.



Gambar II.6. *Activity* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)