BAB II
LANDASAN TEORI

2.1 Permainan (*Game)*

Permainan berasal dari arti kata bahasa Inggris yaitu *game*[4]. Permainan adalah suatu kegiatan atau sebuah aktivitas yang didalamnya terdapat peraturan, cara bermain (*gameplay)*, budaya yang dipahami oleh pemain. Permainan bertujuan untuk menghibur, biasanya permainan banyak disukai oleh anak-anak hingga orang dewasa. *Game* juga bisa diartikan sebagai area keputusan dan aksi pemainnya. Permainan dibedakan menjadi dua jenis permainan yaitu , permainan tradisional dan permainan *modern*.

2.1.1 Permainan tradisional

Permainan tradisional adalah merupakan bagian dari tradisi lisan, pada hakikatnya sama dengan apa yang disebut permainan rakyat[5]. Permainan rakyat ini sangat bermanfaat baik untuk orang dewasa maupun anak-anak. Biasanya berdasarkan gerak tubuh seperti lari dan lompat, atau berdasarkan kegiatan sosial sederhana seperti kejar-kejaran, sembunyi-sembunyian, dan berkelahi-kelahian atau berdasarkan matematika dasar atau kecekatan tangan seperti menghitung dan melemparkan batu ke suatu lubang tertentu atau berdasarkan keadaan untung-untungan seperti main dadu.

Berdasarkan sifat permainan, maka permainan rakyat dapat dibagi menjadi dua golongan, yaitu permainan untuk bermain (*Play*) dan permainan untuk bertanding (*Game*)[6]. Perbedaan dari keduanya ini, bahwa yang pertama lebih bersifat mengisi waktu senggang atau rekreasi sedangkan yang kedua tidak mempunyai sifat itu, karena harus memiliki sebuah keahlian atau ilmu untuk memenangkan sesuatu permainan tersebut. Berikut jenis-jenis contoh permainan :



Gambar 2.1 Permainan balap karung[6]



Gambar 2.2 Permainan Tek Tak[6]



Gambar 2.3 Permainan Lepar Kaleng [6]

2.1.2 Permainan Modern

Permainan modern adalah permainan yang berasal dari industri atau pada umumnya menggunakan teknologi dalam pembuatan serta permainannya[7]. Permainan jenis ini menghadirkan bentuk permainan seperti video *game*, baik itu dalam *console* maupun *computer*. Ada banyak jenis *console game* yang memberikan kemajuan teknologi dalam segi tampilan gambar maupun cara memainkan yang beragam sehingga menarik perhatian untuk dimainkan.

Berdasarkan dari sifat permainannya sama dengan tradisional yaitu untuk bermain dan bertanding bahkan taruhan . permainan modern dari segi negative sangat berdampak besar bagi kalangan masyarakat diantaranya adalah anak-anak menjadi malas karena kecanggihan permainan modern, para orang dewasa biasanya menggunakan permainan sebagai permainan taruhan dikarenakan ilmu atau pemain menguasai bidang tersebut[8]. Berikut adalah contoh permainan modern:



Gambar 2.4 Permainan Modern[8]

2.2 Klasifikasi Permainan

Pentingnya sebuah klasifikasi untuk mengenal beberapa jenis-jenis bentuk permainan atau jenis bidang pada permainan itu sendiri yaitu :

2.2.1 Klasifikasi berdasarkan *Platform* Permainan

Berdasarkan jenis *platform* atau alat yang digunakan pada permainan yaitu :

1. *Arcade Games*

   Permainan yang berbentuk sebuah mesin atau box. Permainan yang terkenal di Indonesia adalah *ding-dong.* Permaianan ini pada umumnya didesign khusus untuk jenis *video games*.

2. *Pc Games*

   Pc *Game*s adalah permainan *video game* yang dimainkan menggunakan *personal computers.* Dimana pemain harus menginstall sebuah permainan pada *computer*.

3. *Console Games*

   Console game adalah permainan yang menggunakan *video game* dan menggunaka*n console* tertentu contohnya seperti playstation ,XBOX 360, Nintendo dan Wii

4. *Handheld games*

   Handheld games adalah permainan *video game* seperti *console* akan tetapi video *game* tersebut dapat dibawa kemana-mana *(portable),* contohnya Sony PSP, Nintendo DS.

5. *Mobile Games*

   Mobile games adalah permainan yang digunakan khusus pada *mobile phone* atau *PDA* yang dapat dibawa tanpa membawa perangkat yang besar.

2.2.2 Klasifikasi berdasarkan pembatas usia

Berikut ini adalah klasifikasi berdasarkan pembatas usia dalam penggunaan permainan yaitu :

1. Early Childhood (3+)

   Memiliki isi dalam permainan yang mungkin cocok untuk usia 3 atau lebih tua. Tidak mengandung kekerasan atau tidak cocok pada usia tersebut.
2. Everyone (+6)

   Memiliki konten yang mungkin cocok untuk usia 6 tahun lebih tua. Judul-judul dalam kategori ini mungkin berisi kartun fantasia atau bahasa yang ringan
3. Everyone (10+)

   Memiliki konten yang mungkin cocok untuk usia 10 tahun lebih tua. Kategori ini mungkin berisi lebih banyak kartun, fantasi , bahasa ringan atau minimal tema sugestif.
4. Teen 13 (13+)

   Memiliki konten yang mungkin cocok untuk usia 13 tahun lebih tua. Judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, darah, simulasi perjudian dan jarang menggunakan bahasa yang berat.
5. Mature (17 +)

   Memiliki konten yang mungkin cocok untuk orang usia 17 atau lebih tua. Judul dalam kategori ini mungkin berisi intens kekerasan, konten seksual bahasa yang berat.
6. Adults Only (18+)

   Memiliki konten yang hanya boleh dimainkan oleh orang-orang yang berusia 18 tahun atau lebih tua. Judul dalam kategori ini mungkin termasuk adegan intens kekerasan atau konten seksual.

7. Rating Pending (RP)

   Penantian pemberian penilaian akhir oleh ESRB dari terciptanya sebuah *game*. (Simbol tersebut hanya akan muncul dalam iklan sebelum permainan dibebaskan).

2.2.3 Klasifikasi berdasarkan *genre* permainan

Berikut ini adalah klasifikasi berdasarkan *genre* atau jenis permainan yaitu:

1. *Shooter*

   Jenis *game* yang di mainkan menggunakan sebuah senjata, biasanya pistol, senapan, atau senjata jarak jauh lainnya. Biasanya tujuan dari permainan ini adalah menembak lawan dan memenangkan misi tanpa harus gugur dimedan perang. Shooter dibagi menjadi 3 kategori yaitu :

   a. First Person Shooter (FPS)

      Jenis *game* yang menggunakan sudut pandang orang pertama yang biasanya dimainkan tanpa bisa melihat karakter yang dimainkan. Hal ini bermaksud untuk memberikan perasaan seolah dalam permainan tersebut dan memungkingkan untuk pemain lebih focus dalam bidikan , hal tesebut kita hanya bisa melihat tangannya saja tanpa melihat tubuh karakter yang dimainkan.

   b. Third Person Shooter (TPS)

      Permainan ini sama dengan FPS yaitu jenis *game* yang menggunakan sudut pandang orang ketiga yaitu tubuh karakter bisa dilihat secara detail.

   c. Massively Multiplayer Online First Person Shooter (MMOFPS)

      Permainan yang dimainkan secara online yang menggabungkan pemain FPS pada sebuah dunia *Virtual*, dimana pemain dapat berinteraksi satu dengan lainnya.

2. *Strategy*

   Jenis *game* yang mengharuskan pemainnya menggunakan taktik dan strategi untuk jeli dalam melihat setiap peluang, kelemahan musuh dan bijaksana dalam menggunakan sumber daya yang ada untuk mengatur suatu unit atau pasukan untuk menyerang markas musuh dalam rangka memenangkan permainan. Biasanya di dalam *game* strategy, dituntut untuk mencari uang, emas, poin atau semua yang berfungsi untuk untuk membiayai pasukan kita. Jenis strategy ini dibagi menjadi 2 katagori yaitu:

   a. Real Time Strategy (RTS)

      Jenis *game* ini, dapat mengendalikan pasukan secara langsung dari sumber daya sehingga menghancurkan musuh. Semua pertempuran ini dapat disaksikan secara langsung.

   b. Turn Base Strategy (TBS)

      Jenis *game* yang dijalankan secara bergiliran, saat mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan harus menunggu, begitu pula sebaliknya, layaknnya catur.

3. *Racing*

   Game yang dimainkkan dengan mengendalikan sebuah kendaraan untuk memenangkan sebuah balapan atau garis finish dari suatu race, dalam *game* ini biasanya pemain dapat memilih & membeli kendaraan, mendandani, mengupgrade mesin dll.

4. *Arcade*

   Jenis *game* yang tidak terfokus pada cerita, melainkan hanya dimainkan untuk bersenang-senang atau untuk kejar-mengejar point/highscore.

5. Fighting

   Jenis *game* yang intinya harus menjatuhkan, mengalahakan bahkan membunuh lawan tandingnya, dengan pukulan, tendangan, *combo*, maupun dengan jurus *special*.

6. *Sport*

   Jenis Permainan bertema olahraga, sistem aturan dan cara permainan akan berbeda-beda tergantung jenis olahraga yang menjadi tema *game* tersebut.

7. *Vehicle Simulation*

   Jenis permainan simulasi pengoperasian beberapa kendaraan, kendaraan bisa berupa pesawat terbang, pesawat tempur, kereta, kendaraan perang, maupun kendaraan konstruksi.

8. *Adventure*

   Jenis *game* yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang telah di desain sedemikian rupa, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, mencari pesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat *game* ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain.

9. *Simulation*

   Jenis *game* yang memberikan pengalaman atau interaksi sama dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut.

10. Tycoon

    Game yang menjadikan sebagai seorang businessman yang akan mengembangkan suatu *property* untuk dikembangkan sehingga laku dipasaran.

11. *Musical*

    Jenis *game* yang menuntut pemainnya untuk menekan tombol sesuai dengan tombol yang ada di layar dengan diiringi dengan musik.

12. *Platform*

    Jenis *game* yang mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang

musuh dan menghindar terhadap serangan. Jenis *game* ini sedikit serupa dengan action *game*, tetapi aksinya tidak secepat *action game*.

13. *Action*

    Jenis *game* dengan fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan.

14. *Role Playing game (RPG)*

    Jenis *game* yang seringkali berupa multi-player *game* di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang spesifik. Para pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafis yang khas untuk setiap karakter pemain ditambah dengan storyline yang mendebarkan akan sangat menarik dan memberikan pengalaman yang berbeda di dalam bermain.

15. *Puzzle*

    Jenis *Game* yang berintikan teknik pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video *game* petualangan maupun *game* edukasi.

16. *Historic*

    Jenis *game* ini dibuat berdasarkan cerita-cerita sejarah , permainan ini dibuat sebagaimana untuk menciptakan suasana-suasana pada jaman sejarah, contohnya merasakan bagaimana peristiwa-peristiwa penting dan sebagai edukasi untuk pembelajaran lebih mengenal sejarah.

17. *Real Life*

    Jenis *game* yang dibuat seperti kehidupan sehari-hari atau aktifitas. Permainan ini sering dimainkan oleh semua kalangan , dimana permainan ini sebagai pandangan jika memainkan karakter dari permainan tersebut. Contohnya pemain merawat tumbuhan dan

tumbuhan itu terus berkembang sesuai dengan perawatannya, jika tidak dirawat maka tumbuhan tersebut akan mati atau layu.

18. *Sandbox game*

    Sandbox Adalah tipe *game* yang umumnya ditujukan untuk menjelajahi suatu kota atau tempat dimana karakter yang kita pakai bebas berinteraksi dengan objek disekitarnya. Mungkin *game* ini bisa di bilang mirip RPG tapi Sandbox sangat berbeda dengan RPG *game*.

2.3 Algoritma

Dalam dunia komputasi, istilah algoritma menjadi dasar pemikiran sebuah formulasi. Algoritma dapat didefinisikan sebagai teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas tetapi tersusun secara logis dan sistematis[9]. Kalimat-kalimat ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Algoritma harus berhenti setelah mengerjakan serangkaian tugas atau langkahnya terbatas dan setiap langkah harus didefinisikan dengan tepat sehingga tidak memiliki arti ganda. Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan (logika Boolean dan perbandingan) sampai tugasnya selesai. *Desain* dan analisis algoritma adalah suatu cabang khusus dalam ilmu *computer* yang mempelajari karakteristik dan performansi dari suatu algoritma dalam menyelesaikan masalah.

Algoritma pun bersifat bebas, artinya tidak bergantung pada sistem komputer atau bahasa pemrograman yang digunakan. Algoritma yang berbeda dapat diterapkan pada suatu masalah dengan kriteria yang sama[9]. Dalam penyelesaian masalah, ukuran banyak komputasi dari suatu algoritma dinyatakan dalam kompleksitas. Jika sebuah permasalahan dalam diselesaikan dalam waktu yang singkat dikatakan kompleksitas algoritma rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

2.3.1 Pencarian Jalur Terpendek

Secara umum penyelesaian masalah pencarian jalur terpendek dapat dilakukan menggunakan dengan dua buah metode, yaitu metode algoritma konvensional dan metode heuristic[10]. Metode algoritma konvensional diterapkan dengan cara perhitungan matematis seperti biasa, sedangkan metode heuristik diterapkan dengan perhitungan kecerdasan buatan, dengan menentukan basis pengetahuan dan perhitungannya. Berikut ini adalah beberapa metode untuk pencarian jalur terpendek.

a. Metode konvensional

   Metode konvensional berupa algoritma yang menggunakan perhitungan matematis biasa. Ada beberapa metode konvensional yang biasa digunakan untuk melakukan pencarian jalur terpendek, diantaranya algoritma Djikstraa, algoritma Floyd-Warshall, dan algoritma Bellman-Ford.

b. Metode heuristik

   Sub bidang dari kecerdasan buatan yang digunakan untuk melakukan pencarian dan penentuan jalur terpendek. Ada beberapa algoritma pada metode heuristik yang biasa digunakan dalam pencarian jalur terpendek. Namun dalam penelitian dibatasi hanya membahas algoritma *Iteratite Deepening A**.

2.3.2 *Iteratite Deepening A** (ITERATIVE DEEPENING A*)

Iterative deepening A* merupakan pengembangan dari A* dan *Depth first search*. Pengembangan dari A* dalam *Iterative deepening A*[11]* sebagai fungsi *heuristic* yang sering disebut dengan f(n) yang merupakan penentuan urutan titik yang akan dikunjungi terlebih dahulu, dan *depth first search* yang berfungsi sebagai pencarian *node* dengan jalur pohon dimana iterasi kedalamannya dibatasi yang digantikan dengan batas nilai f. Dalam pseudocode berikut, h adalah fungsi heuristik yang memperkirakan jalan dari node ke tujuan. Berikut contoh dari psecode algoritmanya :

```
Procedure IDA*(n);
Bound := h(n);
While not solved do

Bound := DFS (n,bound);

Function DFS (n,bound);
If f(n) > bound
Then return f(n);
If h(n) = 0
Then return solved;
Return lowest value of DFS (n,bound)
For all successor n of n
```

2.4 ActionScript 3.0

ActionScript adalah bahasa pemrograman untuk Adobe Flash Player dan Adobe AIR ™ run-time. Itu memungkinkan interaktivitas , penanganan data, dan masih banyak lagi di Flash, Flex, dan AIR konten dan aplikasi. ActionScript mengeksekusi dalam ActionScript Virtual Machine (AVM) , yang merupakan bagian dari Flash Player dan AIR[12] .

Kode ActionScript biasanya diubah ke dalam format bytecode oleh kompilator (bytecode adalah jenis bahasa pemrograman yang ditulis dan dipahami oleh komputer) Contoh kompiler termasuk yang dibangun untuk Adobe Flash Professional dan salah satu yang dibangun untuk Adobe Flash Builder ™ dan tersedia dalam SDK Adobe Flex ™. Bytecode ini tertanam dalam file SWF, Flash Player dan AIR untuk dieksekusi.

ActionScript 3.0 menawarkan model pemrograman yang kuat yang akrab bagi pengembang yang memiliki pengetahuan dasar tentang pemrograman berorientasi obek. Beberapa fitur kunci dari ActionScript 3.0 yang dapat meningkatkan lebih dari versi ActionScript sebelumnya meliputi :

a. Sebuah ActionScript Virtual Machine baru , yang disebut AVM2 , yang menggunakan set instruksi bytecode baru dan menyediakan peningkatan kinerja yang signifikan.
b. Sebuah kode compiler dasar lebih modern yang melakukan optimasi lebih dalam daripada versi sebelumnya dari compiler.
c. dikembangkan dan ditingkatkan dengan application programming interface (API), dengan kontrol low-level objek dan true object-oriented model.
d. Sebuah API XML didasarkan pada ECMAScript untuk XML (E4X) spesifikasi (ECMA - 357 Edisi 2). E4X adalah bahasa ekstensi untuk ECMAScript yang menambahkan XML sebagai tipe data asli dari bahasa.
e. Sebuah model acara didasarkan pada Document Object Model (DOM) Level 3 Acara Spesifikasi.

ActionScript 3.0 melampaui kemampuan scripting dari versi sebelumnya dari ActionScript. Hal ini dirancang untuk memfasilitasi pembuatan aplikasi yang sangat kompleks dengan set data yang besar dan, basis kode reusable object-oriented. ActionScript 3.0 tidak diperlukan untuk konten yang berjalan di Adobe Flash Player. Namun, membuka pintu untuk kinerja perbaikan yang hanya tersedia dengan AVM2 (ActionScript 3.0 virtual mesin). ActionScript 3.0 kode dapat mengeksekusi hingga sepuluh kali lebih cepat daripada kode bawaan ActionScript. Versi sebelumnya dari ActionScript Virtual Machine, AVM1, mengeksekusi ActionScript 1.0 dan ActionScript 2.0 kode. Flash Player 9 dan 10 dukungan AVM1 untuk kompatibilitas.

2.4.1 *Case Sensitive*

ActionScript adalah bahasa pemrograman yang bersifat case sensitive, artinya adalah penulisan huruf kapital (huruf besar) dan huruf kecil dianggap berbeda. Kesalahan dalam penulisan huruf kapital atau huruf kecil akan berakibat error pada program.

2.4.2 Fungsi *Trace*

Trace adalah sebuah fungsi untuk menampilkan pesan pada panel output. Contoh struktur penulisan dari fungsi trace sebagai berikut:

```
trace("pesan di sini");
```

2.4.3 Komentar

Komentar adalah baris kode yang tidak dieksekusi. Fungsi dari komentar adalah untuk memberikan keterangan tertentu pada baris kode. Komentar akan sangat berguna dalam pembuatan aplikasi besar yang terdiri dari banyak baris kode.

Untuk menuliskan komentar ada 2 cara, yaitu dengan memberikan tanda (//) dan tanda (/**/) sebelum menuliskan pesan. Tanda (//) digunakan untuk menuliskan komentar yang terdiri dari 1 baris, sedangkan tanda (/**/) digunakan untuk menuliskan komentar yang lebih dari 1 baris. Baris kode yang diberikan tanda komentar akan berwarna abu-abu (default).

2.4.4 Tipe Data dan Variabel

Data didefinisikan ke dalam variabel. Data juga dapat diload dari file eksternal (file text, xml, dll.), webservice (via http, dll.), dan database. Beberapa tipe data yang akan sering digunakan adalah, sebagai berikut:

1. String, yaitu kumpulan karakter huruf, angka maupun punctuation karakter (!, @, #, $, %, dan lain-lain).
2. Number, yaitu data numerik, defaultnya bertipe float, dapat bertipe integer atau unsigned integer (bilangan integer yang hanya memiliki nilai positif.
3. Boolean, yaitu tipe data yang terdiri atas 2 kondisi yaitu true dan false. True juga dapat diisi dengan angkan > 0, sedangkan false dapat juga diisi dengan angka 0.
4. Object, yaitu kumpulan dari properties, atribut yang mendeskripsikan objek.

5. MovieClip yaitu symbol yang dapat memainkan animasi.
6. Void, mengindikasikan function yang tidak memberikan return value.
7. Null, yaitu segala sesuatu yang bersifat null.
8. Undefined, yaitu tipe data yang belum didefinisikan. Secara default, setiap tipe data yang belum diberikan value akan dianggap undefined.

Seperti yang telah dibahas sebelumnya, data didefinisikan ke dalam variabel. Variabel berfungsi untuk menyimpan informasi. Beberapa aturan dalam penulisan variabel adalah, sebagai berikut:

1. Variabel dapat berupa sebuah atau sekumpulan karakter yang terdiri dari huruf dan angka, hindari penggunaan punctuation karakter (!, @, #, $, %, dan lain-lain).
2. Variabel harus diawali dengan huruf.
3. Variabel bersifat CASE-SENSITIVE, yang berarti variabel lowercase atau huruf kecil dengan variabel uppercase atau huruf besar dianggap berbeda. Jadi variabel A dengan variabel a adalah 2 variabel yang berbeda.
4. Yang paling penting, hindari penamaan variabel sama dengan keyword Actionscript (contoh: trace, var, String, dll.). Keyword Actionscript adalah syntax yang berwarna biru jika dituliskan pada flash.

Untuk melihat value dari suatu variabel, dapat menggunakan perintah trace(nama_variabel). Contoh deklarasi variabel namaMahasiswa dengan tipe data String:

```
var myString:String;
var  myString:String  =  new  String("hello
world");
var myString:String = "hello world";
```

2.4.5 Tipe Data *String*

String adalah kumpulan karakter yang tersusun atas huruf, numerik maupun punctuation. Value variabel bertipe string harus diapit oleh single quotation marks (‘) maupun double quotation marks (“). Berikut beberapa contoh value untuk tipe data string:

```
var namaJurusan:String="TEKNIK INFORMATIKA UNIKOM";
var namaUniversitas:String='Universitas Komputer Indonesia';
var namaKota:String="Bandung";
var namaDepan:String="Felicia";
var namaBelakang:String="Kalalo";
trace(namaJurusan);
trace(namaUniversitas);
trace(namaKota);
trace(namaDepan+" "+namaBelakang);
```

2.4.6 *Escape Sequence* pada *String*

Misalkan terdapat sebuah variabel string bertuliskan “Felicia Kalalo”, pertanyaan yang muncul adalah bagaimana menuliskan tanda kutip (“) di dalam string? Jawabannya dengan menggunakan escape sequence, yang diawali dengan karakter \ (backslash), sehingga scriptnya menjadi:

```
var namaSaya:String="Nama saya \"Felicia Kalalo\"";
trace(namaSaya);
```

Tabel 2.1 Escape sequence

Escape Sequence	Character
\b	Backspace character (ASCII 8)
\f	Form-feed character (ASCII 12)
\n	Line- feed character (ASCII 10)
\r	Carriage return character (ASCII 13)
\t	Tab character (ASCII 9)
\"	Double quotation mark
\’	Single quotation mark
\\	Backslash
\000-\377	A byte specified in octal
\x00-\Xff	A byte specified in hexadecimal
\u0000-\Uffff	A 16-bit Unicode character specified in hexadesimal

2.4.7 Tipe Data *Number*

Secara default, tipe data Number bersifat Float (bilangan real), dengan value minimun 5e- 324 dan maksimum 1.79E+308. Tipe data Number dapat dimanipulasi dengan menggunakan operator aritmatik tambah (+), kurang (-), kali (*), bagi (/), mod (%), increment (++) dan decrement (--). Contoh script:

```
var a:Number=100;
var b:Number=12.5;
trace(a+b); // menghasilkan output 112.5
trace(a-b); // menghasilkan output 87.5
trace(a*b); // menghasilkan output 1250
trace(a/b); // menghasilkan output 8
trace(a%b); // menghasilkan output 0
a++; // sama dengan a=a+1
b--; // sama dengan b=b-1
trace(a); // menghasilkan output 101
trace(b); // menghasilkan output 11.5
```

2.4.8 Fungsi *String*

Fungsi string berguna untuk melakukan konversi dari tipe numerik yaitu float, integer dan unsigned integer, menjadi tipe data string. Perhatikan output script berikut:

```
var h:Number = 2.45;
var i:uint = 10;
var j:int = 2011;
trace(String(h)+" "+String(i)+" "+String(j));
```

2.4.9 Separator dan Operator

Separator yang terdapat di dalam ActionScript 3.0 adalah sebagai berikut:

Tabel 2.2 Separator

Separator	Nama	Fungsi
.	Titik	Pemisah syntax dan pemisah antara objek dengan properties.
[ ]	Kurung siku	Menentukan index array.
( )	Kurung	Menentukan parameter suatu fungsi, menentukan ekspresi untuk kondisi dan pengulangan.
;	Titik koma	Menandakan akhir dari baris kode.
,	Koma	Digunakan sebagai pemisah nilai pada array.
"	Kutip dua	Digunakan untuk mengisi nilai string.
'	Kutip satu	Digunakan untuk mengisi nilai string.
{ }	Kurung Kurawal	Digunakan untuk mengisi kode pada pengulangan, kondisi, fungsi, class dan package.

Separator harus digunakan sesuai dengan struktur dari syntax pada ActionScript. Kesalahan pada separator akan mengakibatkan error pada saat program dijalankan. Flash akan memberikan hint (petunjuk) separator yang harus digunakan untuk beberapa syntax.

Operator pada Flash terdiri dari arithmetic operator dan logical operator. Arithmetic operator berfungsi untuk melakukan operasi-operasi aritmatika, sedangkan logical operator berfungsi untuk operasi boolean. Operator-operator yang terdapat di dalam ActionScript 3.0 adalah sebagai berikut:

Tabel 2.3 Operator

Oparator	Tipe	Fungsi
+	Arithmetic	Penambahan pada numerik dan penyambungan string
-	Arithmetic	Pengurangan pada data numerik
*	Arithmetic	Perkalian pada data numerik
/	Arithmetic	Pembagian pada data numerik
%	Arithmetic	Sisa pembagian integer pada data numerik
!	Logical	Operasi NOT pada data boolean
~	Logical	Negasi pada data boolean
^	Logical	XOR pada data boolean
=	Arithmetic	Hasil dari operasi
<	Logical	Indentifikasi apakah nilai pada bagian kiri lebih kecil dari nilai pada bagian kanan untuk data numerik
>	Logical	Indentifikasi apakah nilai pada bagian kiri lebih besar dari nilai pada bagian kanan untuk data numerik
<=	Logical	Indentifikasi apakah nilai pada bagian kiri lebih kecil atau sama dengan nilai pada bagian kanan untuk data numerik
>=	Logical	Indentifikasi apakah nilai pada bagian kiri lebih besar atau sama dengan nilai pada bagian kanan untuk data numerik
==	Logical	Indentifikasi apakah nilai pada bagian kiri sama dengan nilai pada bagian kanan untuk data numerik dan data string
!=	Logical	Indentifikasi apakah nilai pada bagian kiri tidak sama dengan nilai pada bagian kanan untuk data numerik dan data string
&&	Logical	Operasi AND pada data boolean.
||	Logical	Operasi OR pada data boolean
and	Logical	Operasi AND pada data boolean
or	Logical	Operasi OR pada data boolean
<<	Arithmetic	Operasi left shift pada data numerik biner
>>	Arithmetic	Operasi arithmetic right shift pada data numerik biner
>>>	Logical	Operasi logical right shift pada data numerik biner
+=	Arithmetic	Operasi pertambahan nilai pada bagian kiri dengan nilai pada bagian kanan untuk data numerik dan data string
-=	Arithmetic	Operasi pengurangan nilai pada bagian kiri dengan nilai pada bagian kanan untuk data numerik
*=	Arithmetic	Operasi perkalian nilai pada bagian kiri dengan nilai pada bagian kanan untuk data numerik
/=	Arithmetic	Operasi pembagian nilai pada bagian kiri dengan nilai pada bagian kanan untuk data numerik
%=	Arithmetic	Operasi pencarian sisa hasil pembagian nilai pada bagian kiri dengan nilai pada bagian kanan untuk data numerik
^=	Logical	Operasi XOR untuk nilai pada bagian kiri dengan nilai pada bagian kanan untuk data numerik
<<=	Arithmetic	Operasi left shift nilai pada bagian kiri sebanyak nilai pada bagian kanan untuk data numerik biner
>>=	Arithmetic	Operasi arithmetic right shift nilai pada bagian kiri sebanyak nilai pada bagian kanan untuk data numerik biner
>>>=	Logical	Operasi logical right shift nilai pada bagian kiri sebanyak nilai pada bagian kanan untuk data numerik biner

Contoh penggunaan arithmetic dan logical operator dalam ActionScript seperti pada script berikut ini.

```
var a:int=8;
var b:String="hello";
var c:Boolean=false;
var d:Number=6;
trace(a - d);
trace(a ^ 2);
trace(b + " world");
trace(!c);
trace(~d);
trace(a >> 1);
```

2.4.10 *If - else*

Struktur syntax kondisi if - else adalah sebagai berikut:

```
if(kondisi){
Operasi jika kondisi terpenuhi;
}else if(kondisi){
Operasi jika kondisi pertama tidak
terpenuhi;
}else{
Operasi jika semua kondisi tidak
terpenuhi;
}
```

```
var sambutan;
var bahasa = "indonesian";
if (bahasa == "english") {
sambutan = "Hello";
} else if (bahasa == "japanese") {
sambutan = "Konnichiwa";
} else if (bahasa == "french") {
sambutan = "Bonjour";
} else if (bahasa == "german") {
sambutan = "Guten tag";
} else {
sambutan = "Apa kabar";
}
trace(sambutan);
```

Syntax if - else dapat juga berbentuk if saja, tanpa menggunakan syntax else. Selain itu, syntax if - else juga dapat dibuat bersarang, seperti pada contoh script berikut.

```
var a:Number = 11;
var b:String = "Indonesia";
if((a%2) > 0){
if(b == "Indonesia"){
trace("Bilangan Ganjil");
}else{
trace("Odd Number");
}
```

2.4.11 *Switch*

Teknik kondisi menggunakan syntax switch sangat efektif digunakan jika pada suatu kasus terdapat banyak kondisi, script yang dituliskan akan menjadi panjang jika dituliskan dalam bentuk kondisi if - else. Struktur statement switch adalah sebagai berikut:

```
switch(kondisi){
case nilai1:operasi jika nilai1
terpenuhi;
brek;
case nilai2:operasi jika nilai 2
terpenuhi;
break;
default: operasi jika tidak ada
kondisi terpenuhi;
break;
}
```

Perhatikan struktur syntax switch di atas, masing-masing kondisi dibatasi dengan syntax break. Fungsi dari syntax break adalah untuk menghentikan proses pencocokkan kondisi setelah salah astu kondisi terpenuhi. Script berikut adalah contoh penggunaan syntax switch :

```
var greeting:String;
var language="indonesian";
switch (language) {
case "english" :
greeting="Hello";
break;
case "japanese" :
greeting="Konnichiwa";
break;
case "french" :
greeting="Bonjour";
break;
default :
greeting="Apa kabar";
break;
}
trace(greeting);
```

2.4.12 *Do - while*

Pengulangan do - while akan melakukan identifikasi pada kondisi setelah suatu operasi dilakukan. Struktur dari syntax do - while adalah sebagai berikut.

```
while (kondisi){
operasi;
}
Script berikut adalah contoh penggunaan
syntax while.
var total = 2;
var counter = 0;
while (counter < 0){
total *= 2;
counter ++;
}
trace(total);
```

Pengulangan while akan melakukan identifikasi pada kondisi sebelum suatu operasi dilakukan. Perhatikan output yang dihasilkan pada contoh script menggunakan syntax do - while dengan script yang menggunakan syntax while. Syntax do - while akan menghasilkan output 4 yang diperoleh dari operasi perkalian 2 sebelum dilakukan identifikasi pada kondisi. Berbeda dengan output pada script yang menggunakan syntax while, output yang dihasilkan adalah 2 karena kondisi pengulangan tidak terpenuhi.

```
do {
operasi;
}
while (kondisi);
Script berikut adalah contoh
penggunaan syntax do - while.
var total = 2;
var counter = 0;
do {
total *= 2;
counter ++;
}
while (counter < 0);
trace(total);
```

Script berikut adalah contoh penggunaan syntax while yang digabung dengan syntax if.

```
var address = "feliciakalalo@gmail.com";
var isValidAddress = false;
var i = 0;
while (i < address.length) {
if (address.charAt(i) == "@") {
isValidAddress = true;
break;
}
i++;
}
trace(isValidAddress);
```

2.4.13 *For*

Pengulangan for akan melakukan suatu pengulangan secara beruntun selama kondisi masih terpenuhi. Index pengulangan for dapat menaik atau menurun.

```
var address = "feliciakalalo@gmail.com";
var isValidAddress = false;
for (var i = 0; i < address.length; i++) {
if (address.charAt(i) == "@") {
isValidAddress = true;
break;
}
}
trace(isValidAddress);
```

2.4.14 *For - each*

Pengulangan for - each sering digunakan untuk menampilkan semua properti pada suatu data bertipe objek. Struktur dari pengulangan for - each adalah sebagai berikut:

```
for each (index in object){
operasi;
}
```

2.4.15 *Built-in function*

Function yang tidak tergabung di dalam class disebut sebagai top-level functions atau ada juga yang menyebutnya dengan predefined atau built-in functions. Built-in function adalah function yang sudah disediakan oleh aplikasi Flash.

2.4.16 *Named dan user-defined functions*

Named dan user-defined functions adalah fungsi buatan sendiri dan diberi nama sendiri oleh programmer, script yang terdapat di dalam blok function dapat dituliskan sendiri tergantung kebutuhan. Fungsi ini dipanggil dengan menggunakan namanya diikuti dengan parameter dari fungsi tersebut. Fungsi ini relatif terhadap timeline. Misalkan sebuah function terletak pada frame 1, maka script tersebut tidak dapat diakses pada frame 2, demikian pula sebaliknya.

```
function functionName(parameters) {
function block;
}
//script pemanggilan fungsi
functionName(parameter);
```

Script berikut adalah contoh named and user-defined functions.

```
function cetakPesan() {
trace("Halo, pesan ini dipanggil
menggunakan function");
     }
cetakPesan();
```

Script berikut ini adalah contoh penggunaan parameter di dalam fungsi.

```
var a:String = "Pesan ini dimasukkan
sebagai parameter";
function cetakParameter(b:String) {
trace(b);
}
cetakParameter(a);
```