

F

DASAR PEMROGRAMAN

# FLASH GAME



Wandah W



disertai CD

## **Pengantar Flash**

Adobe flash atau sebelumnya Macromedia Flash merupakan software multifungsi. Terlepas dari fungsi awalnya, yaitu mempermudah pembuatan animasi web, ternyata flash berkembang pesat hingga dapat kita manfaatkan sebagai software multi media yang luar biasa. Bahkan Flash dengan action script-nya dapat dimanfaatkan menjadi program pembuat game yang mudah dan efektif.

Sebelum lebih lanjut mengenai pembuatan game flash, kita harus memahami terlebih dahulu bahwa membuat game flash itu menyenangkan dan menguntungkan. Bagi sebagian orang memainkan game sangat menyenangkan, konsep inilah yang harus kita pegang yaitu kita berusaha untuk menyenangkan orang lain melalui sebuah game. Semakin orang lain menikmati permainan game buatan kita, disitulah letak keberhasilan kita sebagai seorang pembuat game.

Pemrograman komputer pada umumnya akan sulit dipahami oleh seseorang yang awam terhadap sebuah bahasa pemrograman. Disisi lain membuat game merupakan salah satu penerapan dari ilmu pemrograman komputer. Namun kita tidak perlu berkecil hati, karena flash dan action scriptnya sangat mudah untuk dipelajari dan melalui buku ini saya berusaha membagi sedikit pengetahuan saya tentang membuat game dengan flash.

### **Metode Memahami Buku Ini**

Buku ini menjelaskan sebisa mungkin tentang proses pembuatan game secara berurutan. Ikutilah tutorial satu demi satu mulai dari bab awal sampai bab akhir dan jangan anda lompati sebuah tutorial kecuali anda sudah memahaminya. Apabila anda menemui kesulitan dalam menuliskan script nantinya, anda dapat membuka file pada CD tutorial. Karena action yang dipakai dalam buku ini adalah action dasar, maka action-action pada buku ini dapat bekerja pada Action Script 1.0, 2.0, dan 3.0.

### **Area Kerja dalam Flash**

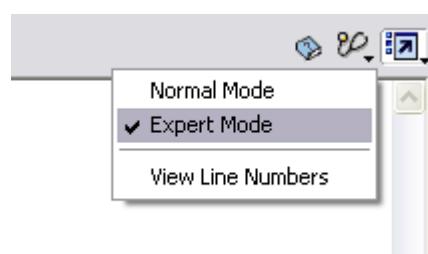
Sebelum memulai proses pembuatan game, akan lebih baik jika kita mengenal siapa dan bagaimana flash itu. Area kerja Flash (mx, mx 2004, dan flash 8) pada dasarnya terdiri atas beberapa komponen yaitu Menu, Toolbox, Timeline, Stage dan Panel.

- Menu** berisi kontrol untuk berbagai fungsi seperti membuat, membuka, menyimpan file, dan sebagainya sesuai dengan menu yang ditampilkan.
- Stage** adalah area persegi empat yang merupakan tempat dimana kita membuat obyek animasi atau aplikasi yang akan di jalankan.
- Toolbox** berisi menu untuk membuat atau menggambar bentuk . Toolbox terbagi menjadi empat bagian yaitu drawing tool, view, color, dan option
- Timeline** adalah tempat kita dapat membuat dan mengontrol obyek dan animasi.
- Panel** berisi kontrol fungsi yang dipakai dalam Flash yaitu untuk mengganti dan memodifikasi berbagai properti obyek animasi dengan cepat

## **Dasar Penulisan Action Script dalam Flash**

Macromedia Flash memiliki fasilitas berupa Action Script, yang dapat kita manfaatkan sebagai media dalam membuat suatu program aplikasi, multimedia, CD interaktif bahkan game. Macromedia Flash memiliki ratusan script dengan fungsi yang berbeda-beda akan tetapi yang sering digunakan hanya beberapa script saja, sehingga akan memudahkan kita apabila kita masih awam terhadap Flash maupun Action Script-nya.

Dalam Flash terdapat dua mode panel penulisan script yaitu mode normal dan mode expert. Pada dasarnya mode expert lebih mudah untuk dipelajari daripada mode normal, sehingga seluruh action pada tutorial di dalam buku ini ditulis dalam mode expert.



Pilihan mode expert dalam flash

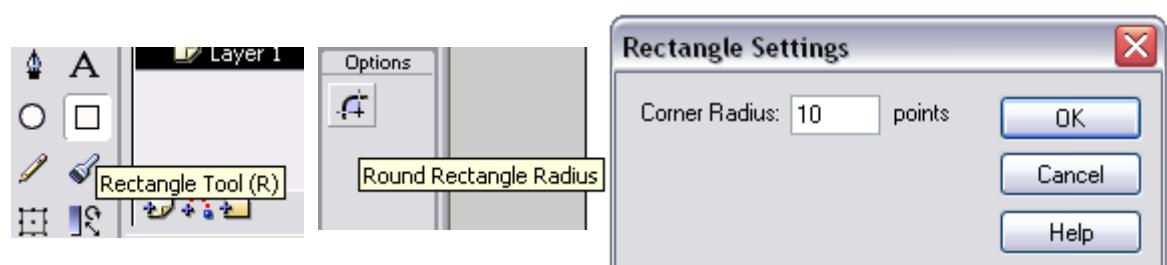
## **Pola Penulisan Action Script dalam Flash**

Action script dalam flash dapat dituliskan pada 3 tempat yaitu pada button, pada movieclip, dan pada frame dengan aturan yang berbeda.

### **Pola Penulisan Action Script pada Button**

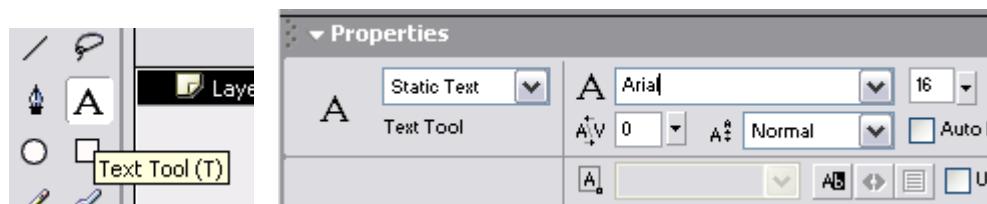
Cara penulisan action pada tombol adalah sebagai berikut :

1. Buatlah sebuah kotak dengan menggunakan **rectangle tool**. Anda juga bisa membuat rounded rectangle dengan memilih option **rounded rectangle radius**.



### Seting dalam membuat kotak

2. Buatlah static text “tombol” tepat diatas kotak yang sebelumnya telah kita buat, caranya klik text tool, kemudian buka properties dan atur jenis huruf, ukuran dan warna. Buka panel **properties** (apabila belum terbuka pilih menu window>properties). Pastikan option static text pada text type.

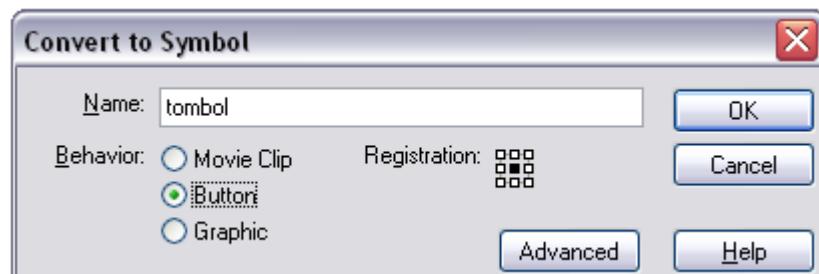


Text tool dan text properties



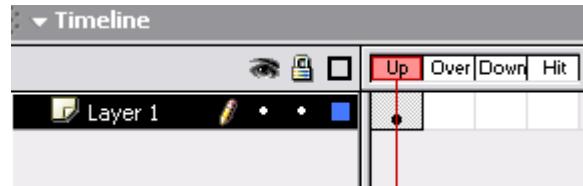
Bentuk tombol

3. Seleksi kotak dan text tersebut, kemudian konvert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **button** pada option behaviour dan ketikan **tombol** pada name.



Convert to symbol

4. Double klik tombol tersebut untuk memasuki mode edit. Perhatikan dalam symbol bertipe button terdapat 4 frame yaitu **up**, **over**, **down** dan **hit**. **Up** menunjukkan kondisi normal suatu tombol. **Over** menunjukkan kondisi tombol saat mouse berada tepat di atasnya. **Down** menunjukkan kondisi tombol saat ditekan dan **Hit** merupakan area penekanan (sensor) tombol tersebut.



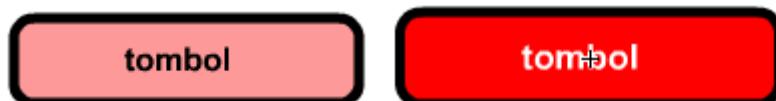
berada tepat di atasnya. **Down** menunjukkan kondisi tombol saat ditekan dan **Hit** merupakan area penekanan (sensor) tombol tersebut.

Frame pada symbol button

5. Klik frame **over** kemudian masukan **keyframe** dengan menekan tombol **F6** (pilih menu insert>keyframe). Ubah warna dan ukuran tombol pada frame **over** tersebut. Untuk frame **Down** dan **Hit** biasanya bisa kita abaikan.



Penambahan keyfame pada frame over



Perbedaan warna dan ukuran antara frame up dan frame over

6. Tekan tombol **Ctrl-E** untuk kembali ke **Stage** utama. Klik tombol dan dalam kondisi tombol terpilih buka panel action( tekan F9 atau klik menu windows>action apabila panel action belum terbuka dan pastikan mode penulisan adalah mode expert jika menggunakan flash MX atau Flash 8). Kemudian ketikan script sebagai berikut :

```
on (release){
    // menampilkan pesan pada output panel saat tombol ditekan
    trace("tombol ditekan");
}
```

action pada button

7. Jalankan movie dengan menekan tombol **Ctrl+Enter**. Tekan tombol tersebut, maka akan mucul tampilan “tombol ditekan” pada output panel. Tekan tombol **Ctrl+W** untuk kembali ke stage

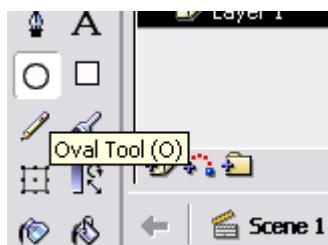
Penjelasan pola penulisan action script pada button adalah sebagai berikut :

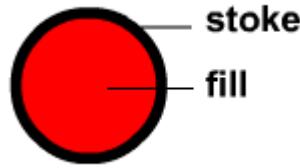
1. Action pada button harus dimulai dengan script **on()**
2. Di dalam tanda kurung () pada scrip **on** berisi **mouse event** seperti release, press, dragover, rollout dan sebagainya. Mouse event menyatakan sebuah kejadian yang akan di eksekusi oleh action tersebut saat mouse melakukan kegiatan tertentu. Sebagai contoh : mouse event **release** berarti perintah dalam blok {} akan dijalankan ketika mouse menekan tombol.
3. Tanda { merupakan awal suatu blok script. Yang dimaksud dengan blok scrip adalah suatu kumpulan perintah yang akan dijalankan sepanjang movie event.
4. Tanda // merupakan tanda yang menyatakan komentar. Baris yang memiliki tanda tersebut tidak dianggap sebagai suatu perintah. Meskipun demikian tanda // sangat membantu memberi informasi pada penulisan action scrip kita.
5. Baris **trace(“tombol ditekan”);** merupakan baris perintah. Perintah dalam button harus ditulis didalam sebuah blok mouse event. Setiap akhir penulisan sebuah perintah selalu diakhiri dengan tanda ;
6. Tanda } merupakan penutup suatu blok script.

### Pola Penulisan Action Script pada Movieclip

Cara penulisan action pada movieclip adalah sebagai berikut :

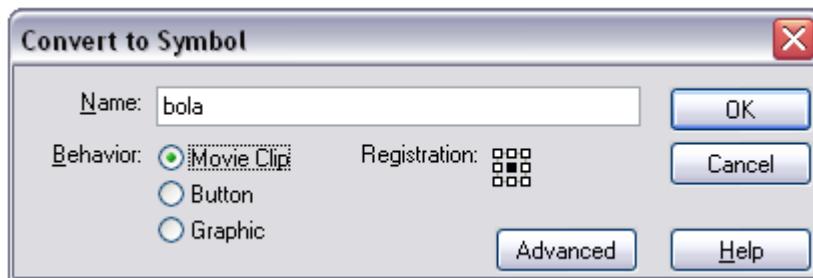
1. Buatlah sebuah obyek lingkaran dengan menggunakan oval tool. Perhatikan bahwa sebuah obyek dalam flash memiliki 2 bagian yaitu stroke (garis tepi) dan fill (isi).





Oval tool dan obyek dalam flash

- Seleksi obyek lingkaran tersebut kemudian convert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name.



Convert to symbol

- Klik movieclip **bola**, dan dalam kondisi terseleksi buka panel action ( tekan f9 atau klik menu windows>action apabila panel action belum terbuka dan pastikan mode penulisan adalah mode expert jika menggunakan flash MX atau Flash 8). Kemudian ketikan script sebagai berikut :

```
onClipEvent (enterFrame) {
    // menggerakan obyek ke kanan
    _x += 10;
}
```

action pada movieclip

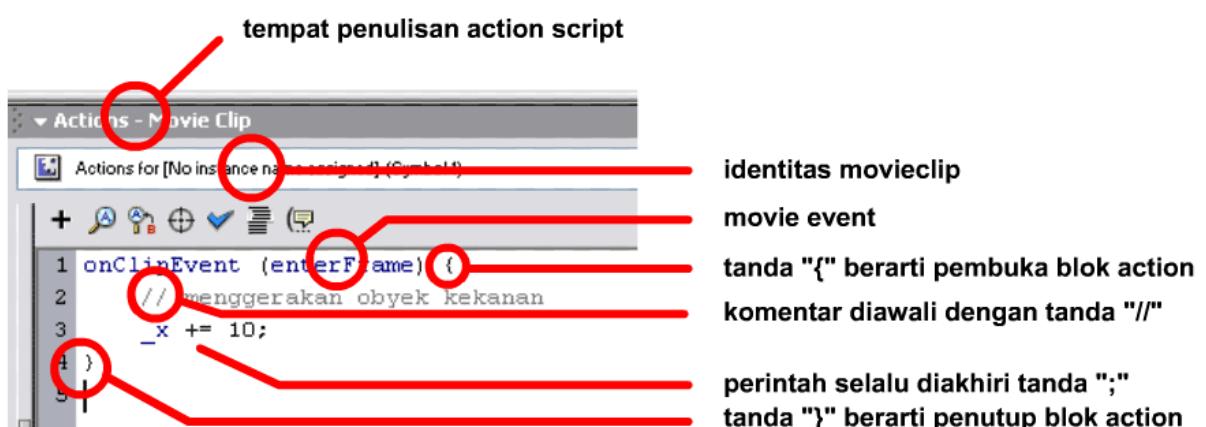
- Jalankan movie dengan menekan tombol Ctrl+Enter. Maka bola akan bergerak ke kanan sampai hilang dari layar. Tekan tombol Ctrl+W untuk kembali ke stage

Penjelasan pola penulisan action script pada movieclip adalah sebagai berikut :

- Action pada movieclip harus dimulai dengan script **on()** atau **onClipEvent()**
- Di dalam tanda kurung () pada scrip **on** maupun **onClipEvent** berisi **movie event** seperti load, enterframe, mouse down, release dan sebagainya. Movie event

menyatakan sebuah kejadian yang akan di eksekusi oleh action tersebut. Sebagai contoh : movie event **enterFrame** berarti perintah dalam blok { } akan dijalankan sepanjang frame tempat movieclip tersebut aktif dan movie event **load** berarti perintah dalam blok { } akan dijalankan satu kali saja yaitu ketika movieclip di load (ditampilkan) oleh flash player.

3. Tanda { merupakan awal suatu blok script. Yang dimaksud dengan blok script adalah suatu kumpulan perintah yang akan dijalankan sepanjang movie event.
4. Tanda // merupakan tanda yang menyatakan komentar. Baris yang memiliki tanda tersebut tidak dianggap sebagai suatu perintah. Meskipun demikian tanda // sangat membantu memberi informasi pada penulisan action script kita.
5. Baris `_x+=10;` merupakan baris perintah. Perintah dalam movieclip harus ditulis didalam sebuah blok movie event. Setiap akhir penulisan sebuah perintah selalu diakhiri dengan tanda ;
6. Tanda } merupakan penutup suatu blok script.



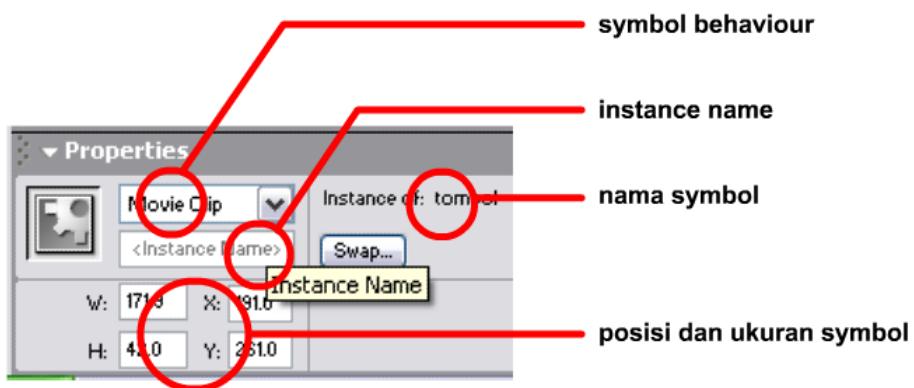
Pola penulisan script pada movieclip

Keuntungan penulisan script pada movieclip :

1. Dibandingkan dengan menulis action diframe, penulisan script di dalam movieclip lebih mudah.
2. Untuk mendeteksi kesalahan satu obyek, pada penulisan ini lebih cepat dibandingkan dengan penulisan action pada frame (dengan catatan apabila suatu saat kita mengolah script dalam jumlah ratusan sampai ribuan baris).

## Pola Penulisan Action Script pada Frame

Kedua cara penulisan script diatas yaitu penulisan script di dalam button dan di dalam movieclip, dapat ditulis dalam bentuk lain yang mana penulisan scriptnya diletakan didalam sebuah frame. Hal utama yang harus diingat dalam penulisan action pada frame adalah **instance name**. Instance name merupakan salah satu identitas yang terpenting dalam flash yang membedakan antara obyek satu dengan obyek yang lain.



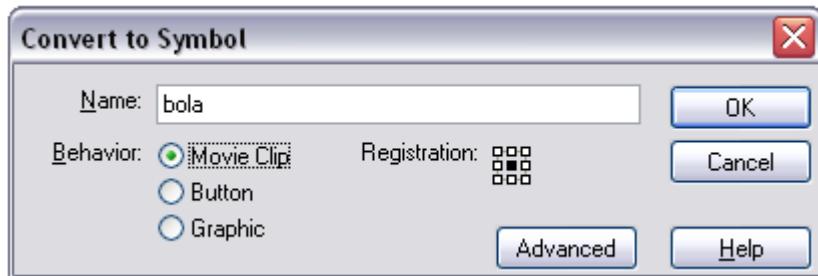
Properties sebuah symbol

Aturan pemberian nama instance name antara lain adalah :

1. Nama intansi boleh sama dengan nama symbol.
2. Tidak boleh menggunakan spasi, titik, dan tanda baca yang lain.
3. Tidak boleh menggunakan nama yang sama dengan suatu perintah dalam flash misalnya : **obyek**, **level**, **break** dan sebagainya.
4. Tidak boleh menggunakan angka saja untuk nama akan tetapi sebuah nama instansi boleh diikuti dengan angka. Contoh yang salah :**12**, **13**, dan sebagainya, contoh yang benar : **bola1**, **bola\_2** dan sebagainya.
5. Tidak boleh ada nama instansi yang sama.
6. Untuk mempermudah proses penulisan script, dapat ditambahkan “**\_mc**” untuk symbol bertipe movieclip dan “**\_btn**” untuk symbol bertipe button. Contoh : **“bola1\_mc”**, **“tombol1\_btn”** dan sebagainya.
7. Dua atau lebih symbol yang sama boleh dipakai, tetapi harus memiliki nama instansi yang berbeda.

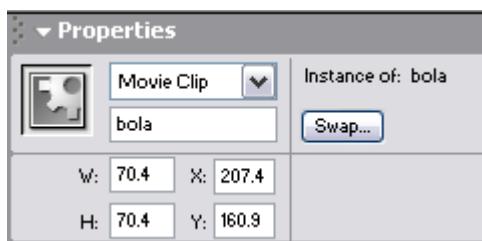
Cara penulisan action pada movieclip adalah sebagai berikut :

- Buatlah sebuah obyek lingkaran dengan menggunakan oval tool. Seleksi obyek lingkaran tersebut kemudian convert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name.



Convert to symbol

- Klik symbol bola tersebut, kemudian buka panel instance dan ketikan nama “bola” pada instance name.



Properties movieclip bola

- Buatlah sebuah layer baru dan ubah namanya menjadi layer **action**. Penambahan layer action akan mempermudah kita dalam penulisan action pada frame, sehingga tidak akan mengganggu obyek yang lain.



layer action

- Klik frame 1 layer **action**, dan dalam kondisi frame 1 terseleksi buka panel action ( tekan f9 atau klik menu windows>action apabila panel action belum terbuka dan pastikan mode penulisan adalah mode expert jika menggunakan flash MX atau Flash 8).

```

bola.onEnterFrame = function() {
    //menggerakan bola kekanan
    this._x += 10;
};

```

5. Jalankan Movie dengan menekan tombol Ctrl+Enter, maka kita akan melihat hasil yang sama dengan dengan cara penulisan action pada movieclip.

Penjelasan pola penulisan action script pada frame adalah sebagai berikut :

1. Sebuah penulisan script di frame memiliki 3 tipe penulisan yaitu **penulisan di dalam** suatu blok **movie event** atau **mouse event**, contoh **onEnterframe = function(){}**  dan penulisan **tanpa movie event** atau **mouse event**, tipe penulisan yang ke tiga adalah kombinasi dari keduanya, contohnya akan kita dapat pada program di bab selanjutnya
2. Pada dasarnya penulisan action pada frame sama dengan penulisan pada movieclip maupun pada button. Hanya saja pada frame terdapat script **function()** dan penggunaan **nama instansi** .
3. Perhatikan perbedaan antara penulisan pada frame dan penulisan pada movieclip berikut

baris	pada frame	pada movieclip	perbedaan
1.	bola.onEnterFrame = function{	onClipEvent(enterFrame) {	Perubahan penulisan dari movieclip ke frame adalah penggunaan <b>nama instansi</b> didepan movieevent dan penggunaan script <b>function()</b>
2.	//menggerakan bola kekanan	//menggerakan bola kekanan	baris komentar sama di seluruh penulisan
3.	this._x+=10;	_x+=10;	penggunaan kata-kata <b>this</b>

			harus disertakan pada penulisan di frame
4.	};	}	pada penulisan di frame sebuah akhir dari movie event function diakhiri dengan tanda ;

perbedaan antara penulisan di frame dan di movieclip

4. Apabila ingin menuliskan action button pada frame, maka yang harus di ubah adalah **movie event** diubah menjadi **mouse event**. contoh : **tombol.onRelease = function(){}**

Keuntungan penulisan script pada frame :

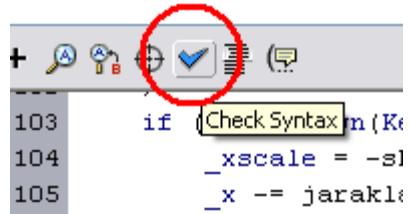
1. Dibandingkan dengan menulis action di movieclip, penulisan script di dalam movieclip cenderung lebih rumit, akan tetapi bagi programmer yang familiar dengan bahasa pemrograman seperti C++, Java, Delphi atau V basic, penulisan pada frame akan menjadi lebih mudah.
2. Lebih mudah dalam menuliskan function tertentu atau perintah berulang yang sering dipakai.
3. Karena seluruh script biasanya dituliskan pada frame, maka mengecek action seluruh obyek yang ada akan lebih mudah, tanpa harus ke masing-masing obyek.

## Kesalahan Action Script

Seseorang yang baru belajar action script maupun seseorang yang telah mahir menggunakan actionscript tidak pernah terlepas dari sebuah kesalahan. Dalam sebuah pemrograman terdapat 2 jenis kesalahan, yaitu kesalahan penulisan dan kesalahan logika.

## Kesalahan Penulisan Action Script

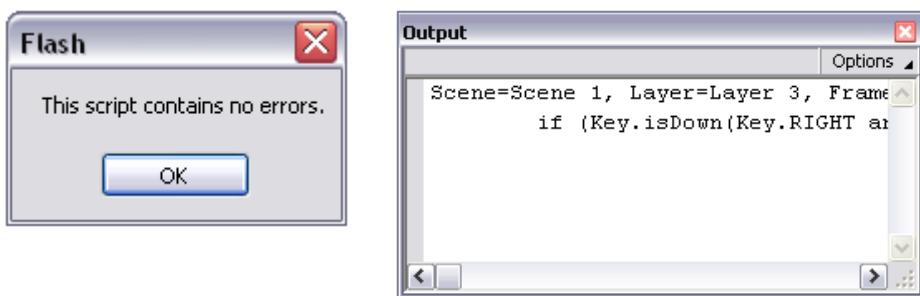
Kesalahan yang paling sering dilakukan oleh orang yang baru belajar script adalah kesalahan penulisan script. Untuk mengetahui kesalahan penulisan, flash menyediakan fasilitas **check syntax** pada panel action.



```
103 if (Check Syntax m (Ke  
104 _xscale = -s)  
105 _x -= jarak1;
```

check syntax

Dengan menekan tombol **check syntax** tersebut, dapat diketahui apakah scrip yang ditulis benar atau memiliki kesalahan.

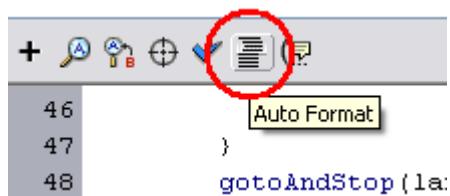


2 kemungkinan saat menekan tombol **check syntax**.

Kesalahan penulisan akan ditampilkan pada **output panel**. Scene, layer, dan frame tempat kita melakukan kesalahan serta baris ke berapa yang salah akan membantu kita dalam memperbaiki sebuah kesalahan.

Kesalahan yang sering muncul antara lain adalah :

1. Kesalahan penulisan action yang sederhana, contoh : action **duplicateMovieClip** salah ketik menjadi **duplicatMovieClip**.
2. Kesalahan penggunaan spasi, menuliskan nama instance atau variabel dengan tanda spasi, contoh : **nilai saya = 100;** seharusnya ditulis **nilai\_saya = 100;** Tips dari penulis : ketika kita menuliskan sebuah action jangan terlalu sering menggunakan spasi, sebab flash dapat menata script secara otomatis apabila kita menekan tombol **auto format**.



```
46 }  
47 }  
48 gotoAndStop (la:
```

auto format

3. Kesalahan penggunaan huruf kecil dan huruf besar. Action script flash sebagian besar tidak **sensitive case** (peka terhadap huruf kecil dan huruf besar). Akan tetapi flash membedakan sebuah teks dengan warna yang berbeda. Ketika kita menuliskan action tanpa memperhatikan huruf kecil dan huruf besar, maka teks tersebut akan berwarna hitam, sedangkan apabila benar maka teks akan berwarna biru gelap. Pembedaan warna teks dalam action script adalah sebagai berikut :
  1. Warna biru gelap digunakan sebagai warna action.
  2. Warna biru terang digunakan sebagai warna string.
  3. Warna abu-abu digunakan sebagai warna komentar.
  4. Warna hitam digunakan untuk menuliskan text selain 3 jenis text di atas.
4. Kesalahan penggunaan tanda “{}”. Seorang yang baru belajar flash sering kebingungan dalam menggunakan tanda tersebut. Tanda tersebut merupakan sebuah tanda blok action, dalam artian seluruh action yang berada dalam tanda “{}” berarti masuk kedalam blok tertentu seperti blok **movie event, mouse event, function, blok if, blok for** dan sebagainya. Jumlah tanda “{“ harus sama dengan jumlah tanda “}”. Tips dari penulis adalah : setelah kita menuliskan tanda “{“, segeralah menuliskan tanda “}”, kemudian letakkan kursor di tengah karakter tersebut dan tekan enter atau langsung tuliskan action yang dibutuhkan di dalamnya.
5. Kesalahan penggunaan tanda “()“. Berbeda dengan tanda “{}” tanda “()” bukan menyatakan suatu blok tetapi biasanya merupakan bagian dari penulisan action digunakan. contoh : if (...){} , trace(...);.. Jumlah tanda “(“ daam satu baris perintah harus sama dengan jumlah tanda “)”.
6. Kesalahan penggunaan tanda = dan tanda ==. Tanda == hanya digunakan pada suatu kondisi. Sedangkan tanda = digunakan untuk mengeset nilai suatu variabel
7. Kesalahan penggunaan titik. Dalam bahasa pemrograman apapun, karakter titik (“.”) adalah karakter yang vital, sehingga harus diperhatikan penggunaannya.

### **Kesalahan Logika Action Script**

Kesalahan logika tidak dapat dideteksi oleh tombol **check syntax**, karena kesalahan ini berasal dari programmernya. Kesalahan logika sebenarnya adalah kesalahan dalam menyusun dan menggunakan action script, sehingga menyebabkan program yang

dibuat berjalan lambat, ada bagian yang seharusnya bergerak menjadi tidak bergerak, memakan memori komputer dan sebagainya yang sering disebut dengan istilah **bug**.

Untuk memperbaiki **bug** jalankan terlebih dahulu, kemudian ujilah program tersebut sesering mungkin dan carilah peluang yang dapat terjadi sebanyak mungkin. Ketika menemukan bagian yang salah, kembali ke action dan carilah bagian mana yang menyebabkan kesalahan tersebut, kemudian perbaiki sebisa mungkin.

## Menggunakan Action Script Dasar pada Flash

Setelah mempelajari pola penulisan scrip dalam flash, langkah selanjutnya adalah memahami penggunaan action script. Setiap action memiliki kegunaan masing-masing, script mana dan kapan kita menggunakannya adalah hal yang harus dilatih secara terus-menerus apabila ingin menjadi pakar dalam pemrograman action script flash.

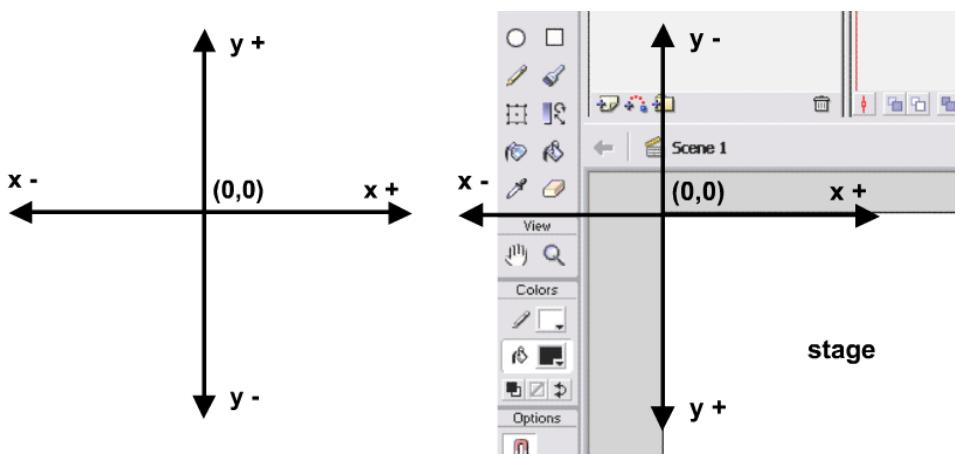
### Mengenal sistem kordinat layar

Pada mata pelajaran matematika, kita mengenal sistem kordinat kartesian.

Pengetahuan yang kuat akan matematika, akan sangat mendukung kita nantinya dalam memahami sistem kordinat layar.

Pada sistem kordinat kartesian sumbu utama dibagi menjadi sumbu x (horisontal) dan sumbu y (vertikal). Persimpangan antara sumbu tersebut merupakan kordinat (0,0). Pada sumbu x semakin ke kiri maka nilai x semakin kecil dan sebaliknya semakin ke kanan nilai x semakin besar. Pada sumbu y semakin ke bawah maka nilai y semakin kecil dan sebaliknya semakin ke atas nilai y semakin besar.

Pada sistem kordinat layar komputer, hampir sama dengan kordinat kartesian, sumbu utama dibagi menjadi sumbu x (horisontal) dan sumbu y (vertikal). Akan tetapi kordinat (0,0) adalah pojok kiri atas layar. Perbedaan selanjutnya adalah pada sumbu y. Pada sumbu y semakin ke bawah maka nilai y semakin besar dan sebaliknya semakin ke atas nilai y semakin kecil. Perhatikan gambar berikut:



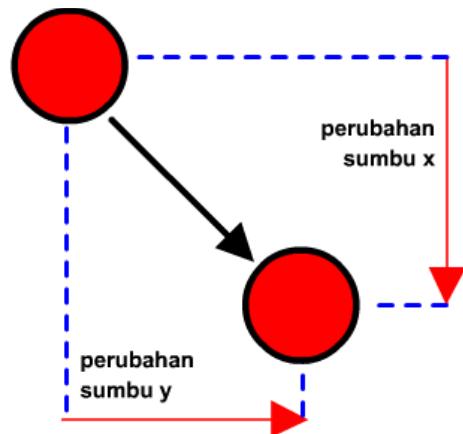
kordinat kartesian dan kordinat layar pada flash

Contoh penerapan dari sistem kordinat layar dalam flash adalah sebagai berikut :

Permasalahan :

Buatlah sebuah bola yang bergerak secara diagonal dari kiri atas ke kanan bawah

untuk menyelesaikan soal tersebut, kita gambarkan dahulu secara visual sebagai berikut :



pergerakan bola secara diagonal

Jadi untuk menggerakan bola secara diagonal, maka pada saat yang bersamaan sumbu x digerakkan ke kanan (ditambah) dan sumbu y digerakkan ke bawah (ditambah).

Proses pembuatan aplikasinya adalah sebagai berikut :

1. Buatlah sebuah obyek lingkaran dengan menggunakan oval tool.



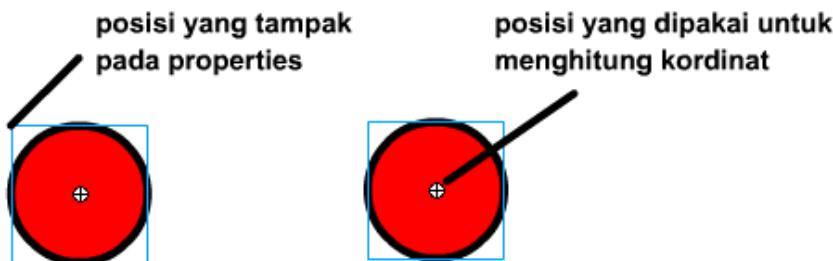
Oval tool dalam flash

Seleksi obyek lingkaran tersebut kemudian convert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name. Perhatikan bagian **registration**, bagian tersebut menentukan posisi suatu movieclip. Sebagai contoh, ketika kita

2. memilih registration pada bagian tengah, maka posisi penghitungan kordinat movieclip nantinya dihitung di bagian tengah tidak seperti yang terlihat pada properties (karena pada properties hitungan posisi dimulai dari pojok kiri atas).



Posisi registration



Perhitungan kordinat suatu symbol

3. Klik movieclip **bola** ubah posisinya tau letakan di pojok kanan atas stage, dan dalam kondisi terseleksi buka panel action ( tekan f9 atau klik menu windows>action apabila panel action belum terbuka dan pastikan mode penulisan adalah mode expert jika menggunakan flash MX atau Flash 8). Kemudian ketikan script sebagai berikut :

```
onClipEvent (enterFrame) {
    // menggerakan bola diagonal
    _x += 10;
    _y += 10;
}
```

4. Jalankan movie dengan menekan tombol Ctrl+Enter. Maka bola akan bergerak diagonal dari kiri atas ke kanan bawah sampai hilang dari layar. Tekan tombol Ctrl+W untuk kembali ke stage

Penjelasan program

1. Baris **onClipEvent (enterFrame) {** berarti bahwa perintah dalam blok {} akan dijalankan sepanjang frame aktif artinya pada default setting frame rate : 12 fps (frame per second), maka perintah dalam blok {} akan dijalankan setiap seperduabelas detik sepanjang movie tersebut dijalankan.
2. Baris **// menggerakan bola diagonal** merupakan baris komentar, baris ini tidak dijalankan oleh flash.
3. Baris **\_x += 10;** berarti setiap seperduabelas detik kordinat x dari bola ditambah 10 pixel, sehingga bola bergerak kekanan. Tanda += merupakan bentuk penyederhanaan dari action **\_x = \_x+10;**
4. Baris **\_y += 10;** berarti setiap seperduabelas detik kordinat y dari bola ditambah 10 pixel, sehingga bola bergerak kebawah.
5. Karena komputer menghitung dengan sangat cepat, perhitungan dalam satu blok {} dapat dikerjakan dengan sekaligus sehingga bola tampak bergerak diagonal.

## Mengenal variabel

Variabel adalah suatu lokasi didalam memori komputer yang digunakan untuk menyimpan suatu nilai dan nilai yang ada didalamnya dapat diubah. terdapat 2 cara dalam membuat variabel dalam flash cara yang pertama adalah dengan menuliskan langsung nama variabel dan nilainya contoh :

**score = 1000;** dengan action tersebut secara otomatis akan membuat satu variabel baru bernama score dengan nilai 1000;

Cara yang kedua adalah dengan menuliskan action **var** di depan variabel yang kita tuliskan, contoh :

**var score = 1000;** dengan action tersebut secara otomatis akan membuat satu variabel baru bernama score dengan nilai 1000;

aturan pemberian nama variabel :

8. variabel tidak boleh sama dengan nama instansi suatu symbol.
9. Tidak boleh menggunakan spasi, titik, dan tanda baca yang lain.
10. Tidak boleh menggunakan nama yang sama dengan suatu perintah dalam flash misalnya : **obyek**, **level**, **break** dan sebagainya.
11. Tidak boleh menggunakan angka saja untuk nama akan tetapi variabel boleh diikuti dengan angka. Contoh yang salah :**12**, **13**, dan sebagainya, contoh yang benar : **nilai1**, **nilai\_2** dan sebagainya.
12. Tidak sensitive case, boleh menggunakan huruf besar atau huruf kecil.

### Tipe variabel

dalam flash terdapat beberapa tipe variabel, beberapa yang sering dipakai antara lain :

Tipe Variabel	Contoh Penulisan
Number	score = 1000; i++;
String	nama = “”; teks = “contoh”;
Boolean	mati = true; kalah = false;
array	datapemain = []; hari = [“senin”, “selasa”] highscore = [100, 80, 75]

### Mengenal obyek properties

Sebuah obyek dalam flash bisa berupa movieclip, button, graphic, sound, video, bitmap dan component. Akan tetapi yang dapat diaplikasikan dengan action hanya obyek movieclip, button dan sound. Component memiliki build in action script, sehingga dapat langsung kita pakai, sedangkan bitmap dan graphic tidak dapat dimasuki oleh action.

Setiap obyek yang dapat dimasuki action memiliki beberapa properties seperti kordinat x, kordinat y, panjang, tinggi, frame yang aktif, warna dan sebagainya. Karena yang akan sering kita pakai adalah movieclip, berikut ini adalah beberapa properties dari sebuah movieclip:

**\_x**

\_x merupakan koordinat sumbu x suatu symbol terhadap stage. Perhatikan pada pelajaran sebelumnya bahwa perhitungan x dihitung dari titik registration nya, bukan dari pojok kiri atas seperti yang terlihat pada properties.

### **\_y**

\_y merupakan koordinat sumbu y suatu symbol terhadap stage. Seperti halnya \_x, perhitungan koordinat sumbu \_y pusat point registration dari symbol terhadap stage. Perhatikan action script berikut :

```
onClipEvent(load){  
    _x = 100;      // kordinat sumbu x  
    _y = 100;      // kordinat sumbu y  
}
```

Action diatas akan mengeset posisi obyek (symbol) kekordinat (100,100) terhadap stage saat symbol tersebut ditampilkan pertama kali.

### **\_xscale**

\_xscale merupakan presentase dari skala pembesaran horisontal suatu symbol terhadap titik regristasi. Nilai default \_xscale sebelum diubah adalah 100. Obyek yang memiliki nilai \_xscale dibawah 100 akan memiliki ukuran yang lebih kecil dari ukuran aslinya, begitu pula sebaliknya.

### **\_yscale**

\_yscale merupakan presentase dari skala pembesaran vertikal suatu symbol terhadap titik regristasi. contoh :

```
onClipEvent(load){  
    _xscale = 200; // pembesaran 2 kali ke sumbu x  
    _yscale = 200; // pembesaran 2 kali ke sumbu y  
}
```

Action diatas memperbesar symbol menjadi 2 kali lebih besar pada saat symbol tersebut ditampilkan pertama kali.

### **\_visible**

\_visible adalah property pada symbol yang menentukan apakah obyek tersebut ditampilkan atau tidak pada stage. \_visible bernilai 1 agar obyek tampak dan bernilai 0 agar obyek tidak tampak.

contoh :

```
onClipEvent(load){  
    _visible = 0; // tidak tampak  
}
```

Action diatas menyebabkan obyek tidak tampak pada stage.

### **\_alpha**

\_alpha adalah property pada symbol yang menentukan alpha value (transparency) obyek tersebut. \_alpha bernilai 0 sampai 100 dimana nilai 0 berarti obyek tidak tampak (transparan) dan 100 berarti obyek tampak (solid).

contoh :

```
onClipEvent(load){  
    _alpha = 50; // semi transparan  
}
```

Action diatas menyebabkan obyek mengalami tranparansi sebanyak 50%.

### **\_rotation**

\_rotation merupakan property yang menunjukkan posisi perputaran obyek tersebut searah jarum pada titik registrationnya.

contoh :

```
onClipEvent(enterFrame){  
    _rotation += 10; // berputar searah jarum jam  
}
```

Action diatas menyebabkan obyek berputar searah jarumjam dengan pusat titik registrationnya.

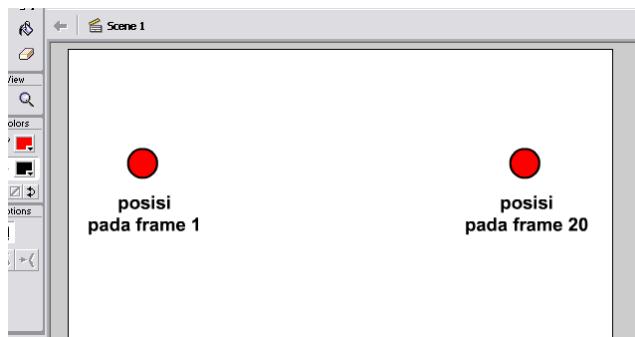
### **\_currentframe**

nilai pada `_currentframe` menunjukkan frame yang aktif pada sebuah movie atau sebuah movieclip. Penjelasan yang lebih mendetail terdapat pada bab selanjutnya.

### Menghentikan Movie dengan `stop()`:

action `stop()` digunakan untuk menghentikan movie yang sedang berjalan. Untuk lebih jelasnya perhatikan contoh berikut :

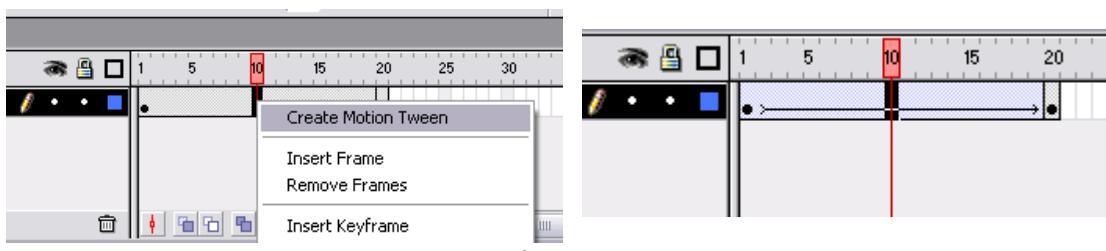
1. Buatlah sebuah bola dengan oval tool, kemudian convert menjadi symbol **movieclip** dengan nama **bola**.
2. Seleksi bola, kemudian letakkan disebelah kiri stage.
3. Klik frame 20, kemudian tambahkan keyframe dengan menekan tombol F6.



Kemudian geser bola ke kanan stage.

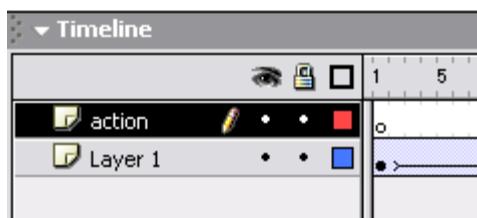
posisi movieclip bola pada frame 1 dan frame 20

4. Klik kanan pada timeline frame 10, kemudian pilih **create motion tween**.



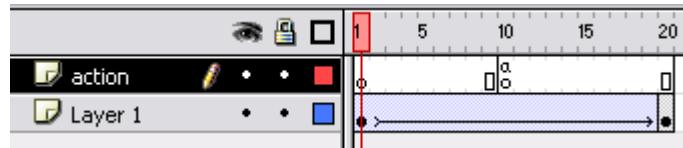
Create motion tween

5. Jalankan movie dengan menekan tombol **Ctrl+Enter**. Maka akan didapatkan sebuah gerakan bola dari kiri ke kanan secara looping
6. Kembali ke stage dengan menekan tombol **Ctrl+W**, kemudian tambahkan sebuah layer baru (layer **action**).



penambahan layer action

7. Klik frame 10 layer action, kemudian masukan keyframe dengan menekan tombol F6. Dalam keadaan frame 10 terseleksi, buka panel action dan ketikan action **stop();**



script stop(); pada frame 10 layer action

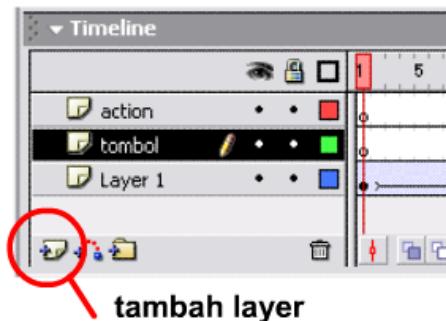
8. Jalankan Movie dengan menekan tombol Ctrl+Enter, maka bola akan berhenti saat berada di tengah stage. Hal tersebut diakibatkan karena adanya action stop(); sehingga movie berhenti.
9. Simpan file tersebut.

### Menjalankan Movie dengan play();

action play() digunakan untuk menjalankan frame yang berhenti akibat script stop(); .

Untuk lebih jelasnya perhatikan contoh berikut :

1. Lanjutkan file latihan sebelumnya (penggunaan stop()).
2. Klik layer 1, kemudian tambahkan layer baru diatasnya. Ubah nama layer menjadi layer **tombol**.



menambah layer

3. Klik frame 10 layer **tombol**, kemudian masukan keyframe. Buatlah sebuah bentuk tombol di tengah stage.



tombol pada frame 10 layer tombol

4. Seleksi bentuk tombol tersebut dan convert menjadi symbol dengan menekan tombol F8. Ketikan nama “play” dan pilih button pada behaviour.
5. Dalam kondisi tombol “play” terseleksi, buka panel action dan ketikan script berikut :

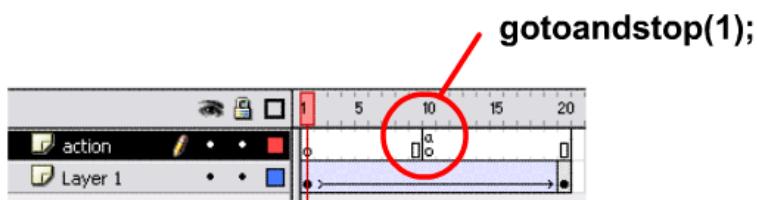
```
on (release) {
    //jika ditekan maka mainkan frame selanjutnya
    play();
}
```

6. Jalankan movie. Pada saat bola berhenti, tekan tombol dan bola akan bergerak lagi. Action **play()**; pada tombol menyebabkan movie berjalan kembali ketika tombol ditekan.

### **gotoAndStop(frame)**

Action **gotoAndStop()** digunakan untuk menghentikan movie pada frame yang inginkan. frame dapat diisi dengan menggunakan angka tempat frame yang diinginkan atau frame label. Perhatikan contoh berikut :

1. Buatlah sebuah movieclip **bola** dan letakan disebelah kiri stage.
2. Klik frame 20 kemudian masukan keyframe. Geser posisi movieclip **bola** ke sebelah kanan stage.
3. Klik kanan frame 10 kemudian pilih **create motion tween**, maka akan terbentuk animasi bola bergerak dari kiri ke kanan secara looping.
4. Tambahkan sebuah layer diatas **layer 1** dan ubah namanya menjadi layer **action**.
5. Klik frame 10 layer **action**, masukan keyframe. Kemudian buka panel action dan ketikan action **gotoAndStop(1);**



Peletakan script gotoAndStop(1);

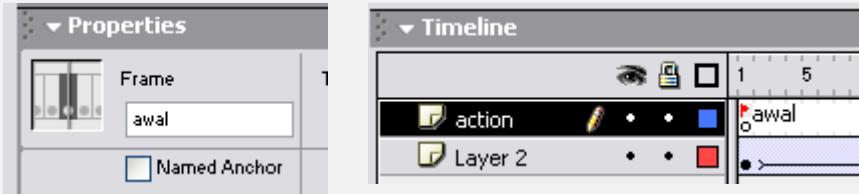
6. Jalankan movie, maka ketika bola sampai di tengah stage, bola akan kembali dan berhenti ke posisi awal di kiri stage.

### **gotoAndPlay(frame)**

Action **gotoAndPlay()** digunakan untuk memainkan movie pada frame yang inginkan, merupakan kebalikan dari action **gotoAndStop()**;

catatan : suatu ketika terdapat action **gotoAndStop("awal")**;

kata "awal" merupakan **frame label**. untuk menambahkan frame label, masukan keyframe pada frame yang diinginkan, buka panel properties dan ketikan pada textbox <frame label>.



### **OnClipEvent(event)**

OnClipEvent adalah suatu script yang digunakan oleh suatu movie klip untuk melaksanakan beberapa perintah lain dalam suatu blok event. Parameter Event dapat berisi :

**load** perintah akan dijalankan ketika pertama kali movie klip di load oleh time line

**unload** perintah akan dijalankan ketika suatu movie klip di remove( dihilangkan ) dari time line

**enterFrame** perintah kan dijalankan secara terus menerus sepanjang frame yang aktif

**mouseMove** perintah akan dijalankan ketika mouse digerakkan

**mouseDown** perintah akan dijalankan ketika mouse ditekan

**mouseUp** perintah akan dijalankan ketika mouse dilepaskan

**keyDown** perintah akan dijalankan ketika mendeteksi adanya tombol yang ditekan dengan menggunakan action Key.getCode

**keyUp** perintah akan dijalankan ketika mendeteksi adanya tombol yang dilepaskan dengan menggunakan action Key.getCode

Pemahaman lebih lanjut mengenai script ini akan terbentuk seiring dengan latihan kita dalam mengolah action script.

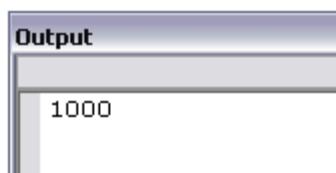
### Trace(pesan)

Perintah **trace()** digunakan untuk menampilkan sesuatu pada **output panel**.

Penggunaan action trace sangat bermanfaat dalam pemrograman lebih lanjut nantinya, sebagai penguji kesalahan akan suatu baris program. Perhatikan contoh penggunaan action script trace berikut:

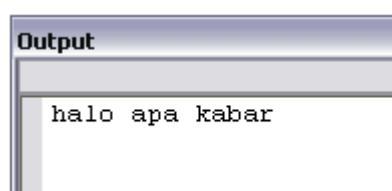
1. Klik frame 1, kemudian buka panel action dan ketikan script:

**trace(1000);** kemudian jalankan movie. Maka akan muncul text “1000” pada **output panel**.



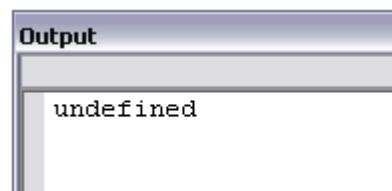
tampilan pada output

2. Tekan Ctrl+W untuk kembali ke Stage. Selanjutnya ubah action pada frame 1 menjadi : **trace("halo apa kabar?");** Kemudian jalankan kembali movie. Maka tampilan yang muncul adalah sebagai berikut :



tampilan pada output

3. Tekan Ctrl+W untuk kembali ke Stage. Selanjutnya ubah action pada frame 1 menjadi : **trace(nilai);** Kemudian jalankan kembali movie. Maka tampilan yang muncul adalah sebagai berikut :

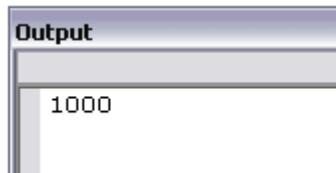


tampilan pada output

4. Tekan Ctrl+W untuk kembali ke Stage. Selanjutnya tambahkan action pada frame 1 menjadi :

```
nilai = 1000;  
trace(nilai);
```

Kemudian jalankan kembali movie. Maka tampilan yang muncul adalah sebagai berikut :



tampilan pada output

Kesimpulan yang dapat diambil adalah sebagai berikut :

1. nilai di dalam tanda () pada action trace() dapat diisi dengan bilangan.
2. kata "**halo apa kabar?**" dapat ditampilkan karena merupakan obyek string (berada didalam tanda "").
3. kata **nilai** tidak dapat ditampilkan, karean tidak didalam tanda "" maka kata **nilai** dianggap sebagai sebuah **variabel** yang harus didefinisikan terlebih dahulu.
4. penambahan action **nilai = 1000;** merupakan sebuah pendefinisian variabel **nilai** sehingga variabel tersebut bernilai 1000 dan dapat ditampilkan oleh action **trace()**.

## Menggunakan Operator Matematika pada Flash

Pada bab sebelumnya dijelaskan bahwa matematika memegang peranan penting dalam pemrograman. Flash menunjang operasi matematika dengan banyak actionscrpit, meskipun demikian tidak semua script yang harus kita kuasai. Hal mendasar yang harus dikuasai adalah operator aritmatika. Bentuk operator aritmatika dasar dalam flash adalah sebagai berikut:

Operator	Fungsi
+	penambahan
-	pengurangan
*	perkalian
/	pembagian

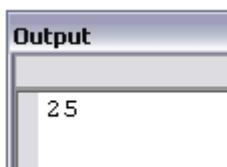
contoh aplikasi aritmatika pada flash:

1. Klik frame 1, kemudian ketikan action **trace( 5 + 4 );** Kemudian jalankan movie. Maka pada output panel akan muncul tampilan sebagai berikut :



tampilan hasil penjumlahan

2. Tekan Ctrl+W untuk kembali ke Stage. Selanjutnya ubah action pada frame 1 menjadi : **trace(5+4\*5);** Kemudian jalankan kembali movie. Maka tampilan yang muncul adalah sebagai berikut:



tampilan hasil penjumlahan

Perhatikan bahwa hasil yang ditampilkan adalah 25 bukan 45 hal tersebut dikarenakan kedudukan perkalian dan pembagian lebih tinggi daripada penjumlahan dan pengurangan, sehingga operasi perkalian dan pembagian didahulukan.

Operator ++ berarti penambahan suatu variabel dengan angka 1. Perhatikan contoh berikut :

1. Buatlah sebuah movieclip **bola**, dan letakkan disebelah kiri frame. Dalam kondisi **bola** terseleksi, buka panel action dan ketikan script berikut :

```
onClipEvent (enterFrame) {  
    _x++;  
}
```

2. Jalankan movie, maka **bola** akan bergerak dari kiri ke kanan secara perlahan.

Hal tersebut dikarenakan setiap seperduabelas detik (bila setting default movie adalah 12 fps), kordinat x bola ditambah 1 pixel.

--

Sebaliknya operator -- berarti pengurangan suatu variabel dengan angka 1.

### Penggunaan variabel dengan operator aritmatika

Dalam pemrograman penghitungan variabel hampir selalu terjadi. Untuk mempermudah penulisan, flash menyediakan beberapa format penulisan, antara lain sebagai berikut:

Contoh kita akan mengubah nilai kordinat x suatu movieclip maka bentuk penulisannya :

bentuk standart	penyederhanaan
$x = x + 10$	$x += 10$
$x = x - 10$	$x -= 10$
$x = x * 10$	$x *= 10$
$x = x / 10$	$x /= 10$

### Penggunaan Script Math

**Math** merupakan sebuah obyek yang memiliki sub script lain. Obyek Math memiliki 18 action dan 8 script properti. 18 Action tersebut antara lain adalah : **Math.abs()**, **Math.sin()**, **Math.random()** dan sebagainya. Sedangkan 8 propertinya antara lain adalah : **Math.PI**, **Math.LN10** dan sebagainya. Penggunaan script Math dalam game tidak terlalu sering dipakai.

## Operasi Teks (String)

Teks atau string dalam flash juga merupakan suatu obyek yang memiliki banyak action. Perhatikan beberapa penulisan action script berikut :

**“halo apa kabar”** merupakan string karena terdapat dalam tanda “”;

**halo apa kabar** merupakan kesalahan penulisan script karena bukan string dan memiliki karakter spasi sehingga bukan variabel.

**nilai** merupakan sebuah variabel (akan tetapi tipe variabel belum dapat ditentukan apakah bertipe string atau bertipe integer).

**nilai = 1000** bukan string akan tetapi berupa variabel bertipe **number**

**nilai = “1000”** merupakan variabel bertipe string karena penambahan tanda “” didefinisikan sebagai teks 1000 bukan angka 1000.

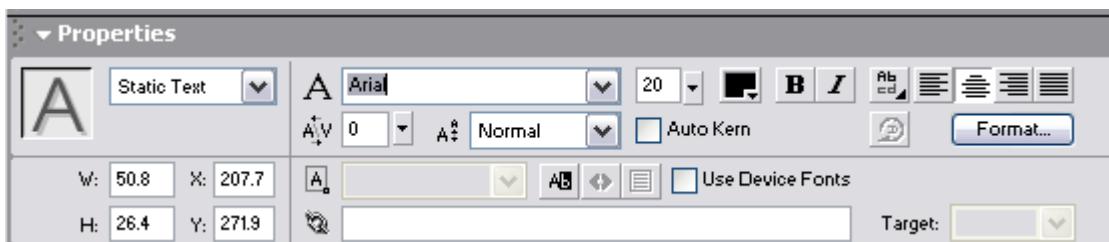
**nilai = “”** merupakan variabel bertipe string

## Menampilkan Teks (String) dalam flash

Untuk menampilkan teks dalam flash kita dapat menggunakan text tool. Terdapat 3 tipe teks yang bisa dibentuk oleh teks tool yaitu **static text**, **dynamic text** dan **inputtext**.

### Static Text

Static text digunakan untuk menampilkan suatu teks yang sifatnya tidak dapat diubah secara manual atau dengan script setelah movie dipublish. Untuk mengatur jenis, bentuk, ukuran dan warna text dapat menggunakan properties.

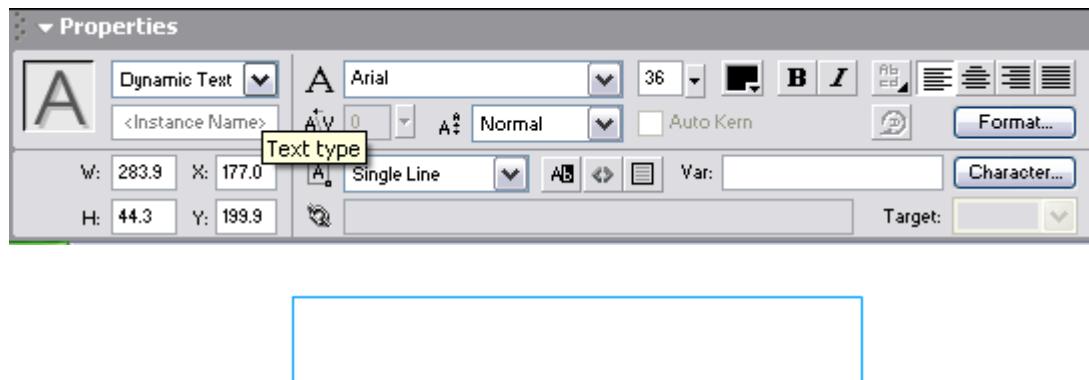


text tool properties

## Dynamic Text

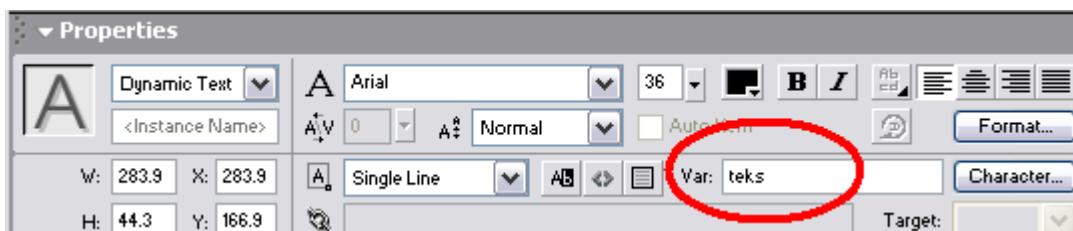
Berbeda dengan static text dynamic text dapat diubah dengan menggunakan actionscript. Penggunaan dynamic text biasanya adalah untuk menampilkan suatu variabel. Perhatikan contoh aplikasi penggunaan dynamic text berikut :

1. Klik text tool kemudian buka panel properties. Pilihlah Dynamic text pada option text type dan atur bentuk, ukuran dan warna huruf sesuai dengan keinginan.
2. Klik dan drag pada stage untuk membentuk sebuah text area



dynamic text properties dan text area yang sudah dibuat

3. Seleksi text area (berwarna biru jika terseleksi) dan buka panel properties. Ketikan kata “**teks**” pada **var**. Var merupakan nama variabel yang akan ditampilkan oleh dynamic text terpilih.



var pada dynamic text properties

4. Buatlah sebuah layer baru diatas layer 1 dan ubah namanya menjadi **layer action**.
5. Klik frame 1 **layer action**, buka panel action dan tuliskan action  
**teks = “contoh dynamic text”**.
6. Jalankan Movie, maka akan muncul tampilan yang sama dengan nilai variabel **teks**.

Permasalahan :

buatlah sebuah program yang menampilkan angka yang selalu bertambah dari waktu ke waktu.

Jawaban dari permasalahan tersebut adalah sebagai berikut :

1. Klik text tool kemudian buka panel properties. Pilihlah Dynamic text pada option text type dan atur bentuk, ukuran dan warna huruf sesuai dengan keinginan.
2. Klik dan drag pada stage untuk membentuk sebuah text area
3. Seleksi text area (berwarna biru jika terseleksi) dan buka panel properties. Ketikan kata “**angka**” pada var.



penulisan variabel “angka”

4. Buatlah sebuah layer baru diatas layer 1 dan ubah namanya menjadi **layer action**.
5. Klik frame 1 **layer action**, buka panel action dan tuliskan action

```
_root.onEnterFrame = function(){
    angka++;
}
```

6. Jalankan Movie, maka akan muncul tampilan angka yang terus bertambah.

Penjelasan program:

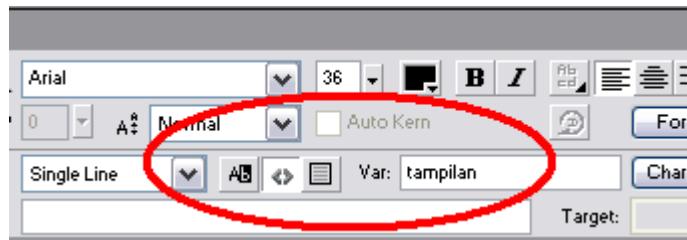
1. Baris **\_root.onEnterFrame = function()**{ menyatakan bahwa perintah dalam blok {} akan dijalankan secara belulang-ulang setiap seperduabelas detik sekali. Penjelasan lebih lanjut mengenai action **\_root** terdapat pada bab selanjutnya
2. baris **angka++** menyatakan bahwa setiap seperduabelas detik variabel angka ditambah satu satuan.

### **Dynamic Text : render text as HTML**

Pada dynamic text properties terdapat tombol **render text as HTML**. Dengan tombol tersebut terpilih, maka dynamic text dapat menampilkan sebuah perintah HTML. Agar lebih jelas perhatikan contoh berikut :

1. Klik text tool kemudian buka panel properties. Pilihlah Dynamic text pada option text type dan atur bentuk, ukuran dan warna huruf sesuai dengan keinginan.
2. Klik dan drag pada stage untuk membentuk sebuah text area
3. Seleksi text area (berwarna biru jika terseleksi) dan buka panel properties. Ketikan kata “**tampilan**” pada var. dan aktifkan **tombol render text as HTML**

penuisan variabel “tampilan”



4. Buatlah sebuah layer baru diatas layer 1 dan ubah namanya menjadi **layer action**.
5. Klik frame 1 **layer action**, buka panel action dan tuliskan action  
**tampilan = "<b>kata ini dicetak tebal </b> sedangkan yang ini dicetak <i>miring</i>";**
6. Jalankan Movie, maka akan muncul tampilan sebagai berikut :

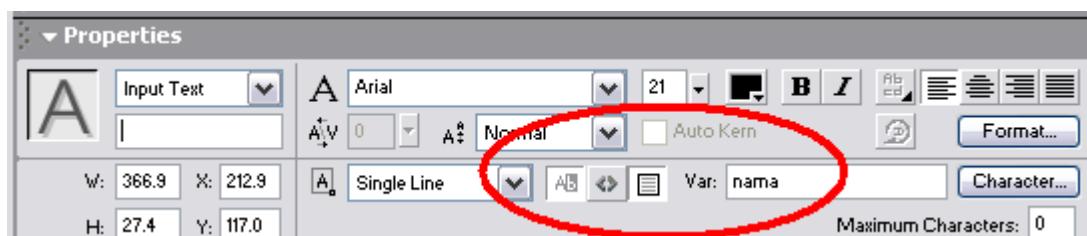
**kata ini dicetak tebal** sedangkan yang ini dicetak *miring*

dalam variabel **tampilan** tersebut terdapat beberapa **tag HTML** seperti tag **bold** (**<b>**) dan **tag italic** (**<i>**). Ketika tombol render text as HTML aktif maka tag-tag tersebut akan dijalankan oleh dynamic text sehingga dihasilkan tampilan seperti pada gambar diatas.

### Input Text

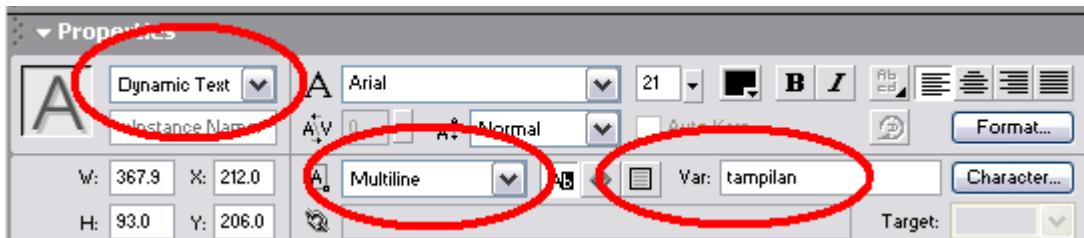
Input text merupakan suatu form yang meminta suatu masukan dari user (pengguna). Masukan tersebut nantinya akan dianggap sebagai nilai dari variabel text input tersebut. Agar lebih jelas perhatikan contoh berikut :

1. Klik text tool kemudian buka panel properties. Pilihlah input text pada option text type dan atur bentuk, ukuran dan warna huruf sesuai dengan keinginan.
2. Klik dan drag pada stage untuk membentuk sebuah text area
3. Seleksi text area (berwarna biru jika terseleksi) dan buka panel properties. Ketikan kata “**nama**” pada **var.** dan aktifkan tombol **show border around text**.

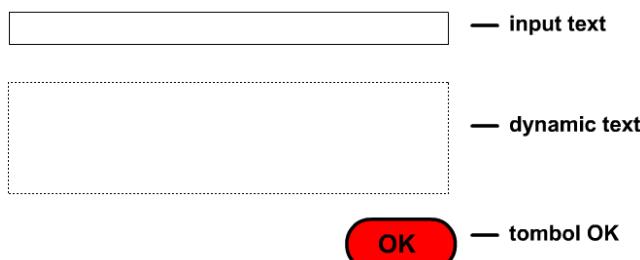


input text properties

- Buat lagi text area yang lebih besar (cukup untuk 2 baris kata atau lebih) di bawah text area yang sudah dibuat pada poin no 2.
- Klik text area yang bawah tersebut, kemudian buka properties dan ubah menjadi **dynamic text** dengan var “**tampilan**” dan pilih **multiline** pada linetype.



- Buatlah sebuah layer baru diatas layer 1 dan ubah namanya menjadi layer **tombol**.
- Buatlah sebuah tombol “**OK**” dan letakkan di bawah text area kedua. Perhatikan peletakan obyek di stage sebagai berikut :



penataan obyek pada stage

- Klik tombol “**OK**” kemudian buka panel action dan ketikan action sebagai berikut :

```
on (release) {
    tampilan = "haloo..."+nama+" apa kabar?, sedang belajar Flash ya?";
}
```

- Jalankan Movie, kemudian masukan nama anda kedalam input text dan tekan OK

action **tampilan** = "haloo..."+**nama**+" apa kabar?, sedang belajar Flash ya?"; menyatakan bahwa teks yang ditampilkan adalah "haloo" ditambah dengan **nama** yang dimasukan dan ditambah lagi dengan kata " apa kabar?, sedang belajar Flash ya?".

### **Mengubah variabel string ke number dengan action Number(string);**

Perhatikan contoh berikut :

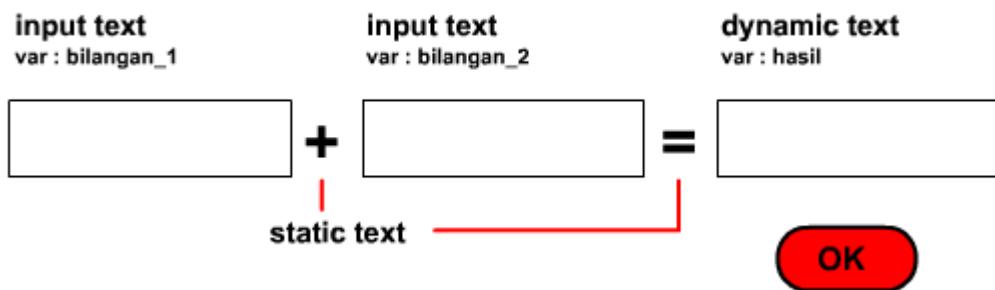
Permasalahan:

Buatlah sebuah program penjumlahan sederhana, pengguna memasukan 2 bilangan dan program menampilkan hasilnya.

Penyelesaian :

Untuk menjawab pertanyaan tersebut, perhatikan contoh berikut :

1. Klik text tool kemudian buka panel properties. Pilihlah **input text** pada option text type dan atur bentuk, ukuran dan warna huruf sesuai dengan keinginan.
2. Klik dan drag pada stage untuk membentuk sebuah text area
3. Seleksi text area (berwarna biru jika terseleksi) dan buka panel properties. Ketikan kata “**bilangan\_1**” pada **var.** dan aktifkan tombol **show border around text**.
4. Buat lagi input text area dengan ukuran yang sama dan letakkan di sebelah text area yang sudah dibuat pada poin no 2.
5. Seleksi input text area kedua (berwarna biru jika terseleksi) dan buka panel properties. Ketikan kata “**bilangan\_2**” pada **var.**
6. Buat lagi input text area dengan ukuran yang sama dan letakkan di sebelah text area yang sudah dibuat pada poin no 5.
7. Seleksi input text area ketiga (berwarna biru jika terseleksi) dan buka panel properties. Ubah tipe text menjadi **dynamci text** dan ketikan kata “**hasil**” pada **var.**
8. Buatlah **static text** “+” dan “=” kemudian atur posisinya (lihat gambar).
9. Buatlah sebuah tombol “**OK**” dan letakkan di bawah text area kedua. Perhatikan peletakan obyek di stage sebagai berikut :



penataan obyek pada stage

10. Klik tombol “**OK**” kemudian buka panel action dan ketikan action sebagai berikut :

```
on (release) {  
    hasil = bilangan_1 + bilangan_2;  
}
```

11. Jalankan Movie, masukan angka **10** pada bilangan 1 dan **20** pada bilangan 2. Kemudian tekan tombol **OK** hasil yang diperoleh adalah **1020** bukan **30**. Hal tersebut dikarenakan hasil yang diperoleh dari **input text selalu bertipe string**, sehingga penjumlahan terhadap 2 bilangan tersebut merupakan penjumlahan string “**10**” + string “**20**” dan hasilnya “**1020**”. Untuk mengatasinya kita harus mengubah terlebih dahulu variabel bertipe string ke variabel bertipe number.

12. Tekan Ctrl+W untuk kembali ke stage. Klik tombol **OK** kemudian ubah scriptnya menjadi :

```
on (release) {  
    hasil = Number(bilangan_1) + Number(bilangan_2);  
}
```

13. Jalankan Movie, kemudian masukan bilangan tertentu pada ke dua input text. Setelah tombol **OK** ditekan, hasil yang benar akan muncul. Action **Number(string)** digunakan untuk mengconvert suatu string yang berada dalam tanda () menjadi variabel bertipe number.

## Action String

String merupakan suatu obyek yang memiliki sekitar 12 action seperti : **slice**, **subString**, **toUpperCase**, **toLowerCase** dan sebagainya dan memiliki 1 property yaitu **length**. Penggunaan action-action tersebut jarang ditemui dalam pemrograman game, dan penerapannya dapat ditemui pada salah satu program di dalam bab selanjutnya

## Memahami Pernyataan if, Operator Logika dan Operasi Pengulangan

Dalam pemrograman salah satu elemen terpenting adalah penyusunan logika. Dan logika dalam pemrograman tidak terlepas dari pengambilan keputusan, operator logika dan pengulangan. Apabila kita memahami benar ke 3 hal tersebut, dapat dipastikan sebuah pemrograman game dapat dilakukan.

### Pernyataan if

Dengan bahasa bebas **if** bisa diterjemahkan menjadi “**jika...maka....**”. Dengan kata lain apabila terdapat pernyataan **if**, dapat dipastikan terdapat dua atau lebih kemungkinan. Sebagai contoh : jika pesawat meledak, maka game akan berakhir.

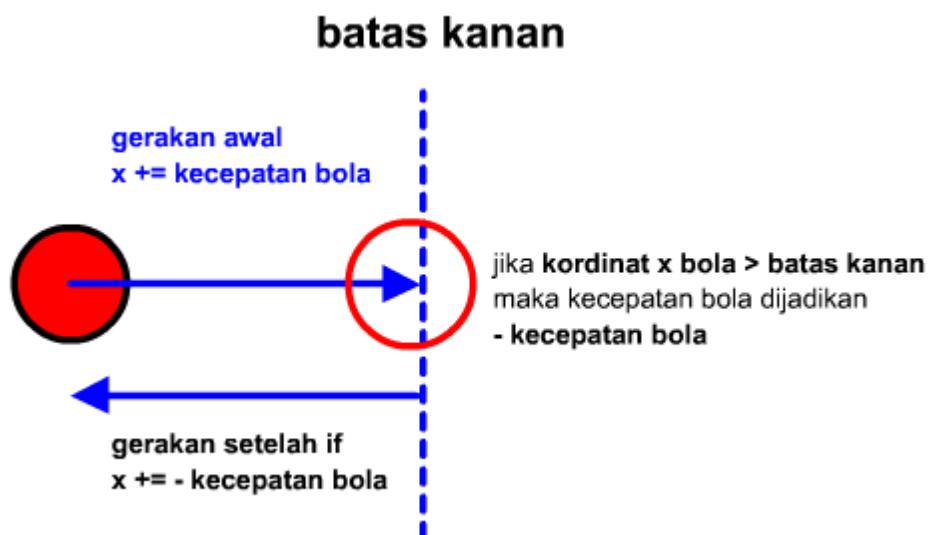
Bentuk penulisan if adalah sebagai berikut:

```
if ( kondisi ){  
    blok pernyataan bila kondisi bernilai benar  
} else{  
    blok pernyataan bila kondisi bernilai salah  
}
```

Perhatikan contoh penggunaan if berikut ini :

Permasalahan :  
gerakan sebuah bola kekanan kemudian setelah berada di batas kanan layar, maka bola bergerak kembali (memantul) ke kiri.

untuk menyelesaikan permasalahan tersebut, kita gambarkan dulu secara visual menjadi :



## logika sederhana pemakaian if

Proses pembuatan aplikasinya adalah sebagai berikut :

5. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps. untuk mengatur properti file dapat melalui panel properties (sebelumnya pastikan dahulu tidak ada obyek yang terseleksi dan tool yang aktif adalah arrow tool).



document properties

6. Buatlah sebuah obyek lingkaran dengan menggunakan oval tool.



Oval tool dalam flash

7. Seleksi obyek lingkaran tersebut kemudian convert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name. Letakkan bola di sebelah kanan stage
8. Klik movieclip **bola** kemudian buka panel action dan ketikan script berikut:

```
onClipEvent (load) {  
    kecepatan = 10;  
    bataskanan = 700;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan) {  
        kecepatan = -kecepatan;  
    }  
}
```

}

9. Jalankan Movie, maka ketika bola telah pada kordinat yang ditentukan pada variabel batas kanan, bola akan terpantul.

Penjelasan program :

1. baris **onClipEvent (load)** { merupakan sebuah perintah yang akan dijalankan satu kali saja, yaitu saat bola pertama kali di load oleh movie. Perintah tersebut sering kali digunakan sebagai alat untuk mengeset awal variabel-variabel yang akan dipakai.
2. baris **kecepatan = 10;** dan **bataskan = 700;** merupakan variabel yang akan dipakai. Kecepatan menentukan arah pergeseran kordinat x, karena bernilai positif, x akan bergeser ke kanan. Batas kanan digunakan sebagai pendekripsi apakah kondisi bola menyentuh batas kanan terjadi atau belum.
3. baris **onClipEvent (enterFrame){** berarti perintah yang berada dalam blok {} akan dijalankan secara berulang setiap seperduabelas detik secara terus menerus hingga movie ditutup.
4. baris **\_x += kecepatan;** berarti kordinat x dari bola akan ditambah sejumlah nilai dari variabel kecepatan.
5. baris **if (\_x>=bataskan){** berarti “jika kordinat x lebih besar atau sama dengan variabel bataskan maka” perintah dalam blok {} akan dijalankan.
6. baris **kecepatan = -kecepatan;** berarti apabila kondisi pada no 5 tercapai maka nilai kecepatan dijadikan negatif (menjadi -10), sehingga gerakan bola kekiri akibat script **\_x += kecepatan ( \_x += -10 ).**

## Operator logika

Pada matematika dikenal beberapa logika seperti **dan**, **atau**, dan **bukan**. Operator logika biasanya digunakan untuk membentuk suatu keadaan logika (boolean) atau kondisi bernilai benar atau salah. Dalam flash juga terdapat operator logika seperti **AND**, **OR**, dan **NOT**.

## Operator and

Operator **and** dalam flash dapat ditulis dalam 2 bentuk yaitu :

**kondisi 1 && kondisi 2** atau **kondisi 1 and kondisi 2**

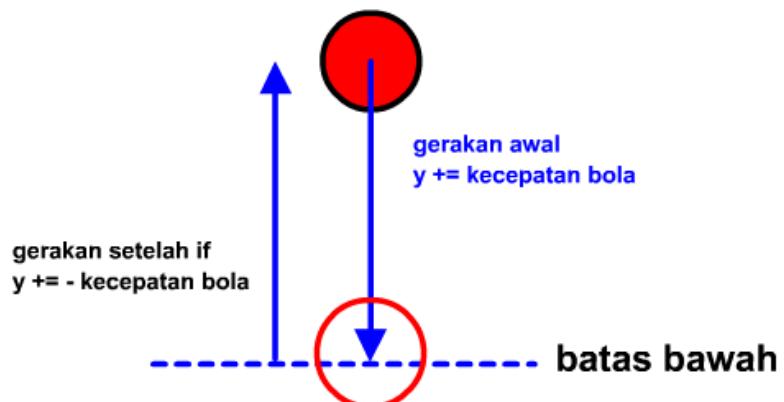
hasil dari operator **and** hanya bernilai benar jika kedua kondisi bernilai benar.

Perhatikan contoh berikut :

## Permasalahan

jatuhkan sebuah bola kemudian setelah berada di batas bawah layar, maka bola bergerak memantul ke atas dan berubah bentuk menjadi kotak.

untuk menyelesaikan permasalahan tersebut, kita gambarkan dulu secara visual menjadi :



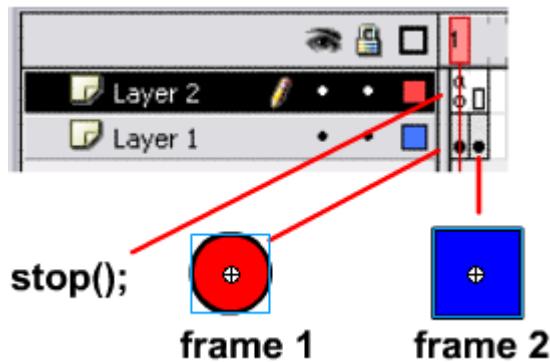
jika kordinat y bola > batas bawah dan bola aktif pada frame 1 (lingkaran)  
maka kecepatan bola dijadikan - kecepatan bola  
dan frame dipindah ke frame 2 (kotak)

penggunaan logika and

Proses pembuatan aplikasinya adalah sebagai berikut :

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps. untuk mengatur properti file dapat melalui panel properties (sebelumnya pastikan dahulu tidak ada obyek yang terseleksi dan tool yang aktif adalah arrow tool).
2. Buatlah sebuah obyek lingkaran dengan menggunakan oval tool.
3. Seleksi obyek lingkaran tersebut kemudian convert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name. Letakkan bola disebelah atas stage
4. Double klik movieclip **bola** untuk masuk kedalam **mode edit bola**.
5. Dalam mode edit **bola**, klik frame 2 kemudian masukan **blank Keyframe** dengan menekan tombol F7. Kemudian gambarlah sebuah kotak berwarna dengan rectangle tool.

tambahkan sebuah layer diatasnya, kemudian klik frame 1 **layer 2**, buka panel action dan tuliskan action **stop();**



susunan layer movieclip bola

6. Kembali ke stage utama dengan menekan Ctrl+E .
7. Klik movieclip **bola** kemudian buka panel action dan ketikan script berikut:

```
onClipEvent (load) {
    kecepatan = 10;
    batasbawah = 500;
}

onClipEvent (enterFrame) {
    _y += kecepatan;
    if (_y>=batasbawah and _currentframe==1) {
        kecepatan = -kecepatan;
        gotoAndStop(2);
    }
}
```

8. Jalankan Movie, maka ketika bola telah pada kordinat yang ditentukan pada variabel batas bawah, bola akan terpantul dan berubah bentuk menjadi kotak.

Penjelasan program :

1. baris **onClipEvent (load)** { merupakan sebuah perintah yang akan dijalankan satu kali saja, yaitu saat bola pertama kali di load oleh movie. Perintah tersebut sering kali digunakan sebagai alat untuk mengeset awal variabel-variabel yang akan dipakai.
2. baris **kecepatan = 10;** dan **batasbawah = 700;** merupakan variabel yang akan dipakai. Kecepatan menentukan arah pergeseran kordinat y, karena bernilai positif, y akan bergeser ke bawah. Batas bawah digunakan sebagai pendekripsi apakah kondisi bola menyentuh batas bawah layar terjadi atau belum.

3. baris **onClipEvent (enterFrame){** berarti perintah yang berada dalam blok {} akan dijalankan secara berulang setiap seperduabelas detik secara terus menerus hingga movie ditutup.
4. baris **\_y += kecepatan;** berarti kordinat y dari bola akan ditambah sejumlah nilai dari variabel kecepatan.
5. baris **if (\_y>=batasbawah and \_currentframe==1){** berarti “jika kordinat x lebih besar atau sama dengan variabel batasbawah **dan** frame yang aktif pada movieclip bola adalah frame 1, maka” perintah dalam blok {} akan dijalankan.
6. baris **kecepatan = -kecepatan;** berarti apabila kondisi pada no 5 tercapai maka nilai kecepatan dijadikan negatif (menjadi -10), sehingga gerakan bola menjadi ke atas akibat script **\_y += kecepatan ( \_y += -10 ).**
7. baris **gotoAndStop(2);** memindah frame aktif movieclip bola ke frame 2, sehingga bentuk bola berubah menjadi kotak.

catatan :

pada baris **if (\_y>=batasbawah and \_currentframe==1){** pada bagian **\_currentframe == 1**, terdapat penggunaan dua tanda =, penggunaan == hanya dalam suatu **kondisi**, sedangkan selain kondisi kita menggunakan satu tanda =.

## Operator or

Operator **or** dalam flash dapat ditulis dalam 2 bentuk yaitu :

**kondisi 1 || kondisi 2** atau **kondisi 1 or kondisi 2**

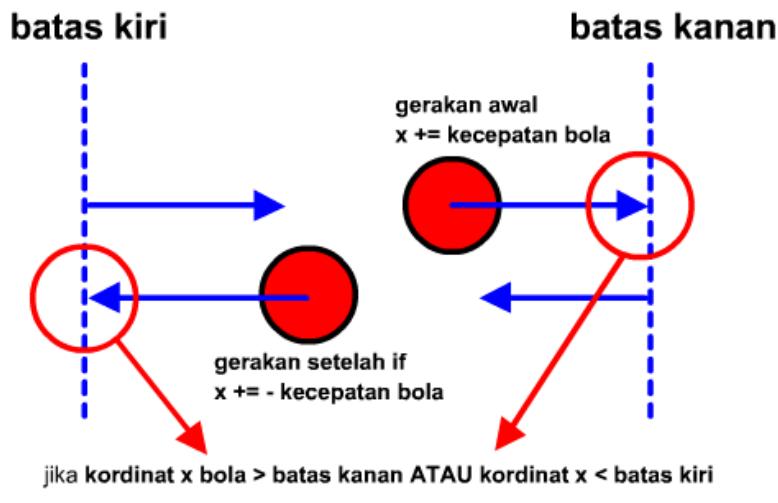
hasil dari operator **or** bernilai salah jika kedua kondisi bernilai salah.

Perhatikan contoh berikut :

Permasalahan

:  
gerakan sebuah bola kekanan kemudian setelah berada di batas kanan layar, maka bola bergerak kembali (memantul) ke kiri dan ketika bola berada dibatas kiri maka bola kembali memantul ke kanan dan seterusnya.

Untuk menyelesaikan permasalahan tersebut, kita gambarkan dulu secara visual menjadi :



penggunaan logika ATAU

Penyelesaian dari masalah diatas adalah sebagai berikut :

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps. document properties
2. Buatlah sebuah obyek lingkaran dengan menggunakan oval tool.
3. Seleksi obyek lingkaran tersebut kemudian convert menjadi symbol dengan menekan tombol F8 (pilih menu insert>convert to symbol). Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name. Letakkan bola disebelah kanan stage
4. Klik movieclip **bola** kemudian buka panel action dan ketikan script berikut:

```

onClipEvent (load) {
  kecepatan = 10;
  bataskanan = 700;
  bataskiri = 100;
}

onClipEvent (enterFrame) {
  _x += kecepatan;
  if (_x<bataskiri or _x>bataskanan) {
    kecepatan = -kecepatan;
  }
}
  
```

5. Jalankan Movie, maka ketika bola telah pada kordinat yang ditentukan pada variabel batas kanan, bola akan terpantul.

Penjelasan program :

1. baris **onClipEvent (load)** { merupakan sebuah perintah yang akan dijalankan satu kali saja, yaitu saat bola pertama kali di load oleh movie. Perintah tersebut sering kali digunakan sebagai alat untuk mengeset awal variabel-variabel yang akan dipakai.
2. baris **kecepatan = 10; bataskiri = 100;** dan **bataskanan = 700;** merupakan variabel yang akan dipakai. Kecepatan menentukan arah pergeseran kordinat x, karena bernilai positif, x akan bergeser ke kanan. Batas kanan dan batas kiri digunakan sebagai pendekripsi apakah kondisi bola menyentuh batas kanan atau bataskiri sudah terjadi atau belum.
3. baris **onClipEvent (enterFrame){** berarti perintah yang berada dalam blok {} akan dijalankan secara berulang setiap seperduabelas detik secara terus menerus hingga movie ditutup.
4. baris **\_x += kecepatan;** berarti kordinat x dari bola akan ditambah sejumlah nilai dari variabel kecepatan.
5. baris **if (\_x<bataskiri or \_x>bataskanan){** berarti “jika kordinat x lebih besar dari variabel bataskanan **atau** kordinat x lebih kecil dari variabel batas kiri maka” perintah dalam blok {} akan dijalankan.
6. baris **kecepatan = -kecepatan;** berarti apabila kondisi pada no 5 tercapai maka apabila nilai kecepatan positif akan dijadikan negatif (menjadi -10) dan sebaliknya jika nilai kecepatan negatif maka kecepatan akan diubah nilainya menjadi positif (menjadi -(-10) atau +10).

## Operator not

Operator not berarti tidak sesuai dengan kondisi. Operator ini jarang dipakai dalam sebuah aplikasi game. Operator **not** dalam flash dapat ditulis dalam 2 bentuk yaitu :

**!kondisi 1** atau **not kondisi 1.** Operator **not** akan bernilai benar jika kondisi yang dimasukan salah dan akan bernilai salah jika kondisi yang dimasukkan adalah benar.

## Operasi Pengulangan dengan for()

Pada dasarnya operasi pengulangan dalam flash dapat dilakukan oleh 3 macam script yaitu **for**, **while**, dan **do..while**. Akan tetapi pada buku ini dan buku seri pemrograman game lainnya yang sering dipakai untuk operasi pengulangan adalah

perintah **for**. Bentuk penulisan operasi for adalah sebagai berikut :

```
for (inisialisasi; kondisi; pengurangan atau penambahan){
```

**blok perintah yang dijalankan selama kondisi dipenuhi**

```
}
```

bagian **inisialisasi** digunakan untuk memberi nilai kepada variabel yang digunakan untuk mengontrol pengulangan.

bagian **kondisi** digunakan untuk menghentikan pengulangan

bagian **pengurangan atau penambahan** digunakan untuk menambah atau mengurangi variabel yang diset pada **inisialisasi**.

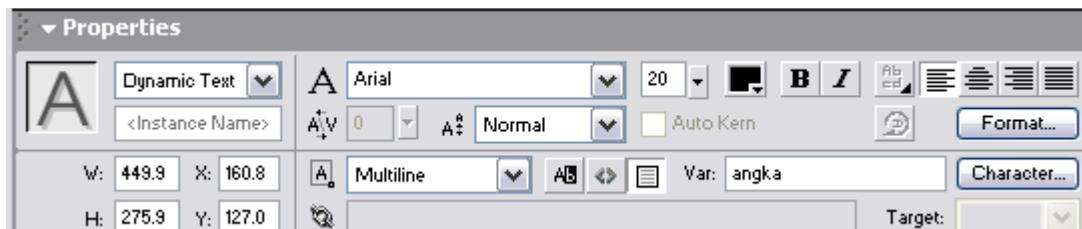
Untuk lebih jelasnya perhatikan contoh penggunaan **for** berikut :

Permasalahan :

Buatlah sebuah program yang menampilkan angka 1 sampai 100

Penyelesaian dari masalah tersebut terlihat sangat mudah, kita tinggal menuliskan angka dari 1 sampai 100 dengan menggunakan **text tool**. Akan tetapi hal tersebut tentunya memakan waktu, karena ada cara yang jauh lebih mudah, yaitu :

1. Buatlah sebuah **dynamic text** area dengan menggunakan **text tool**. Atur warna, ukuran huruf, bentuk huruf dan besar area sesuai dengan yang diperlukan.
2. Klik text area tersebut, kemudian bukalah panel properties. Pilih **multiline** pada line type dan ketikan “**angka**” pada var.



dynamic text properties

3. Kemudian buatlah sebuah layer baru diatas layer 1, dan ubah namanya menjadi **layer action**.
4. Klik frame 1 layer action, kemudian buka panel action dan ketikan action sebagai berikut :

```
for (i=1; i<=100; i++) {  
    angka += i+", ";  
}
```

5. Jalankan movie, maka akan muncul bilangan 1 sampai dengan 100.

Penjelasan program :

1. Sebuah penulisan script di frame memiliki 3 tipe penulisan yaitu **penulisan di dalam** suatu blok **movie event** atau **mouse event**, contoh **onEnterframe = function(){}** dan penulisan **tanpa movie event** atau **mouse event**, contoh adalah penggunaan for pada program yang kita buat di atas. Tipe penulisan yang lain adalah kombinasi dari keduanya, contohnya akan kita dapatkan pada program di bab selanjutnya.
2. baris **for (i=1; i<=100; i++)**{ berarti perintah dalam blok {} akan dijalankan secara terus menerus sebanyak 100 kali (nilai variabel i awal adalah 1 (**i = 1**) dan nilai i pada variabel kondisi adalah 100 (**i<=100**), setiap saat i ditambah dengan angka 1 (**i++**) sampai **i = 100**. Setelah variabel i bernilai 100, maka pengulangan dihentikan karena kondisi sudah tercapai.
3. baris **angka += i+", "**; berarti variabel **angka** (variabel yang ditampilkan oleh **dynamic text** yang kita buat sebelumnya) setiap saat (karena action pengulangan **for**) akan ditambah dengan number **i** dan string **","**. Perhatikan penjelasan berikut:  
saat nilai **i** bernilai 1 maka variabel **angka** bernilai : **"1,"**  
selanjutnya nilai **i** bernilai 2 akibat scrip **i++** maka variabel **angka** yang sudah bernilai **"1,"** ditambah lagi dengan **"2,"** sehingga menjadi **"1, 2,"**.  
proses tersebut diulang secara terus menerus oleh script **for** sampai **i** bernilai **100** sehingga pada akhir proses diakhiri dengan nilai **angka = "...,99, 100,** **"**.

## Movie Clip

Movieclip merupakan sebuah movie kecil yang berada didalam movie utama. Movieclip memiliki timeline sendiri, dan dapat ditampilkan dalam timeline movie utama dengan satu frame saja atau lebih.

Mode action pada flash sering kali melibatkan sebuah atau lebih movieclip. Oleh karena itu action script khusus untuk movieclip harus dipahami secara mendetail jika kita ingin menjadi seorang flash developer.

## Membuat Movieclip

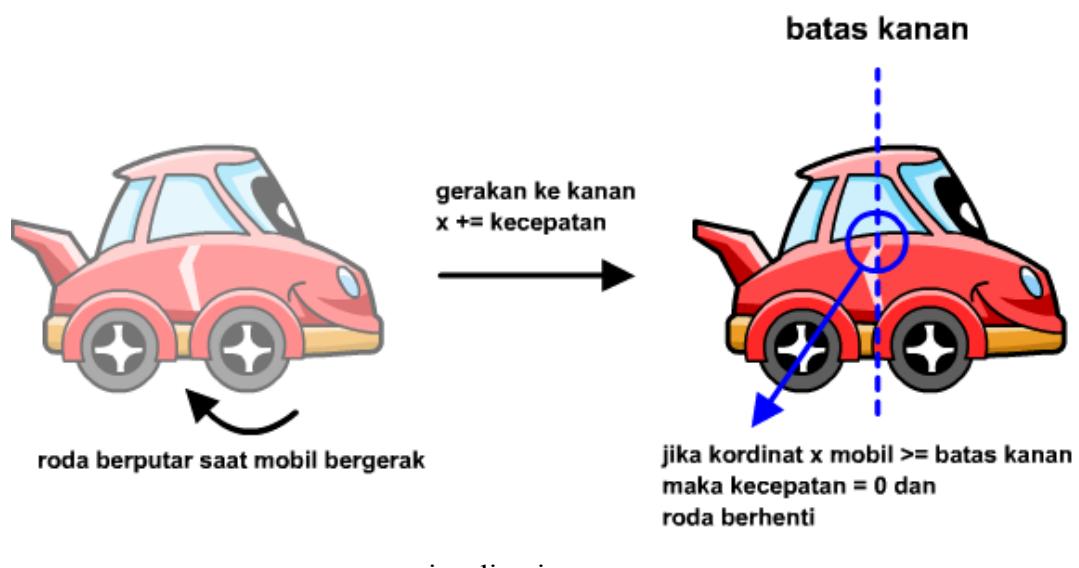
Setelah pelajaran sebelumnya, mungkin kita akan menganggap bahwa membuat movieclip itu mudah, yaitu dengan membuat sebuah obyek, kemudian mengconvertnya menjadi movieclip dengan menekan F8.

Dalam pemrograman game, tidaklah semudah itu. Kita harus memahami bagian yang lebih mendetail. Untuk lebih jelasnya perhatikan contoh sederhana berikut :

Permasalahan :

buatlah sebuah animasi mobil yang bergerak ke kanan dan berhenti saat berada di kanan layar.

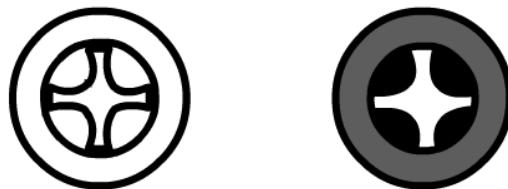
Untuk menyelesaikan permasalahan tersebut perhatikan gambar berikut :



masalah yang muncul pada pembuatan program tersebut adalah pada bagian menghentikan roda saat mobil berhenti. Sedangkan mengenai teknik menggambar, akan dijelaskan pada bab pembuatan karakter game.

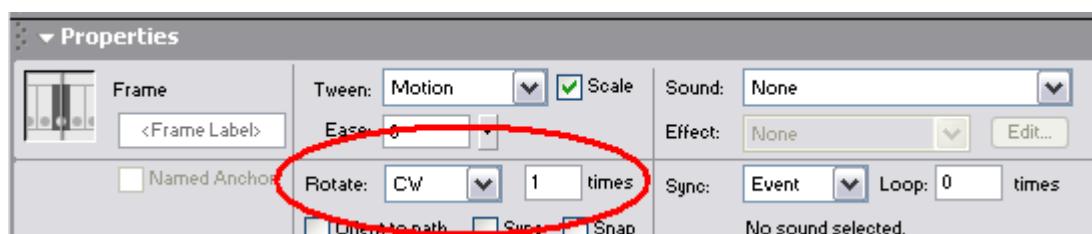
Perhatikan solusi atas permasalahan tersebut, berikut ini :

1. Buatlah file baru berukuran 800 x 600 pixel dan 12 fps.
2. Dengan menggunakan drawing tool buatlah sebuah gambar frame roda, dan warnailah sesuai dengan keinginan anda. Pastikan untuk menggambar velg walaupun sederhana, agar roda kelihatan berputar saat digerakkan.



gambar roda sebelum dan sesudah diberi warna

3. Seleksi gambar roda tersebut, kemudian convert menjadi symbol **movieclip** dengan nama **roda**.
4. Kemudian convert lagi movieclip **roda**, menjadi symbol **movieclip** dengan nama **animasi roda**. Maksud dari mengonvert obyek sebanyak dua kali agar ketika kita memasuki mode edit movieclip **animasi roda** kita tidak perlu mengonvert symbol lagi. ( selain itu mengonvert menjadi suatu symbol diperlukan apabila kita akan membuat sebuah **motion tween**)
5. Double klik movieclip **animasi roda** untuk mengeditnya.
6. Klik frame 15, kemudian masukan **keyframe**.
7. Selanjutnya klik kanan frame 10 dan pilih **create motion tween**.
8. Klik frame 10, kemudian buka properties. Pilih **CW** pada option **rotate**. Maka akan didapatkan sebuah animasi roda yang berputar searah jarum jam (clockwise).



Memutar obyek searah jarum jam dengan rotate

9. Jalankan movie untuk melihat hasil perputarannya. Jika kita jeli kita akan melihat bahwa perputaran roda sedikit janggal, karena ada roda berhenti beberapa saat.

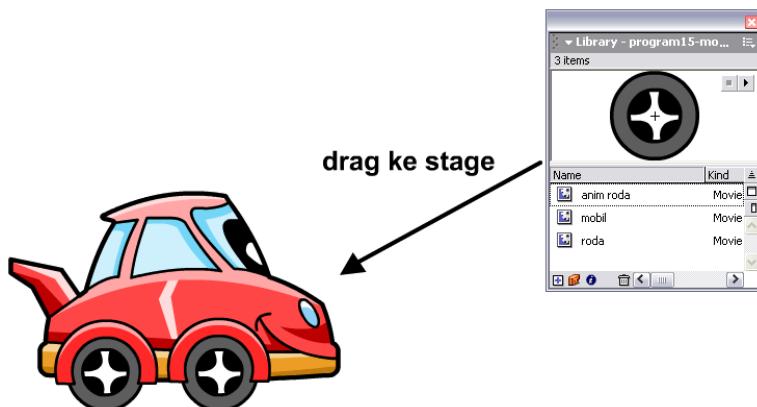
Hal tersebut dikarenakan adanya posisi roda yang sama pada frame 1 dan frame 15. Untuk memperbaikinya tekan Ctrl+W.

10. Pastikan terlebih dahulu anda tetap pada mode edit movieclip **animasi roda**. Kemudian tambahkan sebuah layer, dan ubah namanya menjadi **layer action**.
11. Klik layer action frame 15 **layer action**, kemudian masukan keyframe. Buka panel action dan ketikan action **gotoAndPlay(1);**
12. Jalankan kembali movie dan sekarang perputaran roda sudah tidak janggal. Tutup kembali dengan menekan Ctrl+W.
13. Seleksi movieclip **animasi roda** pada stage dan hapus dengan menekan tombol **Delete**. Sebuah obyek yang sudah di convert menjadi symbol secara otomatis akan masuk ke dalam **library**.
14. Buatlah sebuah gambar badan mobil menggunakan drawing tool. Kemudian konvert menjadi symbol **movieclip** dengan nama **mobil**.



gambar badan mobil pada movieclip **mobil**

15. Double klik movieclip **mobil** tersebut untuk memasuki mode edit symbol. Kemudian buat layer baru di atas layer 1 dan ubah namanya menjadi layer **roda**.
16. Buka **library** dengan menakan tombol **Ctrl+L** atau klik menu window>library, jika panel library belum terbuka. Kemudian **drag** movieclip **animasi roda** dari library ke stage sebanyak 2 kali. Atur posisinya.



drag symbol dari library ke stage

17. Setelah pembuatan movieclip **mobil** dianggap selesai, tekan Ctrl+E untuk kembali ke stage utama. Atur posisinya dengan meletakkan movieclip mobil di sebelah kiri stage.
18. Untuk menggerakan mobil ke kanan, klik movieclip **mobil**, buka panel action dan ketikan action berikut :

```
onClipEvent (load) {  
    kecepatan = 10;  
    bataskanan = 700;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan) {  
        kecepatan = 0;  
    }  
}
```

19. Simpan file dan jalankan movie, dan kita akan mendapati mobile bergerak ke kanan dan berhenti. tetapi roda mobil tersebut masih bergerak.

Penjelasan program tersebut sama dengan penjelasan **program latihan if**. Untuk menghentikan roda mobil terdapat pada latihan selanjutnya.

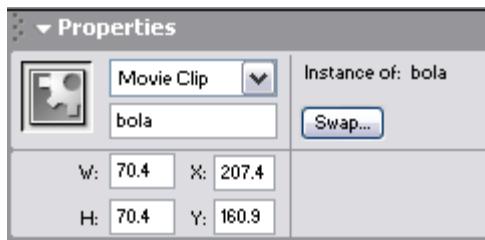
### **Instance Name**

Telah dijelaskan pada bab sebelumnya bahwa identitas sesungguhnya suatu symbol (movieclip atau button) adalah **instance name**. Kegunaan instance name adalah untuk memberi identitas pada suatu symbol, sehingga kita bisa lebih spesifik memberikan action pada movieclip atau button tertentu yang sudah diberi instance name.

### **Mengatur Movieclip dari Frame dengan Movie Event: onEnterFrame()**

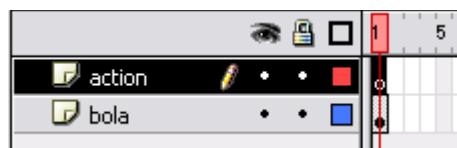
Kita dapat mengatur property movieclip, gerakan, menduplikasi atau menghapus sebuah movieclip dengan menuliskan action pada frame. Untuk lebih jelasnya perhatikan contoh berikut :

- Buatlah sebuah obyek lingkaran dengan menggunakan oval tool. Seleksi obyek lingkaran tersebut kemudian convert menjadi. Pilih **movieclip** pada option behaviour dan ketikan **bola** pada name.
- Klik symbol bola tersebut, kemudian buka panel instance dan ketikan nama “bola” pada instance name.



Properties movieclip bola

- Buatlah sebuah layer baru dan ubah namanya menjadi layer **action**. Penambahkan layer action akan mempermudah kita dalam penulisan action pada frame, sehingga tidak akan mengganggu obyek yang lain.



layer action

- Klik frame 1 layer **action**, dan dalam kondisi frame 1 terseleksi buka panel action

```
bola.onEnterFrame = function() {
    //menggerakan bola ke kanan
    this._x += 10;
};
```

- Jalankan movie, maka bola akan bergerak ke kanan.

Penjelasan pola penulisan action script pada frame adalah sebagai berikut :

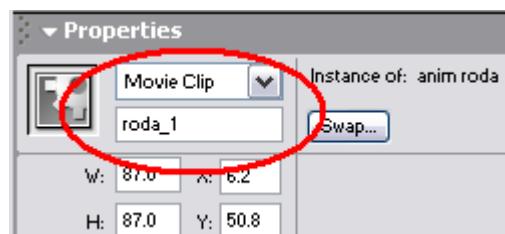
Sebelumnya lihatlah tabel perbedaan antara penulisan action script di frame dengan penulisan action script di movieclip pada bab sebelumnya. Hal yang harus dipegang adalah kata “**bola**” dalam baris “**bola.onEnterFrame**” adalah **instance name** bukan nama symbol. Selain itu dalam mode penulisan action di frame, kita membutuhkan action **this**. untuk mengawali sebuah object property atau obyek action.

## Instance Name Bertingkat

Dalam latihan menggerakkan mobil yang belum terselesaikan sebelumnya, terdapat sebuah movieclip yang bertingkat. Maksudnya didalam movieclip **mobil** terdapat movieclip lagi yaitu movieclip **animasi roda**. Dengan bahasa bebas dapat diterjemahkan sebagai berikut “**animasi roda** adalah anak dari **mobil** dan mobil adalah anak dari **movie utama**”. Dengan demikian kita bisa **menambahkan instance name yang bertingkat** pula pada movieclip **mobil** tersebut.

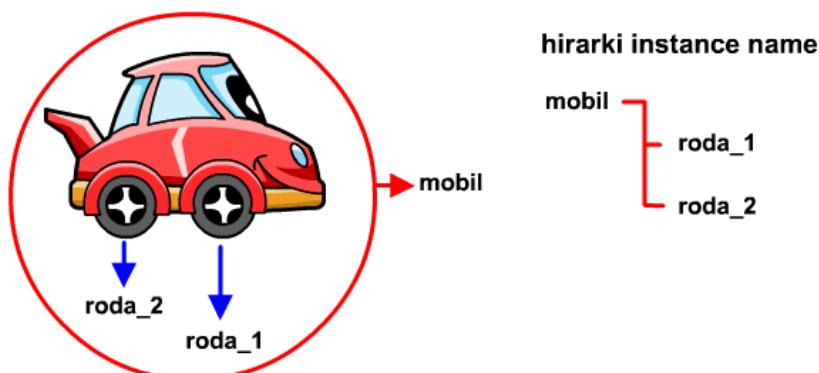
Untuk lebih jelasnya perhatikan contoh berikut :

1. Buka kembali file latihan menggerakan mobil yang anda buat sebelumnya (buka file program15-movieclip.fla).
2. Double klik movieclip **mobil** untuk mengeditnya.
3. Dalam mode edit movieclip **mobil**, klik movieclip **animasi roda** bagian depan, kemudian buka panel properties dan ketikan “**roda\_1**” pada instance name.



instance name animasi roda depan

4. Klik movieclip **animasi roda** bagian belakang, kemudian buka panel properties dan ketikan “**roda\_2**” pada instance name.
5. Kembali ke stage utama dengan menekan tombol Ctrl+E.
6. Klik movieclip **mobil** dan buka panel properties dan ketikan “**mobil**” pada instance name. Perhatikan hirarki instance name bertingkat berikut ini:



susunan hirarki instance name movieclip roda

7. Klik kembali movieclip **mobil** kemudian buka panel action dan tambahkan action menjadi :

```
onClipEvent (load) {  
    kecepatan = 10;  
    bataskanan = 700;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan) {  
        kecepatan = 0;  
        //menghentikan roda  
        roda_1.stop();  
        roda_2.stop();  
    }  
}
```

8. Jalankan movie, maka ketika mobil melewati batas kanan, mobil akan berhenti dan roda ikut berhenti.

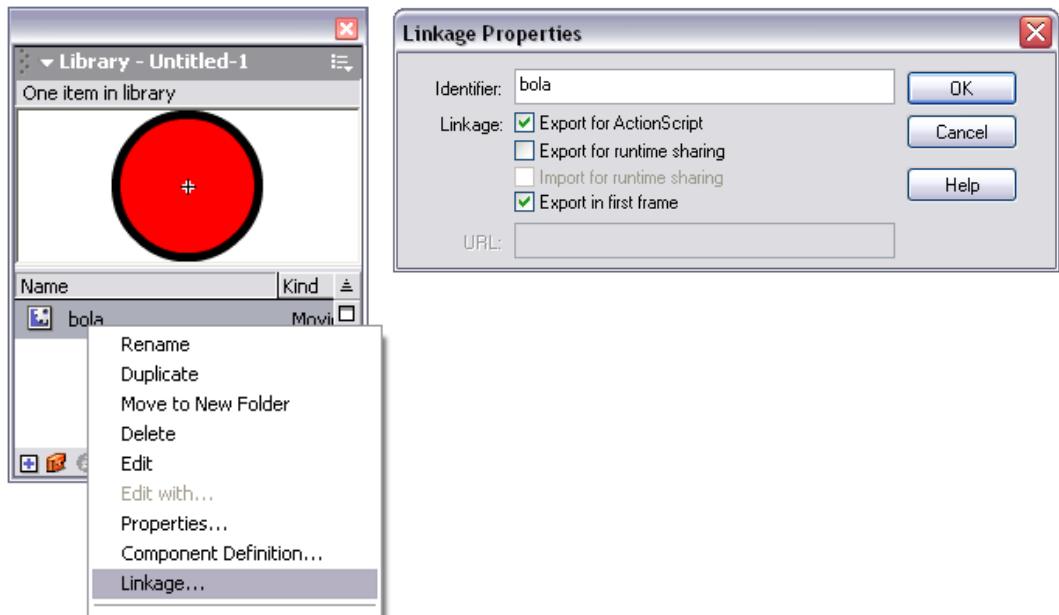
Penjelasan program :

1. Baris **roda\_1.stop()** berarti action **stop()** akan berimbang pada movieclip animasi roda yang sebelumnya telah diberikan instance name **roda\_1** saja.
2. Baris **roda\_2.stop()** memiliki penjelasan yang sama dengan penjelasan no 1.
3. Meskipun menggunakan movieclip yang sama yaitu movieclip **animasi roda**, instance name harus berbeda agar bisa dipengaruhi oleh action.
4. Apabila script tersebut ditulis pada frame, maka bentuk penulisannya menjadi **mobil.roda\_1.stop();** perhatikan bahwa dibagian depan diikuti dengan nama instansi induknya.

## Menggunakan Linkage

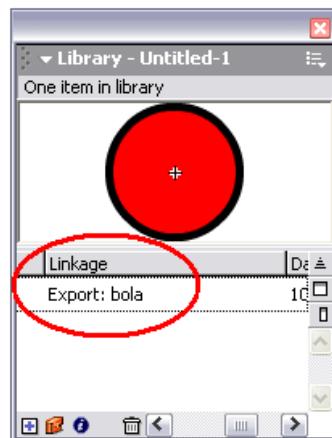
Selain instance name dalam flash juga terdapat istilah **linkage** sebagai identitas pengenal suatu symbol (movieclip, button dan sound). Untuk menambahkan linkage perhatikan contoh berikut :

1. buka panel library, kemudian klik kanan nama symbol dan pilih menu **linkage**, maka akan muncul **linkage properties**. Ketikan nama linkage pada **identifier**, aturan penulisan nama identifier sama dengan aturan penulisan instance name dan pilih option **export for action script**, secara otomatis option **export in first time** akan terpilih.



membuat linkage

2. Klik OK, maka pada library akan muncul nama linkage.



identifier linkage pada library

### **Menambahkan Movieclip ke dalam Stage dengan attachMovie()**

Menambahkan movieclip ke stage dapat dilakukan dengan dua cara, cara yang pertama adalah cara yang sudah kita lakukan pada latihan-latihan sebelumnya, yaitu

dengan menggambar langsung obyek pada stage, lalu mengconvertnya menjadi movieclip atau dengan mendrag suatu movieclip dari library ke stage.

Cara yang kedua adalah dengan menggunakan action **attachMovie()** bentuk perintah tersebut adalah sebagai berikut :

**attachMovie(identifier, instance name, level, property);**

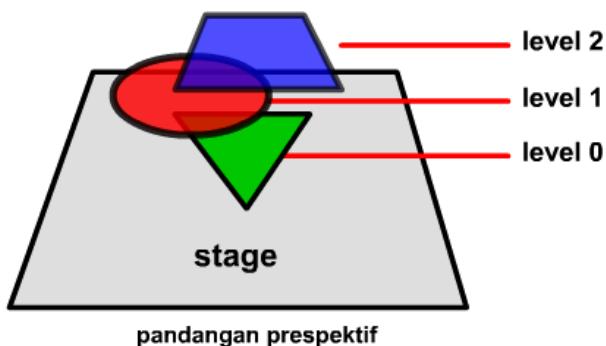
Bagian **identifier** merupakan identifier linkage suatu movieclip.

Bagian **instance name** merupakan instance name baru movieclip yang ditambahkan ke stage.

Bagian **level** merupakan kedalaman dimana movieclip tersebut diletakkan. Perhatikan aturan penulisan level berikut :

1. level dapat dimulai dari angka -9999 sampai +9999, dimana level -9999 adalah kedalaman yang paling bawah dan +9999 adalah kedalaman yang paling atas.

sistem level dalam attachmovie.



2. Satu level hanya dapat ditempati oleh satu obyek. Jadi kedalaman masing-masing movieclip harus berbeda.
3. Bila kita meletakkan movieclip dengan cara manual (tidak menggunakan action), secara otomatis level terbentuk mulai dari level 0.

Bagian **property** dapat ditulis atau tidak. Jika ditulis, bagian ini menyatakan properti dari movieclip yang ditambahkan.

Untuk lebih jelasnya tentang penggunaan action attachMovie, perhatikan contoh berikut :

1. Buatlah sebuah gambar lingkaran dan convert menjadi **movieclip** dengan nama **bola**.
2. Hapus **bola** dari stage. Kemudian buka panel library dengan menekan tombol Ctrl+L atau memilih menu window>library.

3. Pada library, klik kanan nama bola, kemudian pilih **linkage**. Pilih **option export for actionscrip**t dan klik **OK**. Perhatikan bahwa identifier secara default akan mengikuti nama symbol. Perhatikan gambar pada penjelasan mengenai linkage.
4. Klik frame 1, buka panel action. Kemudian ketikan action script berikut :

```
attachMovie("bola", "bola_1", 1, {_x:100, _y:100});
```

5. Jalankan movie, maka gambar **bola** akan muncul pada kordinat **100,100** pada stage yang sebelumnya kosong.

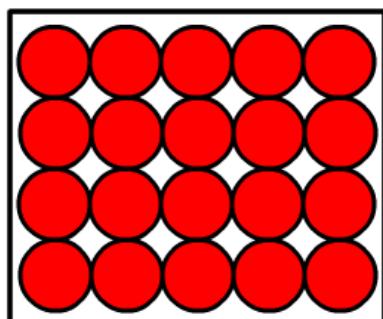
Penjelasan program :

1. Sebelum menggunakan action **attachMovie**, diharuskan terlebih dahulu untuk melinkage suatu symbol.
2. Menghapus movieclip **bola** dari stage dimaksudkan agar kita memulai program dengan stage yang kosong, sehingga penambahan movieclip oleh action **attachMovie** akan terlihat jelas.
3. Kata "**bola**" pada action **attachMovie("bola", "bola\_1", 1, {\_x:100, \_y:100});** merupakan nama identifier linkage.
4. Kata "**bola\_1**" pada action **attachMovie("bola", "bola\_1", 1, {\_x:100, \_y:100});** merupakan nama **instance name** movieclip yang ditambahkan.
5. Kata "**1**" merupakan level tempat kita meletakkan movieclip bola.
6. Kata **{\_x:100, \_y:100}** merupakan bagian pengesetan poperti awal movieclip bola, dimana kordinat x bernilai 100 dan kordinat y bernilai 100. Jika bagian ini tidak ditulis, kordinat movieclip secara default adalah **(0, 0)**.

Permasalahan :

Buatlah sebuah gambar lingkaran yang berderet memenuhi layar.

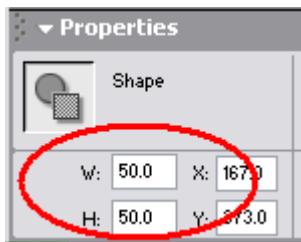
Sebelum menyelesaikan masalah tersebut, kita gambarkan dulu secara visual sebagai berikut :



## visualisasi program

Masalah tersebut dapat kita selesaikan dengan dua cara, yang pertama adalah secara manual kita tata satu demi satu movieclip ke stage, dan cara lain dengan menggunakan action script **for** sebagai berikut :

1. Buatlah sebuah gambar lingkaran dengan ukuran 50 x 50 pixel (gunakan properties untuk mengubah ukuran obyek) dan convert menjadi **movieclip** dengan nama **bola**.



mengeset ukuran obyek dengan properties

2. Hapus **bola** dari stage. Kemudian buka panel library dengan menekan tombol Ctrl+L atau memilih menu window>library.
3. Pada library, klik kanan nama bola, kemudian pilih **linkage**. Pilih **option export for actionscrip**t dan klik **OK**.
4. Klik frame 1, buka panel action. Kemudian ketikan action script berikut :

```
dalam = 0;  
for (var i = 0; i<17; i++) {  
    for (var j = 0; j<13; j++) {  
        attachMovie("bola", "bola_"+i+"_"+j, dalam++, {_x:(i*50),  
        _y:(j*50)});  
    }  
}
```

5. Jalankan movie, maka gambar **bola** akan muncul secara berdekatan memenuhi stage.

Penjelasan program :

1. baris **dalam = 0;** merupakan pengesetan awal variabel kedalaman movieclip.
2. baris **for (var i = 0; i<17; i++){** diikuti dengan baris **for (var j = 0; j<13; j++){** disebut sebagai operasi **pengulangan bertingkat**. Dalam operasi pengulangan

bertingkat, operasi pengulangan terakhir (**for (var j = 0; j<13; j++)**) akan dijalankan terlebih dahulu, sehingga perintah dalam blok {} ( yang berisi **attachMovie(...)**) akan dijalankan dengan nilai **i = 0** dan **j = 0** sampai **j<13** terlebih dahulu, barulah nilai **i** bertambah menjadi **i = 1**, dan nilai **j** menjadi 0 lagi, sehingga operasi pengulangan dilakukan kembali dengan nilai **i = 1** dan **j = 0** sampai **j<13**. Proses tersebut berulang-ulang sampai kondisi **i<17** terpenuhi.

3. Dalam operasi bertingkat diperlukan action **var** sebagai penyimpan variabel.
4. Baris **attachMovie("bola", "bola\_"+i+"\_"+j, dalam++, {\_x:(i\*50), \_y:(j\*50)});** berarti movieclip dengan lingkage “**bola**” pada library ditambahkan ke stage sebanyak 18 x 14 buah (oleh perintah **for** bertingkat), dalam penambahan tersebut masing masing movieclip diberikan nama instansi yang berbeda beda yaitu sesuai dengan nomor penempatannya (“**bola\_"+i+"\_"+j**”). Sebagai contoh, bola yang berada pada kolom ke **12** dan baris ke **5** akan memiliki instansi name “**bola\_12\_5**”. Kata “**dalam++**” menunjukkan lokasi kedalaman movieclip bola, “**++**” berarti setiap proses pengulangan nilai variabel **dalam** ditambah 1. Kata “**{\_x:(i\*50), \_y:(j\*50)}**” berarti pengaturan posisi kordinat masing-masing bola yang ditempatkan. Contoh bola yang berada pada kolom 12 dan baris ke 5 akan ditempatkan pada kordinat **(12 x 50, 5 x 50)** atau **(600, 250)**. Angka 50 sebagai pengali, diperoleh dari lebar dan tinggi dari movieclip bola yang kita tentukan sebelumnya yaitu **50**.

### **Menduplikasi Movieclip dengan duplicateMovieClip()**

Menduplikasi movieclip dapat dilakukan dengan 2 cara yaitu secara manual dengan mengcopy movieclip dan kemudian mempastenya, dan dengan menggunakan action script **duplicateMovieClip()**. Hal utama yang harus diperhatikan adalah setiap duplikasi harus memiliki **instance name** yang berbeda.

Menduplikasi sebuah movieclip sering diperlukan dalam suatu pemrograman game. Seperti memperbanyak musuh, membuat background, menampilkan senjata yang beragam dan sebagainya.

Bentuk penulisan action **duplicateMovieClip** ada 2 yaitu :

**duplicateMovieClip(target, nama instansi baru, kedalaman);** dan

**nama instansi.duplicateMovieClip(nama instansi baru, kedalaman, properti);**

Contoh program berikut ini adalah aplikasi action duplicateMovieClip mode kedua, sedangkan mode penulisan pertama akan dipakai pada program aplikasi pada bab selanjutnya :

1. Buatlah sebuah gambar lingkaran dan convert menjadi **movieclip** dengan nama **bola**.
2. Hapus **bola** dari stage. Kemudian buka panel library dengan menekan tombol Ctrl+L atau memilih menu window>library.
3. Pada library, klik kanan nama bola, kemudian pilih **linkage**. Pilih **option export for actionscript** dan klik **OK**.
4. Klik frame 1, buka panel action. Kemudian ketikan action script berikut :

```
attachMovie("bola", "bola_1", 1, {_x:100, _y:100});  
bola_1.duplicateMovieClip("bola_2", 2, {_x:200, _y:100});
```

5. Jalankan movie, maka gambar **bola** akan muncul pada kordinat **100,100** dan **200,100** pada stage yang sebelumnya kosong.

Penjelasan program :

1. baris pertama merupakan perintah untuk menambahkan movieclip bola ke stage dengan nama instance “**bola\_1**”.
2. Selanjutnya pada baris kedua, movieclip **bola\_1** diduplikasi menjadi movieclip berinstansi name “**bola\_2**”, diletakkan pada kedalaman **2** dan diletakkan pada kordinat (**200, 100**).

### **Menghapus Movieclip dengan removeMovieClip()**

Setelah tidak dipakai lagi, dalam sebuah game atau aplikasi, sebuah movieclip lebih baik dihapus dari stage. Hal tersebut dimaksudkan untuk memperkecil memori dan membersihkan layar tentunya.

Sama dengan action duplicateMovieclip, action removeMovieclip juga memiliki dua bentuk yaitu :

**removeMovieClip(target);** dan  
**nama instansi.removeMovieClip();**

Contoh program berikut ini adalah aplikasi action removeMovieClip mode kedua, sedangkan mode penulisan pertama akan dipakai pada program aplikasi pada bab selanjutnya :

1. Buatlah sebuah gambar lingkaran dan convert menjadi **movieclip** dengan nama **bola**.
2. Hapus **bola** dari stage. Kemudian buka panel library dengan menekan tombol Ctrl+L atau memilih menu window>library.
3. Pada library, klik kanan nama bola, kemudian pilih **linkage**. Pilih **option export for actionscript** dan klik **OK**.
4. Buatlah sebuah layer baru diatas layer 1 dan ubah namanya menjadi **layer action**. Klik frame 1 layer action, buka panel action. Kemudian ketikan action script berikut :

```
attachMovie("bola", "bola_1", 1, {_x:100, _y:100});  
bola_1.duplicateMovieClip("bola_2", 2, {_x:200, _y:100});
```

5. Klik frame 1 **layer 1**. Buatlah sebuah tombol **hapus**, kemudian klik tombol **hapus**, buka panel action dan ketikan script berikut :

```
on (release) {  
    bola_2.removeMovieClip();  
}
```

6. Jalankan movie, tekan tombol **hapus** maka **bola\_2** hasil duplikasi dari **bola\_1** akan hilang akibat action **bola\_2.removeMovieClip()**;

Catatan : apabila kita menambahkan movieclip secara manual (dengan mendrag dari library ke stage), action **removeMovieClip** yang dipakai adalah bentuk pertama. Penjelasan mengenai hal tersebut akan dibahas pada bab selanjutnya.

## Function

Function selalu dihadirkan dalam setiap bahasa pemrograman. Yang dimaksud dengan function adalah kumpulan dari satu atau lebih suatu action script yang dapat dipakai menjadi sebuah perintah baru yang kita definisikan sendiri.

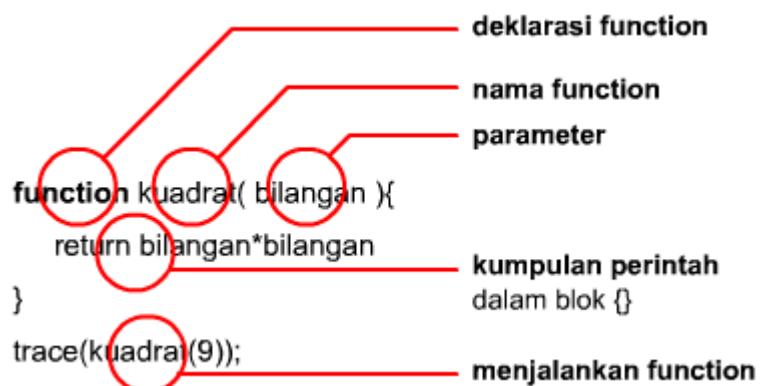
Perhatikan contoh berikut :

1. Buatlah file baru, kemudian klik frame 1, buka panel action dan ketikan script berikut :

```
function kuadrat(bilangan){  
    return bilangan*bilangan  
}  
  
trace(kuadrat(9));
```

2. Jalankan Movie maka pada panel output akan muncul angka **81**.

Struktur dari function adalah sebagai berikut :



Pertama kita harus memberikan sebuah **nama** pada fungsi yang akan kita buat. Aturan pemberian nama function sama dengan aturan pemberian nama instansi. Selanjutnya kita tentukan **parameter** atau **variabel** yang akan kita pakai dalam fungsi tersebut.

Di dalam blok “{}” merupakan kumpulan perintah, yang digunakan untuk menyusun sebuah function. Dalam contoh di atas, digunakan perintah **return bilangan\*bilangan** perintah **return** dalam hal ini adalah untuk menghasilkan suatu nilai.

Penggunaan function dilakukan di luar blok “{}” function itu sendiri, dan penulisannya dapat dilakukan di atas function maupun di bawah function. Perhatikan gambar berikut :

```
trace(kuadrat(9));          function kuadrat(bilangan) {  
function kuadrat(bilangan) {      return bilangan*bilangan;  
return bilangan*bilangan;      }  
}  
trace(kuadrat(9));
```

penggunaan function

Pada contoh diatas baris **trace(kuadrat(9));** berarti menampilkan pesan pada output panel **kuadrat** dari **9**. Contoh penggunaan yang lain adalah **total = kuadrat(17);** maka dapat diartikan sebagai **total = 289;** . Function dapat dipakai berulang kali.

## Array

**Array** adalah suatu kumpulan data yang bertipe sama. Contoh **array** nama-nama hari berisi (“minggu”, “senin”, “selasa”, “rabu”, “kamis”, “jumat”, “sabtu”). Dalam flash untuk membuat sebuah array terdapat 2 cara yaitu

**variabel = new Array();**

Action **new** dalam flash merupakan sebuah perintah untuk membuat obyek baru.

Karena **new** diikuti oleh **Array()**, berarti kita membuat obyek baru bertipe array.

Contoh aplikasi penggunaan action **newArray()**; adalah sebagai berikut :

Permasalahan :

Setelah game berakhir, terdapat 10 nilai pemain yang terekam yaitu : 100, 98, 105, 72, 31, 67, 280, 210, 67, 56. Buatlah sebuah aplikasi yang dapat menampilkan highscore secara berurutan (dari score tertinggi sampai score terendah).

Penyelesaian masalah tersebut dengan menggunakan array adalah sebagai berikut :

1. Buatlah file baru, kemudian klik frame 1. Buka panel action dan ketikan action sebagai berikut :

```
score = new Array(100, 98, 105, 72, 31, 67, 280, 210, 67, 56);
//mencari yang tertinggi
banyakData = score.length;
for (i=0; i<=banyakData-1; i++) {
    dataTerkecil = i;
    for (j=i+1; j<banyakData; j++) {
        if (score[j]>score[dataTerkecil]) {
            dataTerkecil = j;
        }
    }
    if (dataTerkecil>i) {
        penyimpanSementara = score[i];
        score[i] = score[dataTerkecil];
        score[dataTerkecil] = penyimpanSementara;
    }
}
```

```
    }  
}  
// menampilkan data  
trace(score);
```

2. Jalankan movie, maka pada output panel akan muncul angka yang berurutan mulai dari yang tertinggi sampai terendah.

Penjelasan program :

1. baris **score = new Array(100, 98, 105, 72, 31, 67, 280, 210, 67, 56);** merupakan pengesetan variabel bertipe array dan pengesetan isi datanya. Berarti nilai dari **score[5] = 67;** karena penghitungan data dimulai dari angka 0.
2. baris **banyakData = score.length;** **length** merupakan action yang digunakan untuk mengetahui jumlah data dalam **array** dan disimpan pada variabel **banyakData.**
3. baris ke 4 sampai baris ke 16 merupakan sebuah prosedure yang mengurutkan nilai variabel **score** dari yang tertinggi sampai yang terendah. Prinsip dasar program pengurutan data adalah sebagai berikut :
  1. Simpan bilangan pertama (**for (i=0; i<=banyakData-1; i++) {}**)
  2. Ambil bilangan kedua (**for (j=i+1; j<banyakData; j++) {}**)
  3. Bandingkan antara bilangan pertama dan bilangan kedua. (**if (score[j]>score[dataTerkecil]) {}**).
  4. Jika bilangan kedua lebih besar, maka balik *posisi data*-nya (**dataTerkecil = j;**).
  5. Kemudian balik *nilai datanya*. Sebelum membalik nilai data, kita membutuhkan variabel penyimpan data pertama (**penyimpanSementara = score[i];**)
  6. Selanjutnya nilai data pertama di ubah menjadi = nilai data kedua (**score[i] = score[dataTerkecil];**).
  7. Sedangkan nilai data kedua di ubah menjadi = nilai data pertama, akan tetapi berhubung nilai data pertama telah berubah menjadi nilai data kedua akibat action (**score[i] = score[dataTerkecil];**), maka digunakanlah variabel **penyimpanSementara** yang masih memiliki nilai awal dari data pertama dengan menggunakan action (**score[dataTerkecil] = penyimpanSementara;**)

4. Selanjutnya data yang sudah diurutkan ditampilkan pada output panel dengan action **trace(score);**

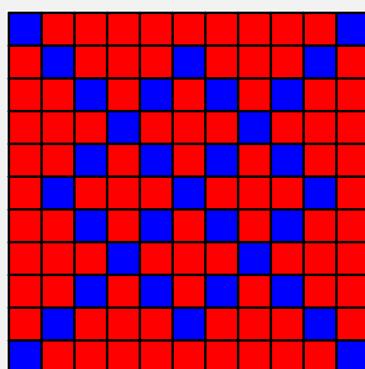
**variabel = [ ];**

Cara membuat sebuah variabel bertipe array selanjutnya adalah dengan menuliskan bentuk **variabel = [ ]**. Cara tersebut sama hasilnya dengan cara yang pertama.

Contoh aplikasi penggunaan action **variabel = [ ];** adalah sebagai berikut :

Permasalahan :

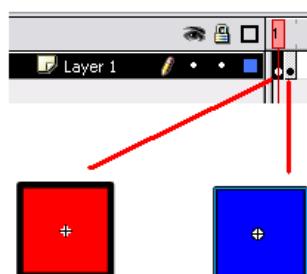
Buatlah sebuah gambar berpola sebagai berikut:



Penyelesaian dari masalah tersebut adalah dengan menggunakan **array berdimensi 2**.

Perhatikan proses berikut :

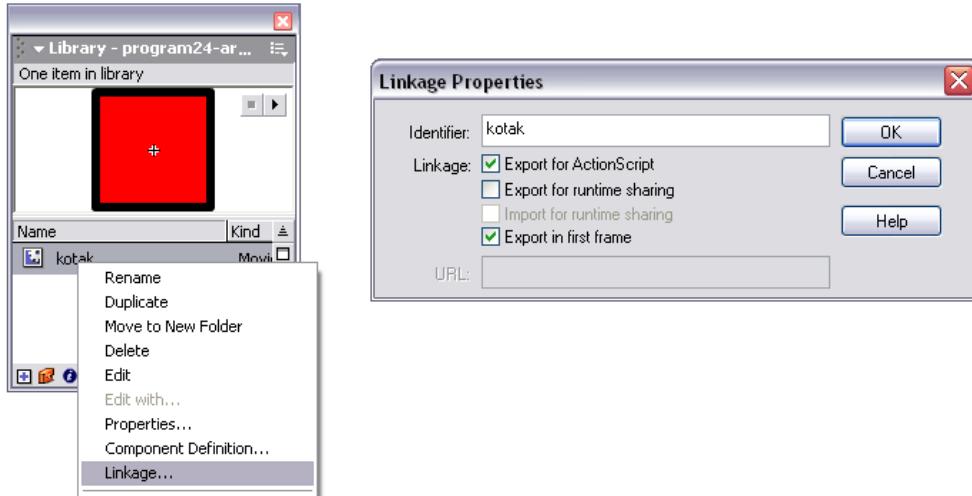
1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps;
2. Buatlah gambar kotak berwarna merah dengan menggunakan rectangle tool, dengan ukuran 40 x 40 pixel. Selanjutnya seleksi kotak tersebut dan convert menjadi **movieclip** dengan nama **kotak**.
3. Double klik movieclip **kotak** untuk masuk ke mode edit symbol. Pada mode edit symbol, klik frame 2 dan masukan keyframe.
4. Ubah warna kotak dari merah menjadi biru. Sehingga pada frame 1 kotak berwarna merah dan pada frame 2 kotak berwarna biru. Selanjutnya Tekan Ctrl+E untuk kembali ke stage utama.



**frame 1      frame 2**

susunan obyek pada movieclip **kotak**

5. Hapus movieclip **kotak** dari stage. Buka panel library, kemudian klik kanan pada nama symbol dan pilih **linkage**. Pilih **export for action script** dan klik **OK**.



melinkage movieclip **kotak**

6. Klik frame 1, buka panel action dan ketikkan scrip berikut :

```
pola = [[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],  
        [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],  
        [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],  
        [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],  
        [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],  
        [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],  
        [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],  
        [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],  
        [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],  
        [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],  
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 1]];  
  
function gambarPola(dataPola, xawal, yawal) {  
    kedalaman = 0;  
    lebar = dataPola[0].length;  
    tinggi = dataPola.length;  
    for (var i = 0; i<lebar; i++) {  
        for (var j = 0; j<tinggi; j++) {  
            attachMovie("kotak", "kotak"+i+"_"+j, kedalaman++,  
                       {_x : xawal+(i*40), _y : yawal+(j*40)});  
    }  
}
```

```

        this["kotak"+i+"_"++j].gotoAndStop(dataPola[j][i]+1);
    }
}

gambarPola(pola, 100, 100);

```

- Jalankan movie, maka akan didapatkan gambar yang diinginkan.

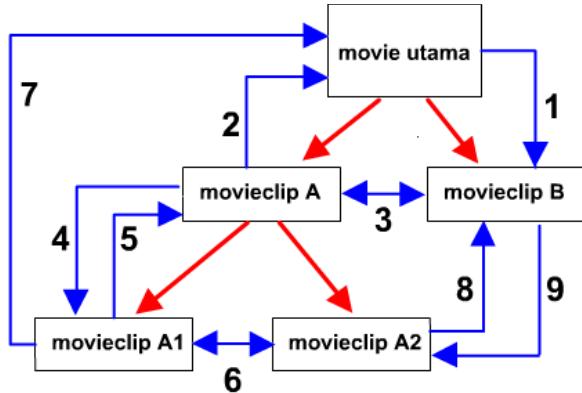
Penjelasan program :

- Baris 1 sampai baris 11 merupakan baris tempat kita membuat variabel **pola** bertipe **array** berdimensi 2 sebesar 11 x 11 data. Secara mudah dapat kita pahami sebagai berikut : bila penulisan array diawali dengan satu tanda [ berarti array tersebut berdimensi 1, contoh penggunaannya adalah **score[2] =105**. Sedangkan apabila diawali dengan dua tanda [, berarti array tersebut berdimensi 2. Contoh penggunaannya **pola[4][2] = 1**.
- Baris ke 12 (**function gambarPola(dataPola, xawal, yawal){ }**) merupakan sebuah fungsi untuk membuat gambar pola dengan variabel **dataPola** sebagai informasi bentuk pola, **xawal** sebagai kordinat x dimulainya proses menggambar, dan **yawal** sebagai kordinat y dimulainya proses menggambar.
- Baris **kedalaman = 0** merupakan pengesetan awal variabel yang akan digunakan untuk menentukan kedalaman (**level**) dari movieclip yang akan dibuat.
- Baris **lebar = dataPola[0].length;** digunakan untuk menghitung banyaknya movieclip yang akan digambar ke arah samping. Perhatikan bahwa **dataPola[0]** bernilai = **[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]**, sehingga panjangnya adalah 11.
- Baris **tinggi = dataPola.length;** digunakan untuk menghitung banyaknya movieclip yang akan digambar ke arah bawah.
- Baris **for (var i = 0; i<lebar; i++) {** dan **for (var j = 0; j<tinggi; j++){** merupakan operator pengulangan bertingkat untuk menggambar sesuai dengan pola.
- Baris **attachMovie("kotak", "kotak"+i+"\_"++j, kedalaman++,{\_x : xawal+(i\*40), \_y : yawal+(j\*40)})**; akan menambahkan movieclip **kotak** ke stage sebanyak 11 kali ke arah samping dan 11 kali ke bawah, dengan level kedalaman yang selalu ditambah 1 (**kedalaman++**), dan peletakan movieclip dimulai dari kordinat **xawal** dan **yawal**. Sedangkan bilangan **40** diperoleh dari ukuran panjang dan lebar movieclip kotak.

8. Baris **this["kotak"+i+"\_"++j].gotoAndStop(dataPola[j][i]+1);** merupakan baris yang membuat pola terbentuk dalam dua warna. Ingat bahwa movieclip kotak memiliki 2 frame, yaitu frame 1 berisi kotak berwarna merah dan pada frame 2 berisi kotak berwarna biru. Action **this["kotak"+i+"\_"++j]** akan berpengaruh pada kotak dengan instance name "**kotak"+i+"\_"++j"**, dan action **gotoAndStop(dataPola[j][i]+1);** menyebabkan movieclip dengan instance name "**kotak"+i+"\_"++j"** akan aktif pada nomer frame sesuai dengan **dataPola[j][i]**, karena variabel **dataPola[j][i]** bernilai antara **0** dan **1**, sedangkan frame movieclip kotak bernilai **1** dan **2** maka variabel **dataPola[j][i]** perlu ditambah **1 (+1)**. Contoh penerapan baris tersebut : suatu ketika nilai **i = 6** dan **j = 2**. Maka instance name yang terbentuk adalah "**kotak6\_2**", dan nilai dari **dataPola[2][6] = 1**. Akibat penambahan 1 pada variabel **dataPola** maka keseluruhan baris action tersebut menjadi : **kotak6\_2.gotoAndStop(2);** dan kotak menjadi berwarna biru.
  
9. baris **gambarPola(pola, 100, 100);** merupakan eksekusi dari function **gambarPola**, sehingga gambarpun terjadi sesuai dengan yang diingikan.

## \_root, \_parent, dan this

\_root. dalam terjemahan bebas dapat kita artikan sebagai “akar”, \_parent dapat diartikan sebagai “induk” dan this dapat diartikan dengan “symbol ini”. Perhatikan diagram berikut untuk lebih memahami 3 bentuk action tersebut:



penggunaan action \_root, this dan \_parent

Penjelasan dari diagram tersebut adalah :

Sebagai contoh : sebuah **movie** memiliki 2 buah movieclip di dalamnya yaitu **movieclip A** dan **movieclip B**. Di dalam **Movieclip A** terdapat 2 buah movieclip dengan instance name yang berbeda yaitu : **movieclip A1** dan **movieclip A2**. Kita dapat menganggap movie utama sebagai pohon, kemudian movieclip A dan movieclip B sebagai akarnya. Movieclip A1 dan movieclip A2 dapat kita anggap sebagai anak dari movieclip A.

Berikut adalah penjelasan masing-masing nomor :

1. Untuk mengakses suatu **movieclip dari movie utama (no 1)** kita gunakan bentuk **instance name.action/property**; contoh: pada **frame 1** kita tuliskan action : **nama instance movieclip B.\_x = 100**; perintah tersebut akan berpengaruh pada movieclip B saja, dan akan mengeset kordinat \_x movieclip B menjadi 100.
2. Untuk mengakses **movie utama dari movieclip (no 2)** kita gunakan bentuk \_root.action/property; atau \_parent.gotoAndStop(20) contoh : di **frame 1** pada **movieclip A** kita tuliskan action di dalam suatu blok movie event : \_root.gotoAndStop(20); perintah tersebut akan menyebabkan frame aktif dari movie utama berpindah dari 1 ke 20.
3. Untuk mengakses **movieclip satu dari movieclip yang lain (no 3)** , maka digunakan bentuk \_root.nama instansi movieklip yang akan

**diakses.action/property;**. Contoh : pada **movieclip A** kita ketikan action di dalam suatu blok movie event : **\_root.nama instance movieclip B.\_visible = 0;** perintah tersebut akan menghilangkan movieclip B dari stage.

4. Untuk mengakses **movieclip yang berada dalam suatu movieclip / mengakses movieclip anak dari movieclip induk (no 4)**, maka digunakan bentuk **this.nama instansi movieclip yang berada di dalamnya.action/property;** Contoh : pada **movieclip A** kita ketikan action di dalam suatu blok movie event : **this.nama instance movieclip A1.gotoAndStop(frame);** perintah tersebut akan membuat **movieclip A1** yang berada di dalam **movieclip A** berhenti pada frame tertentu.
5. Untuk mengakses **movieclip induk dari movieclip anak (no 5)**, digunakan bentuk **\_parent.action /property; atau \_root.instance name movieclip induk.action/property;** Contoh : di dalam **movieclip A1** kita ketikan action di dalam suatu blok movie event : **\_parent.\_xscale = 50;** perintah tersebut akan menyebabkan **movieclip A** berubah bentuk secara horisontal menjadi setengah (50%) dari ukuran aslinya.
6. Untuk mengakses **movieclip anak dari movieclip anak yang lain(no 6)**, digunakan bentuk **\_parent.instance name anak yang akan diakses.action/property; atau \_root.instance name movieclip induk.instance name moieclip anak yang akan diakses.action/property;** contoh : di dalam **movieclip A1** kita ketikan action di dalam suatu blok movie event : **\_parent.nama instansi movieclip A2.stop();** perintah tersebut akan menyebabkan **movieclip A2** berhenti pada frame tertentu.
7. Untuk mengakses **movie utama dari movieclip anak (no 7)**, digunakan bentuk **\_root.action/property; atau \_parent.\_parent.action/property;** contoh : di dalam **movieclip A1** kita ketikan action di dalam suatu blok movie event : **\_parent.\_parent.gotoAndStop("gameover");** perintah tersebut akan menyebabkan timeline movie utama aktif pada frame berlabel "gameover".
8. Untuk mengakses **movieclip lain dari movieclip anak (no 8)** maka digunakan bentuk **\_root.nama instansi movieclip yang akan diakses.action/property;** contoh : di dalam **movieclip A2** kita ketikan action di dalam suatu blok movie event : **\_root.nama instance movieclip B.\_y += 200;** perintah tersebut akan menyebabkan **movieclip B** bergeser kearah bawah sebanyak 200 pixel.
9. Untuk mengakses **movieclip anak dari movieclip lain (no 9)** maka digunakan bentuk **\_root.nama instansi movieclip induk.nama instanse movieklip anak**

**yang akan diakses.action/property;** contoh : di dalam **movieclip B** kita ketikan action di dalam suatu blok movie event : **\_root.nama instance movieclip A.nama instance movieclip A2.\_alpha = 50;** perintah tersebut akan menyebabkan **movieclip A2** menjadi semi transparan.

### **\_root.**

Ada 3 bentuk dasar **\_root** yaitu :

### **\_root.action**

Pada bentuk ini dibelakang **\_root**. diikuti dengan action script lain seperti gotoAndStop(); stop(); atau action yang lain. Bentuk ini pada dipakai untuk mengakses movie utama. Perhatikan contoh berikut. :

1. Sebelumnya buka file latihan menggerakkan movieclip mobil pada latihan sebelumnya (pada CD tutorial buka file program15-movieclip.fla). Kemudian buka panel library.
2. Buatlah sebuah file baru dengan ukuran 800x600 pixel dan 12 fps.
3. Klik **frame 2**, kemudian masukan **keyframe**.
4. Pada **frame 2**, buatlah sebuah **static text** dengan kata “**finish**”.



tampilan pada frame 2

5. Klik **frame 1**, selanjutnya drag movieclip **mobil** dari library file latihan anda -(file tutorial program15-movieclip.fla) yang sudah anda buka sebelumnya- ke stage, dan letakkan disebelah kiri stage.
6. Klik **mobil** dan ketikan action script berikut :

```
onClipEvent (load) {  
    kecepatan = 10;  
    bataskanan = 700;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan) {  
        kecepatan = 0;  
        _root.gotoAndStop(2);  
    }  
}
```

```
    }  
}
```

7. Selanjutnya, buatlah sebuah layer baru diatas layer 1, dan ubah namanya menjadi **layer action**.
8. Klik **frame 1 layer action**, buka panel action dan ketikan action :  

```
stop();
```
9. Simpan file dan jalankan movie.

Penjelasan program :

1. Action **stop()**; diberikan pada frame 1 **layer action** agar movie tidak berjalan secara looping (berulang).
2. Ketika mobil telah sampai di garis batas kanan, maka frame aktif dari timeline movie utama dipindah ke frame 2 dengan action **\_root.gotoAndStop(2);**  
Perhatikan bahwa script tersebut ditulis pada movieclip **mobil**.

### **\_root.variable / property**

Pada bentuk **\_root** yang kedua, dibelakang **\_root**. diisi dengan **variable**, seperti variable score, waktu dan sebagainya. Perhatikan contoh berikut :

1. Lanjutkan file latihan sebelumnya. Kemudian pilih menu **file>save as** dan masukan nama baru. Maksud dari langkah ini adalah agar file proses kita tidak hilang tertindih oleh file yang baru. Dalam proses kerja hal tersebut sering diistilahkan *sebagai work in progress*.
2. Klik **frame 1 layer action**, kemudian tambahkan actionnya menjadi :

```
stop();  
speed = 15; // mengatur kecepatan mobil dari frame
```

3. Selanjutnya klik **mobil** dan ubah scriptnya menjadi :

```
onClipEvent (load) {  
    kecepatan = _root.speed; // mengakses variabel speed pada frame 1 layer action  
    bataskanan = 700;  
}  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan) {  
        kecepatan = 0;
```

```
    _root.gotoAndStop(2);
}
}
```

4. Simpan file dan jalankan movie.

Penjelasan program :

1. Action **speed = 15**; diambahkan pada frame 1 **layer action** agar kita memiliki variabel baru yang akan digunakan untuk mengatur kecepatan **mobil**.
2. Selanjutnya untuk mengakses variabel tersebut dari movieclip, digunakan action **kecepatan = \_root.speed**; sehingga dapat diartikan dengan **kecepatan = 15**;

### **\_root.movieclip**

Bentuk **\_root** yang ketiga tersebut bedakan lagi menjadi 2 yaitu:

### **\_root.instance movieclip.variable/property**

Bentuk tersebut pada umumnya digunakan untuk mengakses variabel yang sudah diset dari movieclip lain. Perhatikan contoh berikut :

1. Sebelumnya buka file latihan menggerakkan movieclip mobil pada latihan sebelumnya (pada CD tutorial buka file program15-movieclip.fla). Kemudian buka panel library.
2. Buatlah sebuah file baru dengan ukuran 800x600 pixel dan 12 fps.
3. Buatlah sebuah **static text** dengan kata “**finish**”, kemudian convert text tersebut menjadi **movieclip** dengan nama **finish**.



movieclip **finish**

4. Klik movieclip **finish**, buka panel properties dan ketikan kata “**finish**” pada instance name, kemudian buka panel action dan ketikan script berikut :

```
onClipEvent (load) {
    _visible = 0;
}
```

5. Selanjutnya drag movieclip **mobil** dari library file latihan anda -(file tutorial program15-movieclip.fla) yang sudah anda buka sebelumnya- ke stage, dan letakkan disebelah kiri stage. Perhatikan peletakkannya :



peletakan obyek pada stage

6. Klik **mobil** dan ketikan action script berikut :

```
onClipEvent (load) {  
    kecepatan = 10;  
    bataskanan = 700;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan) {  
        kecepatan = 0;  
        _root.finish._visible = 1;  
    }  
}
```

7. Jalankan movie.

Penjelasan program :

1. Movieclip **finish** pada awalnya diberi sebuah **instance name** agar bisa diakses oleh movieclip lain.
2. Agar tidak tampak saat pertamakali movie dimainkan, maka property **\_visible** pada movieclip **finish** diset menjadi **0**.

- Ketika mobil telah sampai di garis batas kanan, maka movieclip finish ditampilkan dengan script `_root.finish._visible = 1;` Perhatikan bahwa script tersebut ditulis pada movieclip **mobil** dan dapat mengakses property milik movieclip **finish**.

### **\_root.instance movieclip.action**

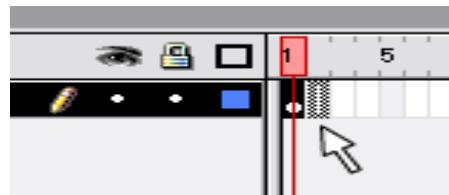
Bentuk ini digunakan untuk mengakses suatu movieclip dari movieclip lain .

Perhatikan contoh berikut :

- Sebelumnya buka file latihan menggerakkan movieclip mobil pada latihan sebelumnya (pada CD tutorial buka file program15-movieclip.fla). Kemudian buka panel library.
- Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
- Buatlah sebuah **static text** dengan kata “**finish**”, kemudian convert text tersebut menjadi **movieclip** dengan nama **teksfinish**.



- Kemudian seleksi **teksfinish**, kemudian convert lagi menjadi **movieclip** dengan nama **finish**.
- Double klik movieclip **finish**. Dalam mode edit symbol, klik **frame 1**, kemudian **drag** dan geser ke **frame 2**, sehingga frame 1 menjadi kosong.

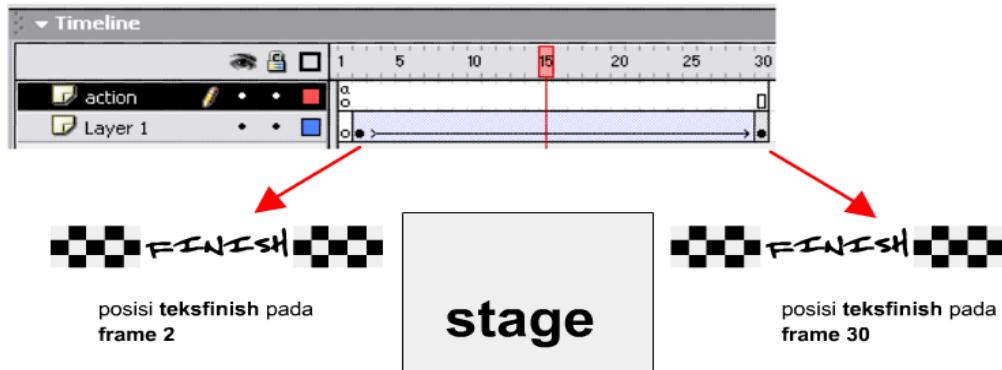


mengegeser keyframe

- Geser posisi movieclip **teksfinish** ke sebelah kiri di luar stage
- Kemudian klik **frame 30** dan masukkan keyframe. Geser posisi movieclip **teksfinish** ke sebelah kanan di luar stage.
- Klik kanan frame 20, kemudian pilih **create motion tween**, maka akan terbentuk animasi teks yang bergerak ke samping.
- Tambahkan sebuah layer di atas layer 1 dan ubah namanya menjadi **layer action**.

10. Klik **frame 1 layer action**, kemudian buka panel action dan ketikan action

```
stop();
```



pengaturan posisi pada movieclip **finish**

11. Kembali ke stage utama dengan menekan Ctrl+E.

12. Klik movieclip **finish** (saat ini hanya berupa titik, karena frame 1 nya kosong), buka panel properties dan ketikan “**finish**” pada instance name.

13. Selanjutnya drag movieclip **mobil** dari library file latihan anda -(file tutorial program15-movieclip.fla) yang sudah anda buka sebelumnya- ke stage, dan letakkan disebelah kiri stage.

14. Klik **mobil** dan ketikan action script berikut :

```
onClipEvent (load) {  
    kecepatan = 10;  
    bataskanan = 700;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>=bataskanan and _root.finish._currentframe == 1) {  
        kecepatan = 0;  
        _root.finish.gotoAndPlay(2);  
    }  
}
```

15. Jalankan movie.

Penjelasan program :

1. Movieclip **finish** pada awalnya diberi sebuah **instance name** agar bisa diakses oleh movieclip lain.

2. Ketika mobil telah sampai di garis batas kanan, maka movieclip **finish** dimainkan pada frame 2 dengan script `_root.finish.gotoAndPlay(2);`; Perhatikan bahwa script tersebut ditulis pada movieclip **mobil** dan dapat mengakses movieclip **finish**.
3. Action **and** `_root.finish._currentframe == 1` pada kondisi **if** diperlukan agar movieclip **finish** tidak “terperangkap” di frame 2. Seandainya action tersebut dihilangkan maka movieclip **finish** tidak akan bergerak.

### **cara penulisan lain bentuk `_root.movieclip`.**

Bentuk `_root` dapat dituliskan dengan cara lain, yaitu:

`_root["NAMA INSTANSI MOVIE CLIP"].action`  
`_root["NAMA INSTANSI MOVIE CLIP"].property`

### **`_parent`.**

bentuk ini pada dasarnya memiliki fungsi yang hampir sama dengan `_root`. Perhatikan diagram penggunaan `_parent`, dan dapat disimpulkan dengan bahasa sederhana bahwa “`_parent` digunakan oleh si anak untuk mengakses induknya”.

### **`this`.**

sedangkan bentuk `this`. akan berimbang pada movieclip dimana action `this`. tersebut diletakkan. Ada dua hal mengenai penulisan `this`, yaitu :

1. `this` bisa tidak ditulis jika action dituliskan pada movieclip. Contoh:

```
onClipEvent(enterFrame){  
    _x += 20;  
}
```

action tersebut ditulis pada movieclip, perhatikan bahwa `this` tidak ditulis didepan action `_x += 20;` meskipun demikian perintah masih dapat dijalankan.

2. `this` harus dituliskan jika action dituliskan pada frame. Contoh :

```
bola.onEnterFrame = function(){  
    this._x += 20;  
}
```

berbeda dengan cara pertama, penulisan pada frame untuk sebuah movie event harus menggunakan action `this`. sebab jika `this`. tidak dituliskan, maka action akan

berimbang pada movie utama (dalam contoh ini, jika **this**. tidak ditulis, movie utama yang akan bergerak ke kanan bukan movieclip bola).

**cara penulisan lain bentuk this.movieclip.**

Bentuk **this.** dapat dituliskan dengan cara lain, yaitu:

**this["NAMA INSTANSI MOVIE CLIP"].action**

**this["NAMA INSTANSI MOVIE CLIP"].property**

## Menggerakkan Movieclip

Game pada dasarnya adalah sebuah gerakan obyek yang terkendali. Dalam game flash obyek yang dikendalikan sebagian besar adalah movieclip. Menggerakkan movieclip dapat dilakukan dengan berbagai cara baik dengan menggunakan motion tween atau menggunakan action scrip. Dengan menggunakan action kita bisa melakukan berbagai gerakan movieclip.

### Menggerakkan Movieclip dengan Teratur

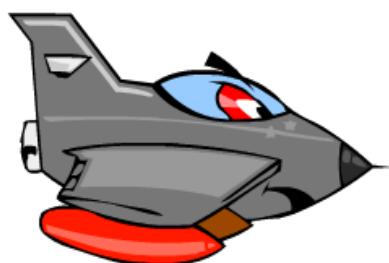
Pergerakan secara teratur sering kita temui dalam sebuah game. Contoh dari gerakan yang teratur ini adalah gerakan pesawat musuh dari kiri ke kanan, gerakan background dan sebagainya



gerakan yang teratur pada game subzero

Untuk menggerakkan movieclip secara teratur, perhatikan contoh berikut :

1. Buatlah sebuah file baru dengan ukuran 800 x 600 px dan 12 fps.
2. Kemudian buatlah sebuah gambar pesawat dengan menggunakan drawing tool



karakter pesawat

3. Seleksi gambar pesawat kemudian convert menjadi **movieclip** dengan nama **jet**.  
Atur posisi **jet**, letakkan di sebelah kiri luar stage.

4. Klik **jet**, kemudian buka panel action kemudian ketikan action berikut :

```
onClipEvent (load) {  
    _y = 100+random(400);  
    xawal = _x;  
    kecepatan = 20;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (_x>800) {  
        _x = xawal;  
        _y = 100+random(400);  
    }  
}
```

5. Simpan movie dan jalankan movie, maka ketika **jet** keluar dari layar, jet akan muncul kembali dengan kordinat y yang berbeda.

Penjelasan program tersebut adalah :

1. Pada saat movieclip **jet** ditampilkan pertama kali **variabel-variabel** yang dibutuhkan diset terlebih dahulu.
2. Baris **\_y = 100+random(400);** berarti kordinat y dari jet diset bernilai acak antara 100 sampai 500. **random(400)** berarti mengacak bilangan 0 – 400, sebagai contoh jika **random(400)** bernilai **250**, maka kordinat y movieclip jet adalah **350**.
3. Pada baris selanjutnya kordinat **x** movieclip jet **ditambah** dengan variabel **kecepatan** secara berulang, sehingga jet akan bergerak ke kanan.
4. Pernyataan **if** dipakai untuk mendeteksi jet. Ketika jet melebihi ukuran stage, posisi **jet** dikembalikan ke **posisi awal** sebelum jet bergerak dan kordinat y movieclip jet **diacak** kembali.

## Mengerakkan Movieclip dengan Acak

Membuat gerakan acak dalam flash dapat dilakukan dengan menggunakan action **random(bilangan acak)**; Perhatikan contoh berikut :

1. Buka file latihan mengerakkan secara teratur diatas, kemudian buka panel library. Kita akan menggunakan movieclip **jet** yang sudah ada.

2. Buatlah sebuah file baru dengan ukuran 800 x 600 px dan 12 fps. Drag movieclip **jet** dari library ke stage. Kemudian atur posisi **jet**, letakkan di sebelah kiri luar stage
3. Klik **jet**, kemudian buka panel action kemudian ketikan action berikut :

```

onClipEvent (load) {
    _y = 100+random(400);
    xawal = _x;
    kecepatan = 10;
    naik = 0;
}

onClipEvent (enterFrame) {
    _x += kecepatan;
    if (random(10) == 3) {
        //gerakan acak naik
        naik = 1;
    } else if (random(10) == 4) {
        //gerakan acak turun
        naik = 2;
    } else if (random(10) == 5) {
        // gerakan lurus
        naik = 0;
    }
    //menggerakkan naik.
    if (naik == 1 and _y>50) {
        _y -= 5;
    }
    if (naik == 2 and _y<550) {
        _y += 5;
    }
    //keluar dari stage, maka kembalikan lagi ke posisi awal
    if (_x>800) {
        _x = xawal;
        _y = 100+random(400);
    }
}

```

```
}
```

4. Simpan movie dan jalankan movie, karena adanya action **random** movieclip **jet** akan bergerak naik turun.

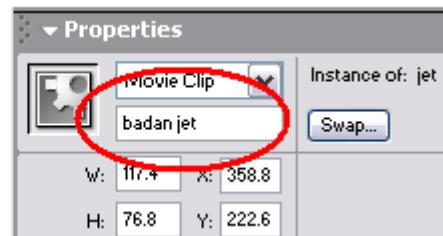
Penjelasan program tersebut adalah :

1. Pada saat movieclip **jet** ditampilkan pertama kali **variabel-variabel** yang dibutuhkan diset terlebih dahulu. Selain itu ditambahkan variabel **naik** dimana akan bernilai 0 jika gerakan lurus, 1 untuk gerakan naik dan 2 untuk gerakan turun.
2. Baris **if (random(10) == 3) {** merupakan baris tempat dimana gerakan acak dihitung. Apabila kondisi bilangan random sama dengan bilangan yang ditentukan, maka **jet** diset bergerak **naik**, **turun** atau **lurus**. Penggunaan **else** dalam hal ini sangat penting, karena tanpa else kemungkinan yang sering muncul adalah pada baris kondisi yang paling bawah yaitu **if (random(10) == 5) {**.
3. Pada baris selanjutnya kordinat **y** diubah seiring dengan nilai variabel **naik**, dan kordinat **x** movieclip jet **ditambah** dengan variabel **kecepatan** secara berulang, sehingga jet akan bergerak ke kanan.

### **Menggerakkan Movieclip dengan Motion Guide**

Selain gerakan yang teratur secara vertikal atau horisontal dan gerakan acak, kita dapat membuat gerakan yang tidak beraturan dalam hal arah, namun teratur dalam pola gerakan. Untuk membuatnya kita dapat memanfaatkan motion guide, contoh penggunaannya adalah sebagai berikut :

1. Buka file latihan menggerakkan secara teratur diatas, kemudian buka panel library. Kita akan menggunakan movieclip **jet** yang sudah ada.
2. Buatlah sebuah file baru dengan ukuran 800 x 600 px dan 12 fps.
3. Drag movieclip **jet** dari library ke stage. Kemudian atur posisi **jet**, letakkan disebelah kiri luar stage.
4. Klik **jet** kemudian buka panel properties dan ketikan “**badanjet**” pada instance name.



movieclip **jet** dan instance name **badanjet**

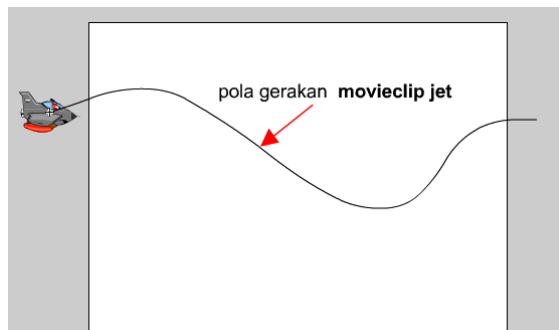
5. Klik movieclip **jet** kemudian convert kembali menjadi **movieclip animasi\_jet**. Kemudian double klik movieclip **animasi\_jet** untuk mengeditnya.
6. Pada mode edit, klik frame 30 dan masukan keyframe.
7. Tambahkan **layer guide** dengan menekan tombol **add motion guide**.



**add motion guide**

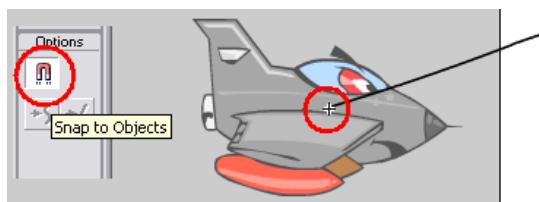
8. Klik **frame 1 layer guide**, kemudian gambarlah **pola gerakan** movieclip jet.

Perhatikan gambar berikut :



**garis motion guide**

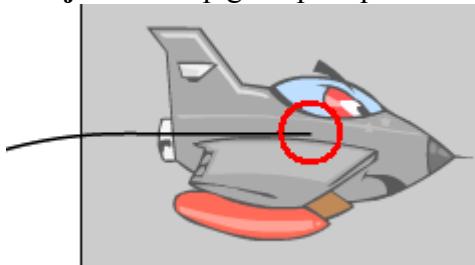
9. Klik **frame 1 layer 1**, pastikan tombol **snap to object** dalam keadaan aktif. Kemudian klik dan drag movieclip **jet** sampai ter-snap pada ujung kiri garis pola.



**option snap to object** dan posisi **jet** terhadap garis pola

10. Klik **frame 1 layer 30**, kemudian klik dan drag movieclip **jet** sampai ter-snap pada ujung kanan garis pola.

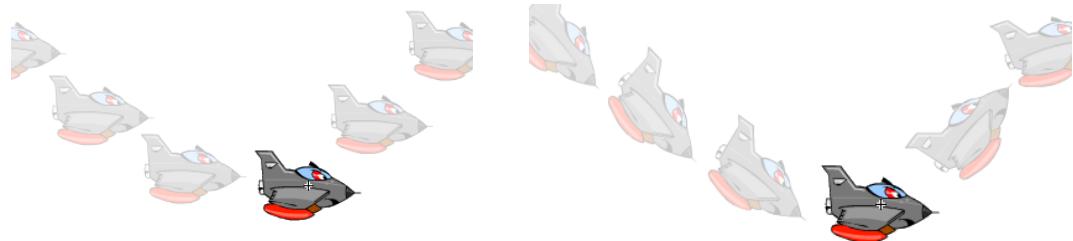
posisi jet terhadap garis pola pada frame 30



11. Klik kanan **frame 20 layer 1**, kemudian pilih **create motion tween**. Maka akan terbentuk sebuah animasi **motion** movieclip jet mengikuti pola yang dibuat. (garis yang kita pakai sebagai pola tidak akan ketika movie dijalankan).
12. Klik **frame 20 layer 1**, kemudian buka panel properties. Seleksi **menu orient to path**, agar arah movieclip jet mengikuti pola.



penggunaan **orient to path**



perbedaan tanpa **orien to path** dan dengan **orient to path**

13. Buatlah sebuah layer di atas **layer guide** dan ubah namanya menjadi **layer action**.
14. Klik **frame 1 layer action**, buka panel action dan ketikan action :

```
stop();
```

15. Keluar dari mode edit symbol dengan menekan tombol Ctrl+E.
16. Klik movieclip **animasi\_jet**, buka panel action dan ketikan action script sebagai berikut :

```
onClipEvent (enterFrame) {  
    if (random(10) == 5 and _currentframe == 1) {  
        _y = 100+random(300);  
        play();  
    }  
}
```

```
}
```

17. Simpan dan jalankan movie. Akan terlihat bahwa movieclip **jet** keluar secara acak dan bergerak sesuai pola

Penjelasan program :

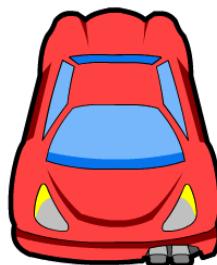
1. Pada awal movie dijalankan, movieclip **jet** tidak bergerak karena adanya action **stop()**; pada **frame 1** layer action di dalam movieclip **animasi\_jet**.
2. Ketika kondisi **if** terpenuhi, maka kordinat **y** dari movieclip **animasi\_jet** diacak dan movieclip tersebut dimainkan, sehingga tampak bergerak.
3. Ketika movieclip animasi **jet** berada pada **frame 30**, akibat sifat movieclip yang selalu looping, maka frame akan berhenti ke **frame 1** lagi, dan kondisi **\_currentframe == 1** dapat dipenuhi lagi jika ada bilangan random yang tepat.

### **Menggerakkan Movieclip dengan Keyboard**

Salah satu alat pengendali gerakan dalam sebuah game adalah keyboard. Hampir sebagian besar game selalu melibatkan penggunaan keyboard. Flash menyediakan action khusus untuk mendeteksi adanya masukan dari tombol keyboard yaitu dengan action **Key**. Contoh dari penggunaan action **Key**. adalah sebagai berikut.

1. Buat sebuah file baru dengan ukuran 800 x600 pixel, dan 12 fps.
2. Buatlah sebuah mobil dengan pandangan atas-dengan menggunakan drawing tool.

Seperti contoh gambar berikut :



gambar mobil prespektif tampak atas

3. Seleksi gambar mobil tersebut dan convert menjadi **movieclip** bernama **mobil**.
4. Klik **mobil**, kemudian buka panel action dan ketikan action berikut :

```
onClipEvent (enterFrame) {  
    if (Key.isDown(Key.LEFT)) {  
        // bergerak ke kiri jika panah kiri ditekan  
        x -= 10;
```

```
    }
    if (Key.isDown(Key.RIGHT)) {
        // bergerak ke kanan jika panah kiri ditekan
        _x += 10;
    }
}
```

5. simpan file dan jalankan movie. Kemudian tekan keyboard panah kanan dan panah kiri, lihat pergerakan mobilnya.

Penjelasan program :

1. Action **Key.isDown()** akan membaca apakah ada tombol yang ditekan, dan di dalam tanda () action **Key.LEFT / Key.RIGHT** adalah kode tombol yang ditekan.
2. pada baris kondisi yang pertama (**if (Key.isDown(Key.LEFT)) { }** ), berarti ketika tombol panah kiri ditekan, maka kordinat x mobil dikurangi agar mobil bergerak kekiri. Begitu juga dengan gerakan ke kanan ketika tombol panah kanan ditekan.

Catatan :

Dalam flash beberapa tombol telah didefinisikan, seperti tombol panah, Escape, Enter, Spasi, Shift dan tombol penting lainnya. Akan tetapi beberapa tombol seperti tombol karakter A..Z, angka 1..0, tombol fungsi F1...F12 tidak didefinisikan secara otomatis. Untuk itu kita harus menggunakan perintah **Key.getCode()** pada movie event **keyDown**.

Agar proses lebih mudah, pada CD tutorial terdapat file **keyboard.exe** yang dapat mendefinisikan kode tombol secara otomatis. Contoh penggunaannya sebagai berikut: misal kita akan menggunakan tombol “A”, dengan program tersebut diketahui kode tombol “A” adalah **65**, maka script yang dipakai adalah :

**if (Key.isDown(65)){...}.**

## Menggerakkan Movieclip dengan Mouse

Selain keyboard kita juga dapat menggerakan sebuah movieclip dengan mouse. Gerakan yang dikendalikan dengan mouse sering kita temui pada game bertipe shooting atau permainan menembak. Pada dasarnya terdapat dua cara menggerakkan mouse yaitu dengan menggunakan action **startDrag()** dan menggunakan teknik mouse tracking.



contoh gerakan yang dikendalikan dengan mouse

## Menggerakkan Movieclip dengan Mouse Menggunakan **startDrag();**

Pada beberapa game bertipe shooting, kita akan mendapati bentuk kursor mouse yang dirubah menjadi bentuk sasaran tembak. Untuk merubah bentuk mouse, perhatikan contoh berikut :

1. Buatlah sebuah file baru berukuran 800 x 600 pixel dan 12 fps.
2. Dengan menggunakan drawing tool, buatlah sebuah gambar kursor yang baru.



bentuk kursor

3. Seleksi gambar kursor tersebut, kemudian convert menjadi symbol **movieclip** dengan nama **kursor**.
4. Klik **kursor**, buka panel action dan ketikan action berikut :

```
onClipEvent (load) {  
    Mouse.hide();  
}  
  
onClipEvent (enterFrame) {
```

```
    startDrag(this, true);  
}
```

5. Jalankan movie, kemudian gerakkan mouse.

Penjelasan program :

1. Action **Mouse.hide()**; berfungsi untuk menghilangkan gambar kursor default mouse pada layar.
2. Action **startDrag(this, true)**; berarti movieclip yang dituju (**this**) akan mengikuti gerakan mouse. Selain itu, posisi registration movieclip yang dikenai action startDrag, akan dijadikan sama dengan kordinat x dan y mouse akibat penambahan variabel boolean **true**.

Catatan :

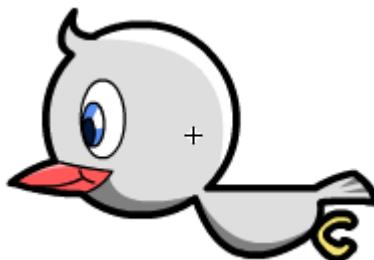
action **startDrag()**; hanya bisa diberikan pada satu movieclip saja, apabila ada dua atau lebih yang memiliki actin tersebut di dalamnya, maka action hanya akan berefek pada salah satu dari movieclip tersebut.

### **Mengerakkan Movieclip dengan Mouse secara Tracking**

Dengan **startDrag** gerakan movieclip selalu **sama** dengan gerakan mouse. Akan tetapi dalam beberapa game dengan tombol kendali mouse seperti feeding frenzy, alien sky, dan beberapa game lainnya, karakter game bergerak **mendekati** kursor mouse dengan kecepatan tertentu. Hal tersebut sering disitilahkan dengan nama **mouse tracking**.

Contoh pembuatan mouse tracking adalah sebagai berikut :

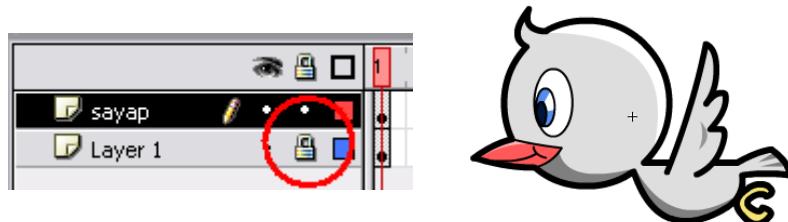
1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, dan 12 fps.
2. Dengan menggunakan drawing tool buatlah sebuah gambar burung tanpa sayap.



gambar burung tanpa sayap dengan drawing tool

3. Seleksi gambar tersebut dan convert menjadi symbol **movieclip** dengan nama **burung**.
4. Double klik movieclip **burung**, untuk mengeditnya.

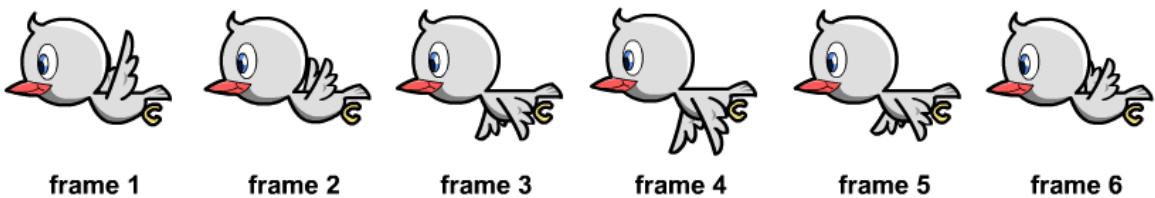
- Dalam mode edit, tambahkan sebuah layer diatas layer 1 dan ubah namanya menjadi **layer sayap**.
- Klik **frame1 layer sayap**, kemudian gambarlah bentuk sayap. Untuk memudahkan proses, klik **lock layer 1**.



lock layer dan gambar sayap

- Klik **frame 6 layer 1**, kemudian masukkan **frame** dengan menekan tombol **F5**.
- Klik **frame 2 layer 2**, masukkan **keyframe** dengan menekan tombol **F6**, kemudian ubah posisi sayap.
- Lakukan langkah no 8 pada frame 3 sampai frame 6 sehingga terbentuk sebuah gerakan sayap secara **frame by frame**.
- Buatlah layer baru di bawah layer 1 dan ubah namanya menjadi **layer sayap2**.
- Dengan cara yang sama, buatlah animasi frame by frame sayap keduanya.

Perhatikan gambar berikut.



animasi burung secara frame by frame

- Keluar dari mode edit symbol, dengan menekan **Ctrl+E**;
- Klik **burung**, kemudian buka panel action dan ketikan action berikut:

```
onClipEvent (load) {
    Mouse.hide();
    skala = _xscale;
    perlambatan = 20;
}

onClipEvent (enterFrame) {
    // mengubah skala burung
    if (_root._xmouse > _x) {
```

```

_xscale = -skala;
} else {
    _xscale = skala;
}
// tracking
_x = _x+(_root._xmouse-_x)/perlambatan;
_y = _y+(_root._ymouse-_y)/perlambatan;
}

```

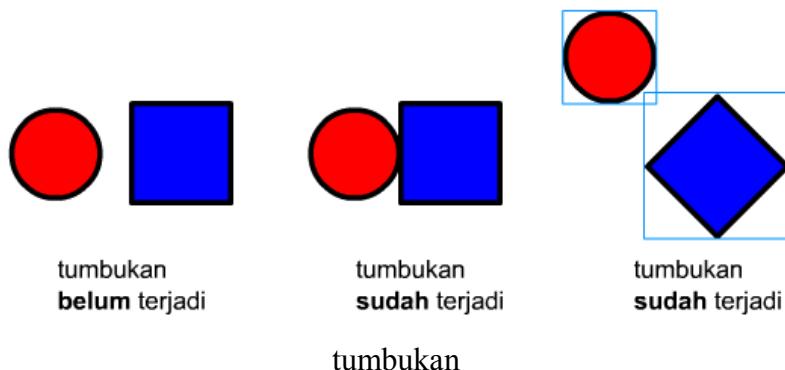
14. Simpan file dan jalankan movie. Gerakkan mouse untuk melihat gerakan burung.

Penjelasan program :

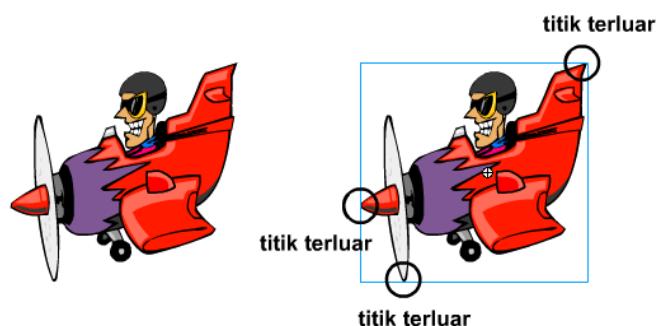
1. Sebelum memulai tracking, kursor mouse dibuat tidak tampak terlebih dahulu dengan action **Mouse.hide()**. Selanjutnya variabel **skala** akan digunakan untuk merubah bentuk movieclip burung, dan variabel **perlambatan** menentukan kecepatan bergerak burung, semakin tinggi nilai perlambatan, gerakan burung semakin lambat.
2. Baris kondisi (**if (\_root.\_xmouse>\_x){ }**) digunakan untuk mendeteksi posisi kursor. Jika kursor ada disebelah kiri burung, maka posisi burung dibalik dengan menggunakan action **\_xscale = -skala**.
3. Rumus sederhana dari mouse traking adalah menghitung jarak antara mouse dengan movieclip kemudian dibagi dengan perlambatan. (**\_x = \_x + (\_root.\_xmouse-\_x) / perlambatan;**)

## Mendeteksi Tumbukan

Tumbukan atau tabrakan adalah sebuah kejadian dimana terdapat dua movieclip yang saling bertumbukan. Perhatikan gambar berikut untuk mengetahui sebuah tumbukan :



Perhatikan gambar tumbukan yang ketiga, menurut mata kita movieclip lingkaran dengan movieclip kotak . Akan tetapi dalam flash tumbukan dihitung dari sebuah bentuk kotak yang dibentuk oleh garis-garis yang ditarik dari titik terluar sebuah movieclip. Perhatikan kembali gambar berikut :



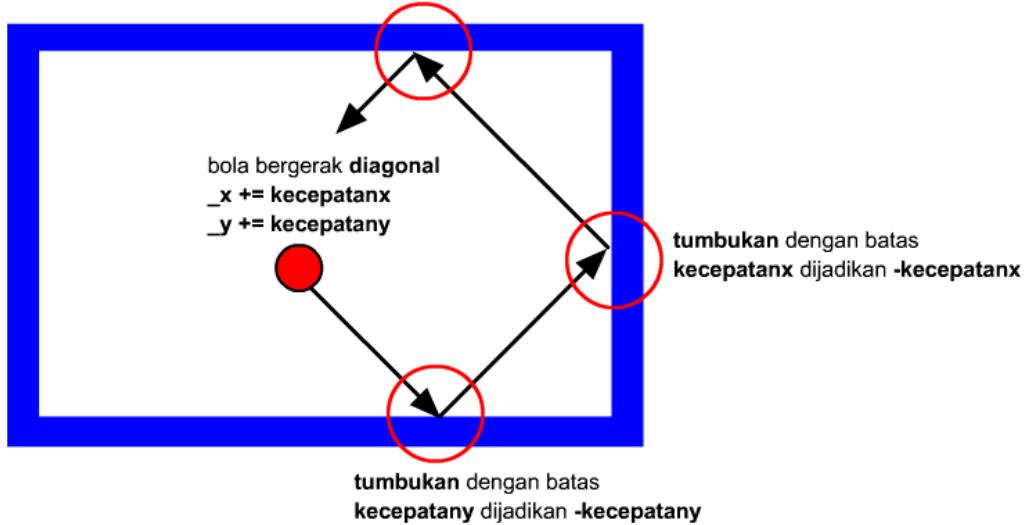
area yang dideteksi pada saat tumbukan

## Mendeteksi Tumbukan dengan **hitTest()**:

Salah satu action untuk mendeteksi tumbukan dalam flash adalah dengan menggunakan **hitTest()**. Contoh dari penggunaan action **hitTest()** adalah sebagai berikut :

Permasalahan:

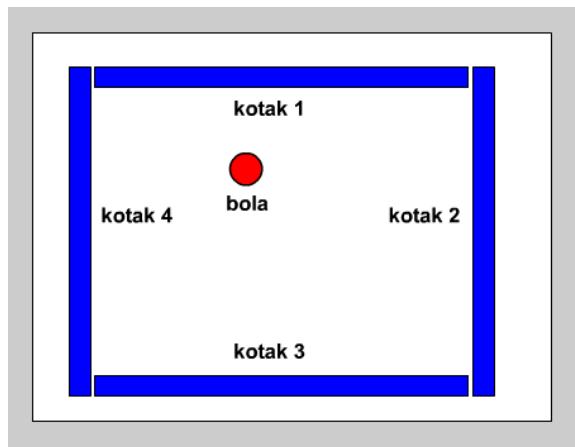
Buatlah sebuah bola yang bergerak memantul secara terus menerus dalam sebuah area Untuk menyelesaikan masalah tersebut, kita visualisasikan terlebih dahulu menjadi :



logika tumbukan sederhana

Proses pembuatan aplikasi tersebut adalah :

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, 12 fps.
2. Buatlah sebuah gambar lingkaran menggunakan oval tool. Seleksi lingkaran dan convert menjadi **movieclip** dengan nama **bola**.
3. Klik **bola**, kemudian buka panel properties dan ketikan “**bola**” pada instance name.
4. Buat sebuah gambar kotak menggunakan rectangle tool. Seleksi kotak tersebut dan convert menjadi **movieclip** dengan nama **kotak**.
5. Klik **kotak**, lakukan **copy** dan **paste** sebanyak 3 kali dan atur posisinya sesuai dengan gambar berikut :



penataan obyek pada stage.

6. Klik **bola**, kemudian bka panel action dan ketikan action berikut :

```
onClipEvent (load) {
    kecepatanx = 10;
```

```
kecepatany = 10;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatanx;  
    _y += kecepatany;  
}
```

7. Klik **kotak** pertama (sebelah atas), kemudian ketikan action sebagai berikut :

```
onClipEvent (enterFrame) {  
    if (hitTest(_root.bola)) {  
        _root.bola.kecepatany = -_root.bola.kecepatany;  
    }  
}
```

8. **Copy** seluruh action pada movieclip **kotak** tersebut, kemudian klik movieclip **kotak** ke 3 (sebelah bawah), buka panel action dan **paste**-kan action yang sudah dicopy sebelumnya.
9. Klik **kotak** kedua (sebelah kanan), kemudian ketikan action sebagai berikut :

```
onClipEvent (enterFrame) {  
    if (hitTest(_root.bola)) {  
        _root.bola.kecepatanx = -_root.bola.kecepatanx;  
    }  
}
```

10. **Copy** seluruh action pada movieclip **kotak** tersebut, kemudian klik movieclip **kotak** ke 4 (sebelah kiri), buka panel action dan **paste**-kan action yang sudah dicopy sebelumnya.
11. Simpan dan jalankan movie.

Penjelasan program :

1. Pada movieclip **bola** diberikan instance name agar bisa diakses oleh movieclip lain.
2. Selanjutnya **bola** digerakkan secara diagonal dengan penambahan kordinat x dan y, dengan variabel **kecepatanx** dan **kecepatany**.

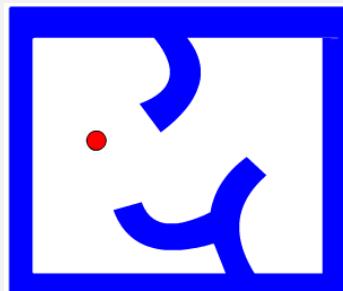
3. Pada **kotak 1** sampai **kotak 4** terdapat suatu kondisi **if (hitTest(\_root.bola))** yang berarti jika kotak bertumbukan dengan bola maka...
4. Pada **kotak 1** dan **kotak 3**, jika kondisi tumbukan tercapai, variabel **kecepatany dibalik** nilainya sehingga bola memantul ke atas atau ke bawah.
5. Sedangkan pada **kotak 2** dan **kotak 4**, jika kondisi tumbukan tercapai, variabel **kecepatanx** dibalik nilainya sehingga bola memantul ke kanan atau ke kiri.

### **Mengatasi Kesalahan hitTest dengan getBounds();**

Kelemahan action hitTest adalah perhitungan area tumbukan yang dimulai dari titik-titik terluar, sehingga ketika obyek belum bertumbukan, ada peluang kondisi hitTest terpenuhi. Untuk mengatasi kelemahan tersebut, ada suatu action dalam flash yaitu **getBounds();**

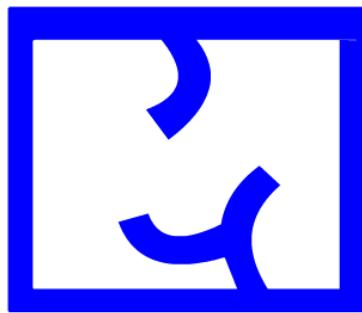
Permasalahan:

Buatlah sebuah bola yang bergerak memantul secara terus menerus dalam sebuah area yang tampak pada gambar



Permasalahan yang muncul adalah , bentuk area tersebut cukup rumit dan kurang beraturan, sehingga penggunaan action hitTest() saja tidak akan menghasilkan program yang baik. Untuk itu digunakan action getBound() sebagai berikut :

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, 12 fps.
2. Buat sebuah gambar area seperti pada gambar menggunakan drawing tool. Seleksi kotak tersebut dan convert menjadi **movieclip** dengan nama **ruangan**.



gambar movieclip ruangan

3. Klik **ruangan**, kemudian buka panel properties dan ketikan “**ruangan**” pada instance name.
4. Buatlah sebuah gambar lingkaran menggunakan oval tool. Seleksi lingkaran dan convert menjadi **movieclip** dengan nama **bola**. Letakkan bola didalam ruangan, lihat gambar sebelumnya.
5. Klik **bola**, kemudian buka panel action dan ketikan action berikut :

```
onClipEvent (load) {  
    kecepatanx = 5+random(10);  
    kecepatany = 5+random(10);  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatanx;  
    _y += kecepatany;  
    // mendeteksi tumbukan dengan getBounds  
    if (_root.ruangan.hitTest(getBounds(_root).xMax, _y, true)) {  
        kecepatanx = -kecepatanx;  
    }  
    if (_root.ruangan.hitTest(getBounds(_root).xMin, _y, true)) {  
        kecepatanx = -kecepatanx;  
    }  
    if (_root.ruangan.hitTest(_x, getBounds(_root).yMax, true)) {  
        kecepatany = -kecepatany;  
    }  
    if (_root.ruangan.hitTest(_x, getBounds(_root).yMin, true)) {  
        kecepatany = -kecepatany;  
    }  
}
```

}

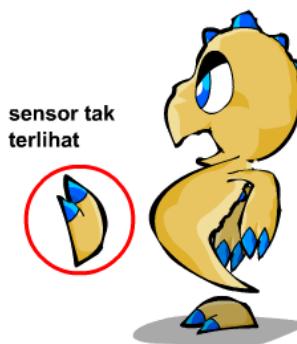
6. Simpan file dan jalankan movie.

Penjelasan program :

Penggunaan **getBounds** dilakukan di dalam action **hitTest()**, dengan **\_root.nama instansi obyek yang akan ditumbuk** didepan **hitTest**. Variabel **xMax**, **xMin**, **yMax** dan **yMin** merupakan property yang dimiliki oleh obyek yang dihitung tumbukannya (**bola**).

### **Mengatasi Kesalahan hitTest dengan Sensor**

Selain menggunakan getBound kita juga dapat menyiasati sebuah tumbukan dengan menambahkan sensor yang tidak terlihat. Sebagai contoh, dalam game bertipe action, setiap anggota badan harus dibuat sedetail mungkin dalam hal tumbukannya dengan obyek lain. Perhatikan gambar berikut:



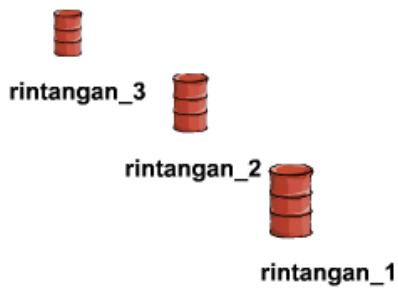
gerakan karakter dalam game **Mocil**

Pada salah satu gerakan karakter dalam game **Mocil** pada saat menendang, maka area yang dihitung dalam hitTest hanya **kaki** dari monster tersebut, sedangkan bagian badan yang lain tidak dihitung. Perhatikan contoh penggunaan sensor berikut :

Permasalahan :

Buatlah sebuah gerakan mobil yang dapat menghindari rintangan secara otomatis.

Visualisasi dari permasalahan tersebut adalah sebagai berikut :



visualisasi dari aplikasi sensor

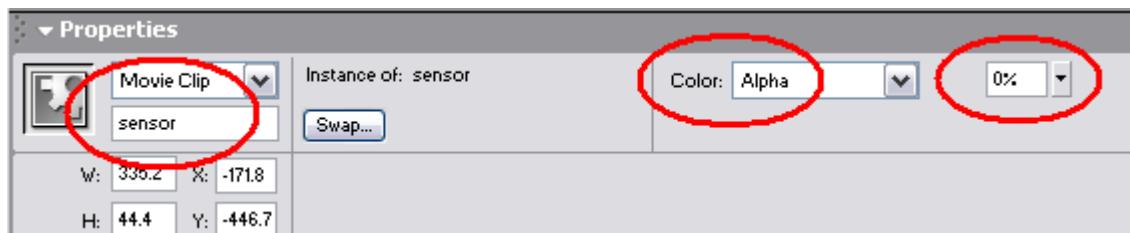
Penyelesaiannya adalah sebagai berikut :

1. Sebelumnya buka terlebih dahulu file latihan menggerakkan movieclip dengan keyboard yang kita buat pada latihan sebelumnya (buka file program32-mc\_gerakan\_keyboard.fla pada CD tutorial). Kemudian buka panel **library**.
2. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, 12 fps.
3. Buatlah sebuah gambar kotak dengan menggunakan rectangle tool, kemudian convert menjadi **movieclip** dengan nama **sensor**.
4. Drag **mobil** dari library file yang kita buka sebelumnya (pada point 1) dan letakkan disebelah kanan bawah stage.
5. Double klik **mobil**, untuk mengeditnya.
6. Dalam mode edit, buatlah sebuah layer baru diatas **layer 1** dan ubah namanya menjadi **layer sensor**.
7. Klik **frame 1 layer sensor**, drag movieclip **sensor** dari library ke stage, kemudian atur posisinya. Letakkan **sensor** di depan mobil.



posisi sensor terhadap gambar mobil

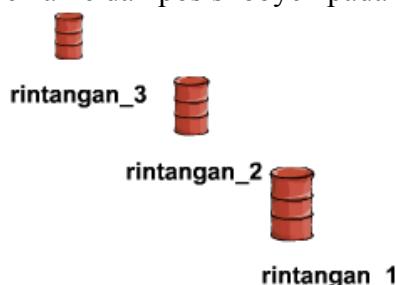
8. Klik movieclip **sensor**, kemudian buka panel properties. Ketikan “**sensor**” pada instance name dan tambahkan efek **alpha 0%** pada **color**.



properties movieclip sensor

9. Kembali ke stage utama dengan menekan Ctrl+E.
10. Kemudian buatlah sebuah gambar rintangan berupa gambar tong, kemudian seleksi gambar dan convert menjadi **movieclip** dengan nama **tong**.
11. **Copy tong** dan **paste-kan** sebanyak 2 kali kemudian atur posisi masing-masing sesuai pada gambar.
12. Tambahkan instance name pada masing-masing movieclip **tong**, yaitu “**rintangan\_1**”, “**rintangan\_2**” dan “**rintangan\_3**”.

instance name dan posisi obyek pada stage



13. Klik movieclip **mobil**, buka panel action dan ketikan script berikut :

```

onClipEvent (load) {
    skala = _xscale;
    kecepatan = 10;
}

onClipEvent (enterFrame) {
    //gerakan prespektif ke atas
    if (_y>50) {
        _y -= kecepatan;
        kecepatan -= 0.1;
        _xscale = skala;
    }
}

```

```
_yscale = skala;  
skala -= 0.1;  
}  
// menghindari rintangan  
if(sensor.hitTest(_root.rintangan_1) or sensor.hitTest(_root.rintangan_2)  
or  
sensor.hitTest(_root.rintangan_3)) {  
_x -= 10;  
}  
}
```

14. Simpan dan jalankan movie.

Penjelasan program :

1. Sebelumnya movieclip **sensor** diletakkan didalam movieclip mobil dan diberikan efek **alpha = 0%** agar tidak terlihat saat movie dimainkan.
2. Selanjutnya dilakukan pengesetan variabel yang akan digunakan yaitu variabel **skala** dan variabel **kecepatan**.
3. Untuk menggerakan **mobil** secara prespektif, maka setiap gerakan ke atas diikuti dengan pengurangan variabel **skala** dan variabel **kecepatan** sebanyak **(0,1)** bilangan tersebut dapat dirubah sesuai dengan ukuran movie.
4. Pada baris kondisi **if** selanjutnya, kordinat x movieclip **mobil** dikurangi ketika **sensor** dari mobil bertumbukan dengan **rintangan**.

## **Load Movie**

Salah satu kelemahan dari flash adalah movie yang berjalan lebih lambat ketika ukuran file besar dan action yang dijalankan lebih banyak. Berapakah batasan maksimal suatu ukuran movie flash agar berjalan normal? Jawaban dari pertanyaan tersebut sangat tergantung dengan spesifikasi komputer yang akan menjalankan movie tersebut. Akan tetapi jika kita mengkhususkan diri pada flash game yang dipublis pada suatu situs internet, maka ukuran maksimal movie tergantung pada kecepatan modem.

Meskipun tidak ada kepastian akan ukuran maksimal suatu flash movie, membuat ukuran movie flash menjadi semakin kecil akan mengoptimalkan kerja file. Untuk memperkecil ukuran file, salah satu cara yang dipakai adalah memecah movie menjadi file-file kecil-yang pada akhirnya file tersebut dikumpulkan menjadi satu kesatuan.

Perhatikan contoh berikut :

Game DigDog memiliki 60 level. Pada saat pemain memasuki level 1, berarti level 2 sampai level 60 tidak dipakai. Apabila seluruh level dijadikan dalam satu file, akan menyebabkan suatu kerugian-yaitu file level 2 – level 60 yang sudah terload tidak dipakai. Hal itulah yang menyebabkan game berjalan lebih lambat. Solusi yang terbaik adalah memecah file game menjadi 1 level per 1 level.

Untuk memanggil pecahan file, flash menyediakan action loadMovieNum() dan loadMovie().

### **loadMovieNum()**

Action ini memiliki bentuk sebagai berikut :

**loadMovieNum(“nama file yang akan dipanggil”, level kedalaman);**

penulisan nama file harus dibuat secara mendetail dengan ekstensi filenya. contoh “**level1.swf**”.

Level kedalaman merupakan level tempat movie di load. Perhatikan sistem level pada bab movieclip. Hal yang perlu diperhatikan adalah penggunaan level 0 dan level selain 0. Perhatikan contoh berikut :

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, 12 fps.
2. Dengan menggunakan drawing tool, buatlah tampilan pada stage seperti tampak pada gambar berikut :

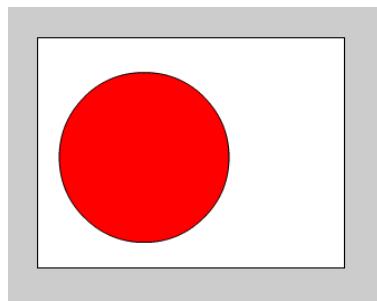


tampilan pada stage

3. Buatlah sebuah **tombol “load file2”**, dan letakkan di pojok kanan bawah stage.
4. Klik tombol “**load file2**”, kemudian buka panel action dan ketikan action berikut :

```
on (release) {  
    loadMovieNum("file2.swf", 0);  
}
```

5. Simpan file dengan nama “**file1.fla**”, dan jalankan movie.
6. Tutup movie dengan menekan tombol Ctrl+W.
7. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, 12 fps.
8. Dengan menggunakan drawing tool, buatlah tampilan pada stage seperti tampak pada gambar berikut :



tampilan “file 2” pada stage

9. Simpan file dengan nama “**file2.fla**”, dan jalankan movie.
10. Tutup movie dengan menekan tombol Ctrl+W.
11. Buka kembali file “**file1.fla**”, kemudian jalankan movie. Tekan tombol “**load file2**”, maka tampilan pada movie akan berganti dengan tampilan file 2.

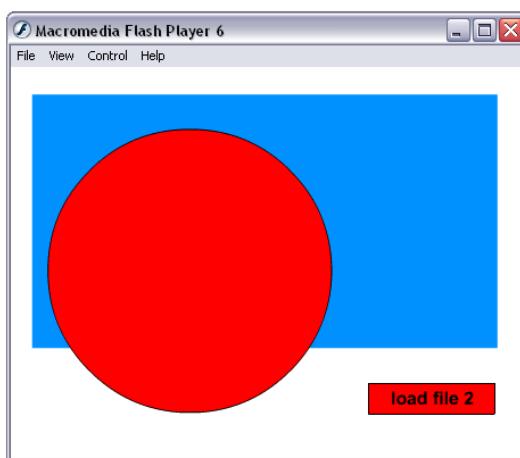
Ketika nilai **level kedalaman** adalah **0**, maka eksekusi perintah **loadMovieNum(namafile, 0);** menyebabkan pergantian tampilan file secara total. Pada contoh diatas, ketikan tombol **load file2** ditekan, maka tampilan movie pada **file 1 segera berganti dengan** tampilan movie **file 2.**

Untuk memahami penggunaan level kedalaman selain 0, perhatikan contoh berikut:

1. Buka file “**file1.fla**” pada latihan diatas (buka file “**program38-loadMovieNum\_file1.fla**” pada CD tutorial);
2. Klik tombol “**load file2**”, buka panel action dan ubah action script nya menjadi :

```
on (release) {  
    loadMovieNum("file2.swf", 1);  
}
```

3. Simpan file dan jalankan movie. Tekan tombol “**load file2**”. maka akan mucul tampilan sebagai berikut :



penggunaan level 1 pada **loadMovieNum()**.

Penggunaan level 1 akan menyebabkan movie tidak berganti dengan yang baru, tetapi movie yang diload menindih (berada diatas) file yang memanggil. Perhatikan bahwa tombol “**load file2**” masih aktif, begitu juga ketika kita meletakkan tombol pada file yang diload (file 2), tombol tersebut juga aktif.

Catatan :

File yang di-load harus berada dalam satu folder yang sama atau jika diluar folder harus diberikan sebuah nama folder yang jelas (contoh:

**loadMovieNum(“...\\gambar\\musuh1.swf”, 1);**

Penggunaan level kedalaman bernilai negatif (-1, -2, .. ), akan mengganti tampilan file dengan tampilan file yang diload, tetapi akan ditampilkan dalam sebuah **internet**

**browser.**

### **unloadMovieNum()**

Untuk menutup / menghilangkan / meng-unload file yang telah kita load dengan menggunakan action loadMovieNum() sebelumnya, flash menyediakan action : unLoadMovieNum(level kedalaman movie yang akan di-unload). Perhatikan contoh penggunaannya berikut ini :

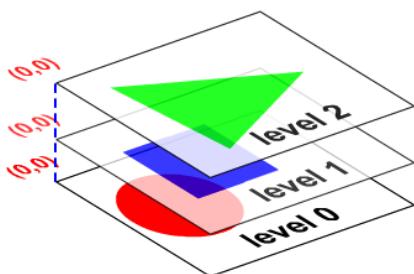
1. Buka “file2.fla” pada latihan sebelumnya (buka file “program39-loadMovieNum\_file2.fla” pada CD tutorial).
2. Seleksi lingkaran besar atau obyek apapun yang ada pada stage, kemudian convert menjadi **button** dengan nama “**unload file2**”.
3. Seleksi tombol “unload file2”, buka panel action dan ketikan action berikut :

```
on (release) {  
    unloadMovieNum(1);  
}
```

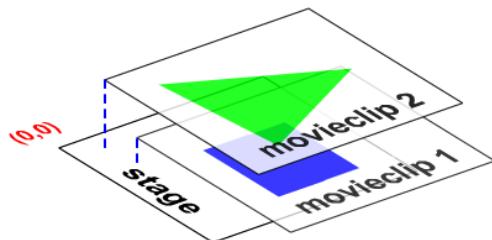
4. Simpan dan jalankan file.
5. Tutup movie dengan menekan tombol Ctrl+W.
6. Buka kembali file “**file1.fla**”, kemudian jalankan movie. Tekan tombol “**load file2**”, setelah tampilan **file 2** menumpuk diatas tampilan **file 1**, klik tombol “**unload file2**”, maka tampilan **file 2** akan hilang (ter-unload).

### **loadMovie()**

Action **loadMovieNum()** memiliki kelemahan yaitu kordinat tempat me-load selalu **(0, 0)**. Jika kita menginginkan file diload pada kordinat tertentu, kita harus menggunakan cara lain, yaitu menggunakan action **loadMovie()**; Untuk lebih jelasnya perhatikan gambar berikut :



penggunaan **loadMovieNum()**



penggunaan **loadMovie()**

penggunaan loadMovieNum dan loadMovie

Berbeda dengan loadMovieNum(), pada penggunaan **loadMovie()** kita membutuhkan sebuah **movieclip** sebagai tempat di-loadnya suatu movie. Untuk jelasnya perhatikan permasalahan berikut :

Permasalahan :

Buatlah sebuah game arcade sederhana dimana pada setiap level terdapat musuh yang berbeda-beda.

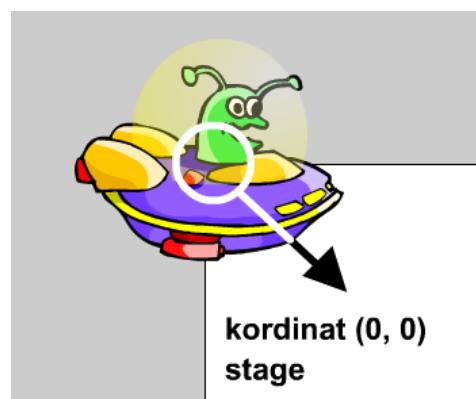
Perhatikan contoh dibawah ini untuk membuat **gerakan musuh** pada masalah tersebut:

1. Hal yang difokuskan pada masalah tersebut adalah “adanya musuh yang berbeda-beda setiap level”. Untuk mengoptimalkan kerja, kita dapat menggunakan teknik **loadMovie()**; dengan membuat movie musuh terpisah-pisah.
2. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, dan 12 fps.
3. Buatlah sebuah gambar pesawat musuh yang pertama, seperti pada gambar atau sesuai dengan kreativitas anda.



musuh level pertama

4. Seleksi gambar pesawat, kemudian convert menjadi **movieclip** dengan nama “**musuh1**”. Kemudian atur posisinya terhadap stage. Letakkan pada pojok kiri atas stage, sampai titik registrationnya berada pada kordinat **(0, 0)**.



peletakan movieclip terhadap stage

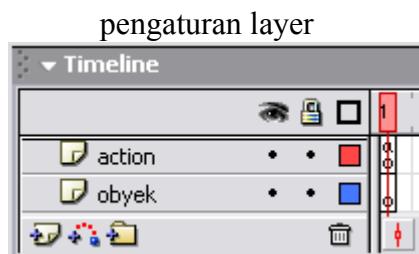
5. Klik “**musuh1**”, buka panel properties dan ketikan “**musuh**” pada instance name.

6. Simpan file dengan nama “**musuh1.fl**”, kemudian jalankan movie.
7. Kembali ke stage utama dengan menekan Ctrl+W.
8. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, dan 12 fps.
9. Buatlah sebuah gambar musuh yang kedua, seperti pada gambar atau sesuai dengan kreativitas anda.

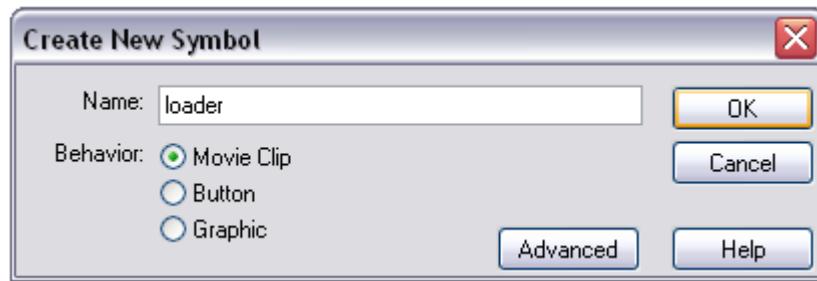


musuh level kedua

10. Seleksi gambar musuh kemudian convert menjadi **movieclip** dengan nama “**musuh2**”. Letakkan pada pojok kiri atas stage, sampai titik registrationnya berada pada kordinat **(0, 0)**.
11. Klik “**musuh2**”, buka panel properties dan ketikan “**musuh**” pada instance name.
12. Simpan file dengan nama “**musuh2.fl**”, kemudian jalankan movie.
13. Kembali ke stage utama dengan menekan Ctrl+W.
14. Setelah movie musuh dibuat langkah selanjutnya adalah membuat file utamanya.
15. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, dan 12 fps.
16. Buatlah sebuah layer baru, dan ubah masing-masing nama layer menjadi layer “**obyek**” dan layer “**action**”.



17. Selanjutnya buatlah sebuah **movieclip kosong** dengan cara menekan menu **insert>new symbol**. Pilih movieclip pada **behaviour** dan ketikan **loader** pada nama, kemudian tekan **OK**.



membuat symbol baru

18. Selanjutnya pada mode edit symbol movieclip **loader**, jangan tambahkan apa-apa dan langsung kembali ke stage utama dengan menekan tombol Ctrl+E.
19. Klik **frame 1 layer obyek**, kemudian drag movieclip **loader** dari library ke stage (buka panel library terlebih dahulu jika belum terbuka). Letakkan di sebelah kiri luar stage. Kemudian buka panel properties dan ketikan “**loader**” pada instance name.
20. Klik **loader** kemudian buka panel action dan ketikan script sebagai berikut :

Catatan :

Action berikut ini adalah action untuk membuat gerakan acak, sama dengan action pada latihan menggerakan movieclip secara acak pada latihan sebelumnya. Anda dapat melakukan copy dan paste script.

```
onClipEvent (load) {  
    _y = 100+random(400);  
    xawal = _x;  
    kecepatan = 10;  
    naik = 0;  
}  
  
onClipEvent (enterFrame) {  
    _x += kecepatan;  
    if (random(10) == 3) {  
        //gerakan acak naik  
        naik = 1;  
    } else if (random(10) == 4) {  
        //gerakan acak turun  
        naik = 2;  
    } else if (random(10) == 5) {  
        // gerakan lurus
```

```

naik = 0;
}
//menggerakkan naik.
if (naik == 1 and _y>50) {
    _y -= 5;
} else {
    //menggerakkan turun.
}
if (naik == 2 and _y<550) {
    _y += 5;
}
//keluar dari stage, maka kembalikan lagi ke posisi awal
if (_x>800) {
    _x = xawal;
    _y = 100+random(400);
}
}

```

21. Selanjutnya klik **frame 1 layer action**, buka panel action dan tambahkan script :

```

level = 2;
// mengubah jenis musuh berdasarkan level game
_root.loader.loadMovie("musuh"+level.toString()+".swf");

```

22. Jalankan movie, kemudian pada **frame 1 layer action**, ubahlah nilai variabel **level**

pada action **level = 1;** menjadi **level = 2;** dan lihat hasilnya.

Penjelasan program :

1. Untuk menggerakan secara acak, penjelasan yang sama terdapat pada latihan menggerakan movieclip secara acak pada bab sebelumnya.
2. Pada frame dilakukan pengesetan variabel level dengan nilai awal **level = 1;**.
3. Pada baris `_root.loader.loadMovie("musuh"+_root.level.toString()+".swf");` dilakukan peng-load-an movie “musuh” sesuai dengan nilai variabel level. action `level.toString()` akan mengconvert variabel level yang bertipe number menjadi variabel baru bertipe string. Bentuk `_root.loader.loadMovie("musuh"+_root.level.toString()+".swf");` juga dapat ditulis dengan :

```
loadMovie("musuh"+_root.level.toString()+".swf", _root.loader);
```

4. Penambahan instance name pada masing masing movie ("musuh1" dan "musuh2") dimaskudkan untuk proses selanjutnya, seperti perhitungan tumbukan, kordinat dan sebagainya.

## **Memulai Pembuatan Game**

Setelah kita belajar dan memahami tentang action script dasar yang dipakai dalam flash, selanjutnya kita dapat memulai proses pembuatan game.

### **Ide Game**

Ide game adalah sebuah gagasan mengenai game yang akan dibuat. Ide game tersebut meliputi jenis atau tipe game, karakter yang digunakan, cerita / storyboard dari game, dan segala detail yang ada pada bayangan kita sebelum membuat game.

Manusia memiliki 2 sistem memory yaitu memori jangka panjang dan memori jangka pendek. Ide game yang terlintas begitu saja dalam pikiran kita akan diingat dalam memori jangka pendek, sehingga ide tersebut sangat mudah kita lupakan. Untuk itu disarankan agar kita menuliskan atau memvisualisasikan suatu ide sejak pertama kali ide tersebut terlintas pada pikiran kita.

### **Jenis Game**

Saat ini sangat banyak kita temukan game, baik itu game PC, game console (seperti PS, XBOX, PSP, Nintendo DS), game HP (mobile game) dan game flash. Dari banyaknya game yang ada, berdasarkan gameplaynya (cara memainkannya), game dapat diklasifikasikan menjadi beberapa jenis. Dalam buku ini klasifikasi game flash adalah sebagai berikut :

#### 1. Board game

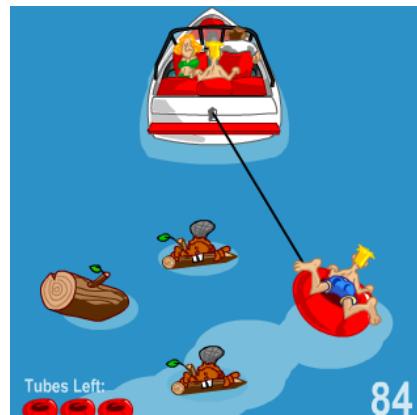
Board game atau dapat kita istilahkan sebagai “permainan papan”. Pada jenis game ini, pemain diberikan sebuah tampilan yang berisi tentang masalah untuk diselesaikan. Contoh dari game ini antara lain: Pipe dream, Hangaroo, Rotation, Catur, permainan kartu dan sebagainya



game “rotation” bertipe board game

## 2. Arcade

Game bertipe arcade merupakan game yang menguci kecepatan tangan dari pemainnya. Pada permainan bertipe arcade, semakin tinggi level permainan, permainan akan berjalan semakin cepat. Contoh dari game bertipe arcade adalah : zuma, feeding frenzy, Pacman, Arkanoid, Pong, Baba ball dan sebagainya.



game “ballistic biscuit” merupakan game bertipe arcade

## 3. Action

Berbeda dengan game bertipe arcade, game bertipe action menjadikan pemain mengendalikan karakter utama dalam game tersebut untuk melakukan beberapa kegiatan (action) seperti melompat, menembak dan sebagainya. Contoh dari game bertipe action antara lain : super mario, mocol, petualangan paddle pop, alloy tease exile dan sebagainya.



game “mocol”, salah satu game bertipe action

## 4. Shooting

Game bertipe shooting sebagian besar menggunakan mouse sebagai alat pengendaliannya. Pada game ini pemain seolah-olah berperan sebagai penembak (first person) atau pemain mengendalikan seorang penembak (third person).

Contoh game bertipe shooting antara lain : Duck hunt, Counter Strike, Onimusha dan sebagainya.



game “terorist” merupakan salah satu game bertipe shooting

#### 5. Fighting

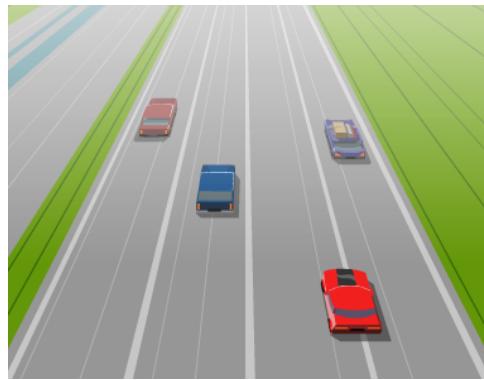
Game bertipe fighting pada dasarnya sama dengan game bertipe action, hanya saja game bertipe fighting pemain mengendalikan sebuah karakter untuk berkelahi dengan karakter lain sampai salah satu karakter kalah. Contoh dari game bertipe fighting antara lain : street fighter, tekken, duel, pencak silat dan sebagainya.



game “super fighter” salah satu game bertipe fighting

#### 6. Racing

Game bertipe racing pada dasarnya adalah sebuah permainan menggerakkan kamera. Pemain diberikan sebuah kendaraan atau sejenisnya untuk menempuh rute tertentu. Contoh dari game bertipe racing antara lain : NFS, Auto bahn, MotoGP, Formula 1 dan sebagainya.



game “auto bahn” salah satu game bertipe racing

## 7. Simulation

Game bertipe simulasi adalah sebuah game yang mensimulasikan suatu kegiatan yang sesungguhnya. Contoh dari game bertipe simulasi antara lain : tycoon, simulator pesawat, burger empire dan sebagainya.



game “burger empire 2” salah satu game bertipe simulation

## 8. Real Time Strategi

Game bertipe RTS memposisikan pemain sebagai seorang pemimpin yang mengatur sesuatu (bisa berupa pasukan, koloni, kerajaan dan sebagainya). Contoh dari game bertipe RTS antara lain: war commander, empire earth, romance of the three kingdom dan sebagainya.



game “war commander” salah satu game bertipe RTS

## 9. Role Playing Game

Pada game bertipe RPG pemain memerankan sebuah karakter dalam game.

Berbeda dengan game bertipe action, pada game RPG hal yang diutamakan adalah cerita dalam game. Selain itu didalam game bertipe RPG biasanya terdapat sub game dengan tipe lain (seperti bertipe fighting, action atau RTS). Contoh game bertipe RPG antara lain : BOBO games, Zelda, Megaman, Final Fantasy dan sebagainya.



game “Smarty” salah satu game bertipe RPG

## 10. MMO

MMO (Masive Multiplayer Online), merupakan game yang dapat dimainkan secara bersama-sama pada internet browser. Hal yang diutamakan dalam sebuah game bertipe MMO adalah kebersamaan dengan pemain lain. Contoh game bertipe MMO antara lain : ragnarok, pangya, gun bound, seal dan sebagainya. Game flash bertipe MMO masih sangat sedikit-hal tersebut dikarenakan untuk membuat game flash MMO diperlukan software lain (server side software) seperti PHP, CGI, Fire Fox, dan sebagainya. Contoh dari game flash bertipe MMO antara lain : Dofus dan battle-on.



game “Dofus” salah satu game bertipe MMO-RPG

### Menulis Storyboard game

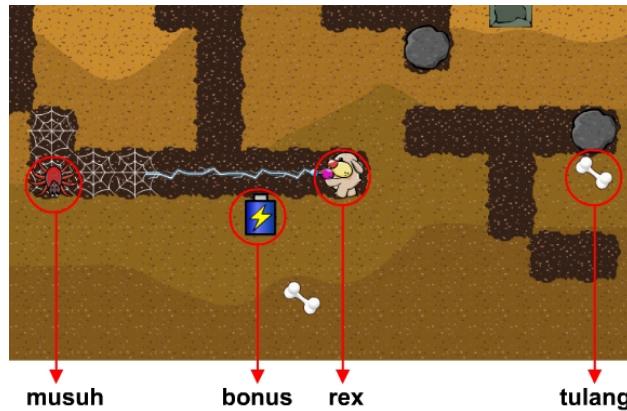
Setelah mendapatkan sebuah ide yang gemilang, akan lebih baik kita segera menuliskannya menjadi sebuah storyboard. Contoh dari sebuah storyboard game adalah sebagai berikut :

Judul game	Dig Dog
Jenis game	Arcade
Sistem kendali	Keyboard - tombol panah dan tombol A dan Z untuk menembak
Karakter utama	Rex si Anjing
Musuh	Mittymouse si tikus, Scorpy si Kalajengking dan Rat si tikus penggali.
Sistem permainan	Pemain diharuskan menembak musuh dan menggali tanah untuk mengambil tulang. Setelah musuh habis dan tulang diambil semua, permainan dilanjutkan dengan level baru dengan musuh yang lebih banyak dan lebih pintar. Sesekali bonus berupa energy, nyawa, bonus poin muncul.

### Memvisualisasikan Ide

Setelah menuliskan storyboard dari game, langkah selanjutnya adalah memvisualisasikan ide tersebut. Visualisasi dapat kita lakukan pada selembar kertas, maupun dengan media komputer (tergantung media mana yang lebih dikuasai).

Contoh dari visualisasi storyboard diatas adalah sebagai berikut:



visualisasi dari sebuah storyboard

### Menggambar Karakter

Setelah ditentukan karakter yang akan dipakai, selanjutnya karakter tersebut dibuat dengan menggunakan drawing tool atau menggambar terlebih dahulu secara manual

pada kertas, kemudian dilakukan proses pewarnaan pada komputer. Proses detail mengenai pembuatan karakter terdapat pada bab selanjutnya.

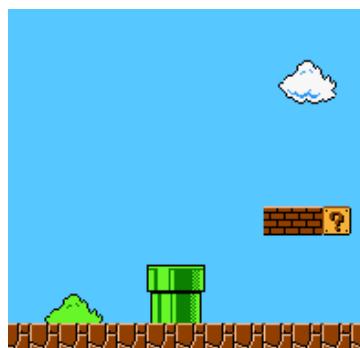
Perhatikan contoh berikut :



sebuah karakter dalam game

### Menggambar Background

Selain karakter dalam sebuah game terdapat background. Teknik penggambaran Background dapat juga di lakukan secara manual maupun dengan menggunakan drawingtool.



contoh background dalam game

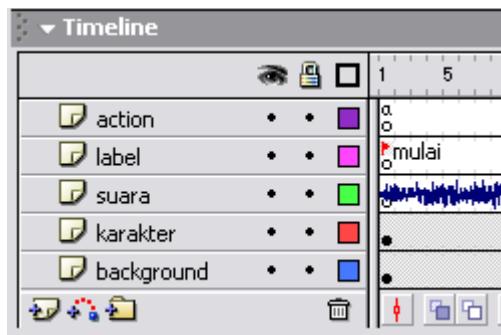
### Programming

Setelah semua gambar siap, bagian yang boleh dikatakan sebagai bagian terumit adalah bagian programing. Untuk mempermudah proses programming sebuah game dalam flash, dapat dilakukan pembagian proses menjadi beberapa bagian yang berurutan sebagai berikut :

1. Menyusun layer dengan sistematika nama yang teratur

Susunan layer yang teratur sangat menunjang kelancaran pembuatan game.

Perhatikan susunan layer standart berikut :



susunan layer standart

## 2. Menata obyek pada stage.

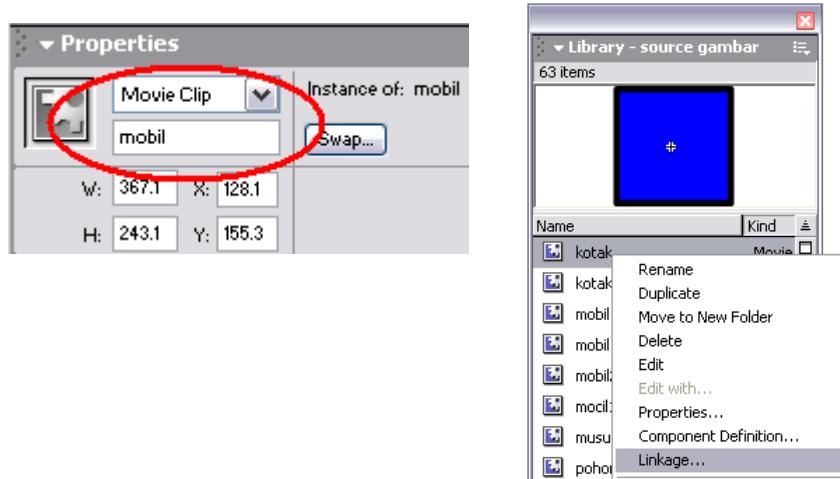
Setelah layer disusun, selanjutnya obyek diatur pada stage. Sebagai catatan, penataan obyek pada stage dapat dilakukan dengan action attachMovie(), sehingga proses penataan obyek pada stage bisa dilompati jika kita menggunakan action tersebut.



menata obyek pada stage

## 3. Menambahkan instance name / linkage.

Jika kita menambahkan obyek pada stage secara manual, langkah selanjutnya adalah menambahkan instance name. Jika kita menambahkan obyek dengan action, langkah selanjutnya adalah menambahkan linkage.



penambahan instance name atau linkage

## 4. Menuliskan action utama.

Yang dimaksud dengan action utama disini adalah action-action penting yang menyusun suatu game seperti menambahkan karakter (jika menggunakan linkage), menggerakkan karakter utama, menggerakkan musuh, dan menghitung tumbukan. Setelah action utama ditambahkan baik pada frame maupun pada movieclip, maka game sudah dapat dijalankan meskipun belum ada detail seperti pengaturan score, penambahan cover dan sebagainya.

#### 5. Menambahkan Sound.

Sound merupakan elemen yang harus diperhatikan dalam game. Cara membangkitkan sound dapat dilakukan secara manual (drag dari library ke stage) atau menggunakan action. Lihat secara mendetail pada bab Sound.

#### 6. Menguji Game

Langkah selanjutnya adalah menguji game. Menguji movie setiap menambahkan action sangat bermanfaat untuk mengetahui suatu kesalahan sejak awal.

#### 7. Menambahkan Action pendukung.

Yang termasuk dalam action pendukung antara lain : penambahan score, sistem bonus, sistem level, menambah AI (kecerdasan buatan musuh) dan sebagainya.

#### 8. Membuat detail game.

Sebagai nilai tambah dalam game adalah kedetailan game. Kedetailan game dapat dicapai dengan membuat obyek-obyek tambahan yang tidak mengganggu kecepatan game.

#### 9. Menguji game dan mencari bug

Tahapan selanjutnya adalah mencari segala kemungkinan kesalahan dalam game. Untuk mencari bug, cara yang paling mudah adalah menguji game berkali-kali dan membiarkan orang lain untuk menguji prototype game yang kita buat.

#### 10. Proses publishing game.

Setelah segala kesalahan diperbaiki, tahapan selanjutnya adalah mempublis game. Lihat secara mendetail proses publising game pada bab publishing.

#### FAQ :

Apakah kita boleh memiliki sebuah ide game yang hebat?

Ide yang hebat biasanya menghasilkan sebuah karya yang hebat pula. Akan tetapi ketika kita baru belajar mengenai proses pembuatan game, simpan terlebih dahulu ide hebat tersebut menjadi sebuah story board, dan kerjakan game-game bertipe sederhana terlebih dahulu.

Sebuah game yang kompleks akan memakan waktu yang banyak pada saat membuat tampilan game (grafis), programming dan testing. Sebagai contoh game “Dofus” dikembangkan oleh satu tim yang berisi lebih dari 20 programmer dan 10 game artis selama 6 bulan dan ditambah 3 bulan testing. Tentunya dengan waktu yang selama itu kita dapat menghasilkan banyak game sederhana sambil memperdalam ilmu pembuatan game.

Untuk memvisualisasikan sebuah ide yang hebat kita membutuhkan banyak hal, yaitu:

1. waktu yang banyak
2. tim yang solid, tim meliputi programmer, game artis dan game tester. Sebuah game yang hebat sulit dikerjakan oleh satu orang saja. Sebagai contoh game Bobo karya penulis membutuhkan waktu 3 bulan untuk menyelesaikan prototipenya, dan tambahan 1 bulan untuk menyusun ceritanya.
3. modal yang besar. Dengan banyaknya waktu yang dipakai dan jumlah tim, secara otomatis modal yang diperlukan dalam membuat game kompleks menjadi besar pula. Untuk sebuah game flash kompleks, developer luar negeri mengeluarkan anggaran sekitar \$600 sampai \$20.000 (nilai tersebut masih dinilai sangat kecil bila dibandingkan dengan pembuatan game kompleks dengan bahasa pemrograman lain semacam GTA, NFS dan game besar lainnya yang menghabiskan dana diatas 500.000 US\$).
4. tim pemasaran yang kuat. Jika kita mendalami bidang game, tentunya dibutuhkan sebuah tim yang memasarkan game-game yang sudah dibuat. Selain untuk mendapatkan keuntungan, tentusaja pemasaran juga digunakan untuk mengembalikan modal yang kita pakai dalam membuat game tersebut.

Diluar itu semua, jika anda memiliki ide yang benar-benar baru dan hebat, segera wujudkan ide tersebut dalam sebuah storyboard plus visualisasinya. Anda dapat menjual ide tersebut kepada developer asing atau meminta sponsor untuk mewujudkan ide anda tersebut.

## **Membuat Karakter**

Salah satu elemen terpenting dalam game adalah game grafis, dan salah satu didalam game grafis tersebut adalah karakter (tokoh utama) game. Karakter dalam game memiliki pengaruh yang sangat kuat terhadap game, dimana dengan karakter tersebut sebuah game dapat menjadi terkenal atau tidak. Contoh karakter yang menjadikan suatu game terkenal antara lain : Super Mario pada semua seri game Super Mario, Sonic, Ray-man, dan beberapa karakter pada game RPG terkenal seperti Final Fantasy, Shining Force, dan sebagainya.

Untuk membuat karakter yang menarik kita perlu mengetahui seni menggambar, baik itu menggambar secara manual maupun menggambar secara digital-pada dasarnya kedua cara tersebut sama hanya saja media yang digunakan berbeda- dan pengetahuan terhadap suatu gaya desain.

Pembagian karakter berdasarkan gaya desain menurut penulis dalam buku ini adalah:

### 1. Gaya Realis Detail

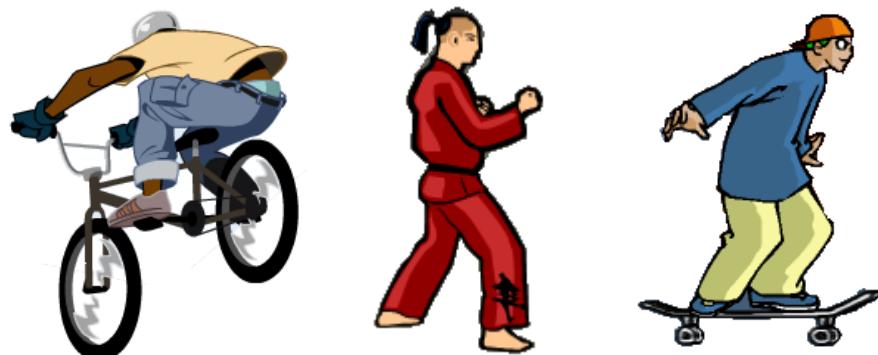
Gaya ini biasanya ditangani oleh orang profesional yang telah mahir mengolah gambar. Selain itu untuk menangani gaya realis yang detail, Flash mengalami kesulitan- karena flash tidak dapat memberikan efek shading (bayangan) dan lighting (pencahayaan)- meskipun dalam Flash 8 terdapat efek blur dan shadow, tetapi efek tersebut tidak optimal untuk membuat gaya realis dan detail, sehingga dibutuhkan software lain semacam Adobe Photoshop, Firework atau sejenisnya.



Game Street fighter versi Flash salah satu game bergaya Realis Detail

### 2. Gaya Realis Sederhana

Gaya ini merupakan penyederhanaan dari gaya realis sederhana. Untuk membuatnya dapat dilakukan proses tracing atau menjiplak sebuah gambar realis dan mengurangi detailnya. Dalam game flash gaya ini lebih banyak dipakai, karena selain lebih mudah membuatnya, karakter yang tidak terlalu detail membuat movie berjalan normal.



karakter bergaya Realis Sederhana

### 3. Gaya Kartun

Gaya ini merupakan penyederhanaan dan perubahan bentuk dari gaya realis sederhana. Gaya kartun tetap mempertahankan suatu bentuk standard, sebagai contoh manusia memiliki kepala, badan, tangan dan kaki. Bentuk tersebut tetap dipertahankan namun dengan proporsi dan penggayaan yang berbeda.



karakter bergaya kartun

### 4. Gaya Imajinasi

Gaya ini bersifat bebas, tidak terikat dengan bentuk realis dan merupakan hasil dari imajinasi pencipta game. Beberapa game flash menggunakan karakter bergaya imajinasi, karena selain lebih mudah membuatnya, karakter bergaya imajinasi juga dapat mempermudah dalam membentuk sebuah image (trademark) kepada public.



karakter bergaya imajinasi

### Membuat Karakter dengan Menggunakan Drawing Tool

Membuat karakter dapat dilakukan secara langsung dalam flash dengan menggunakan Drawing tool. Menggambar dengan menggunakan drawing tool memiliki kelemahan yaitu untuk menghasilkan suatu gambar yang detail, kita membutuhkan waktu yang lama dan tingkat ketelitian yang tinggi.

Perhatikan contoh menggambar sebuah karakter mobil berikut :

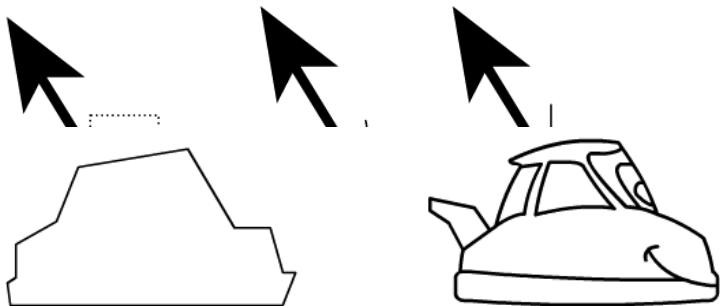
1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
2. Klik **rectangle tool**, atur **stroke color**: hitam, **fill color** : tanpa warna (no color) dan ukuran garis 1 poin.



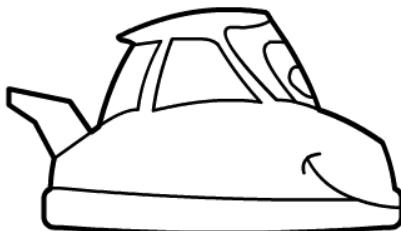
mengeset rectangle tool

3. Selanjutnya buatlah gambar **kotak** pada stage.
4. Bentuk kotak tersebut menjadi bentuk dasar sebuah mobil. Perhatikan teknik mengubah bentuk dengan cepat :
  - Dekatkan kursor mouse ke garis, maka kursor mouse akan berubah bentuk menjadi bentuk lengkung. Klik dan drag untuk membuat garis menjadi melengkung.
  - Dekatkan kursor ke sudut, maka kursor akan menjadi bentuk sudut. Klik dan drag untuk mengubah posisi sudut.
  - Dekatkan kursor ke garis dan tahan tombol Ctrl, kursor berbentuk lengkung. Klik dan drag untuk menambah sudut.

bentuk kursor mouse pada saat arrow tool aktif



5. Selanjutnya edit kembali dan tambahkan beberapa garis dengan **menggunakan line tool** menjadi bentuk seperti yang tampak pada gambar.  
bentuk kotak setelah diedit
6. Seleksi garis yang berada ditepi luar mobil, kemudian tambah ketebalan garis menjadi 5 poin (atur sesuai dengan keinginan anda). Penambahan ketebalan garis luar akan menambah kuat karakter gambar.



gambar mobil setelah ditambah ketebalan garis luar

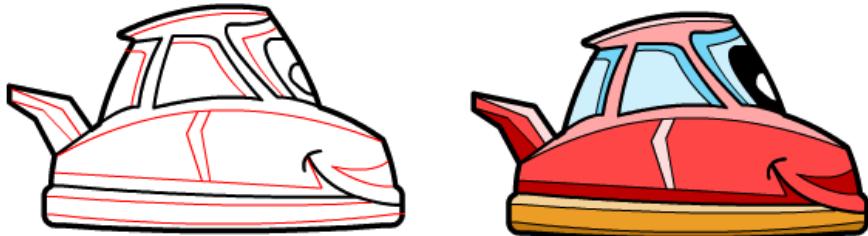
7. Seleksi seluruh garis kemudian convert menjadi fill dengan memilih menu **modify>shape>convert line to fill**. Peng-convert-an garis menjadi fill dimaksudkan untuk menjaga ukuran ketebalan garis agar mengikuti pembesaran atau pengecilan dengan menggunakan transform tool.

Perhatikan:

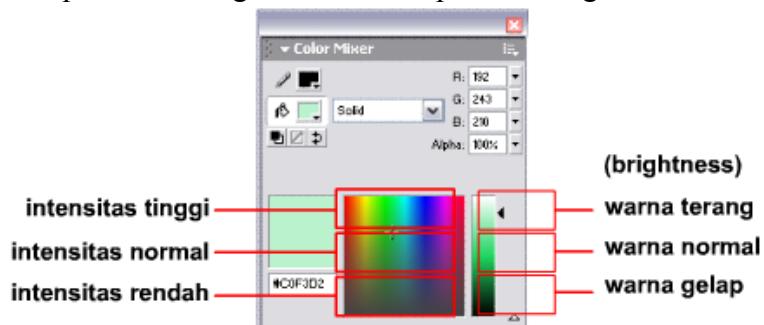
menconvert garis menjadi fill terkadang tidak berjalan sempurna (ada garis yang hilang). Apabila hal tersebut terjadi, tekan tombol **undo**, kemudian ubahlah ukuran ketebalan garis. Naikkan atau turunkan (contoh : bila garis dengan ketebalan 1 pixel hilang saat diconvert menjadi fill ubahlah ukurannya menjadi 1.1 – 2 pixel)

8. Buatlah **garis-garis bantu** dengan warna lain, yang nantinya digunakan untuk memberikan efek gelap terang pada gambar.

9. Selanjutnya berikan warna pada gambar tersebut. Perhatikan penggunaan warna gelap dan warna terang. Penambahan gelap dan terang tersebut dapat menambah elemen estetik dalam sebuah gambar.



penambahan garis bantu dan pewarnaan gambar



warna pada panel color mixer

10. Hapus garis bantu, selanjutnya seleksi seluruh gambar dan convert menjadi **movieclip** dengan **nama mobil**.



bentuk mobil setelah garis bantu dihapus

11. Tahapan selanjutnya adalah menambahkan **roda**. Lihat proses pembuatan roda pada bab sebelumnya (bab menggerakan movieclip).
12. Lanjutkan pembuatan **mobil**. Double klik **mobil** untuk mengeditnya.
13. Tambahkan sebuah layer di atas layer 1 dan ubah namanya menjadi **layer bumper**.
14. Klik **frame 1 layer bumper**, kemudian dengan menggunakan **oval tool dan line tool** buatlah gambar tempat roda mobil, seperti yang tampak pada gambar.
15. Buatlah garis bantu untuk proses pewarnaan dan warnailah dengan menambahkan warna gelap dan terang.



menggambar dan mewarnai bumper

16. Tambahkan sebuah layer baru diatas layer 1 dan dibawah layer bumper.

Kemudian drag movieclip **roda** dari library ke stage sebanyak dua kali.



posisi layer roda dan peletakan roda

17. Keluar dari mode edit dan movieclip mobil siap dipakai.

### **Mengolah Gambar Manual dengan Flash**

Alternatif lain dalam membuat karakter-selain menggunakan drawing tool adalah dengan menggambar secara manual pada selembar kertas. Bagi beberapa orang menggambar di kertas jauh lebih cepat dari pada menggambar menggunakan drawing tool. Meskipun demikian, pengolahan gambar selanjutnya (proses pewarnaan) selalu melibatkan software grafis seperti photoshop dan software sejenis lainnya.

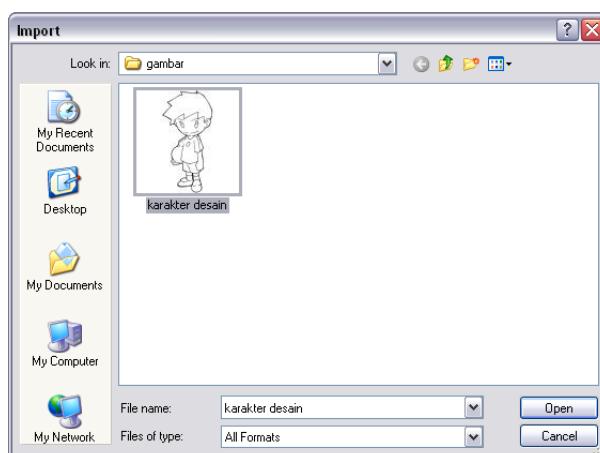
Dalam flash kita juga dapat mengolah sebuah gambar manual yang telah kita buat sebelumnya. Perhatikan proses pengolahan gambar berikut:

1. Buatlah sebuah gambar karakter pada kertas dengan pensil, kemudian lakukan poses penintaan, dan hapus bekas pensil yang masih tampak. Gambar berikut adalah gambar desain karakter pada game “**Mini Soccer**”.



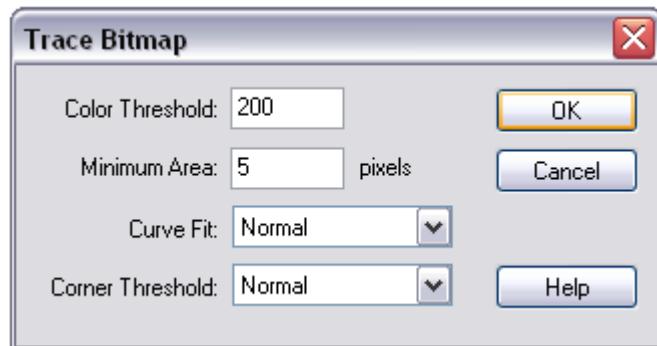
contoh karakter yang digambar secara manual

2. Scan gambar tersebut menjadi sebuah file berformat **jpeg** atau **bmp**.
3. Buka program flash, kemudian buat sebuah file dengan ukuran 800 x 600 px dan 12 fps.
4. Import gambar yang sudah discan sebelumnya dengan memilih menu **file>import...** . Apabila anda belum membuat gambar karakter, import file gambar “**karakter desain.jpg**” pada folder gambar pada CD tutorial.



panel import file

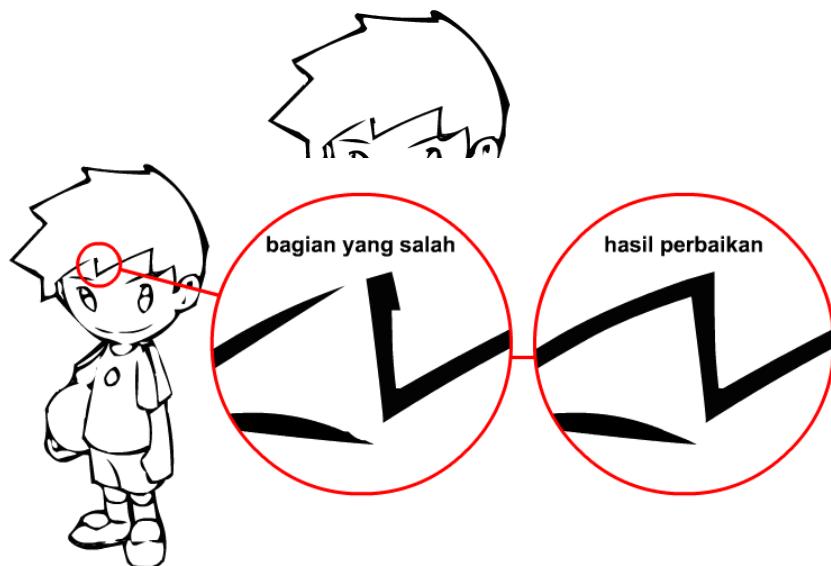
5. Setelah gambar tampak pada stage, klik gambar dan pilih menu **modify > trace bitmap** (pada Flash MX) atau **modify>bitmap>trace bitmap** (pada Flash 2004 dan Flash 8).
6. Masukan bilangan **200** pada **Color Threshold** (semakin kecil bilangan yang dimasukan, gambar akan di trace semakin detail). Masukan **5** pada **Minimum Area** dan pilih option **normal** pada **Curve Fit** dan **Corner Threshold**.



Trace bitmap

7. Klik **OK**, maka gambar akan terpisah berdasarkan warnanya.
8. Seleksi gambar yang diinginkan lalu tekan **Ctrl + X (cut)**, kemudian hapus semua yang ada pada stage (biasanya berwarna putih atau abu-abu), kemudian tekan **Ctrl + V (paste)**, maka hanya gambar yang diinginkan saja yang muncul. Seleksi gambar dan ubah warnanya menjadi hitam.

hasil dari trace bitmap



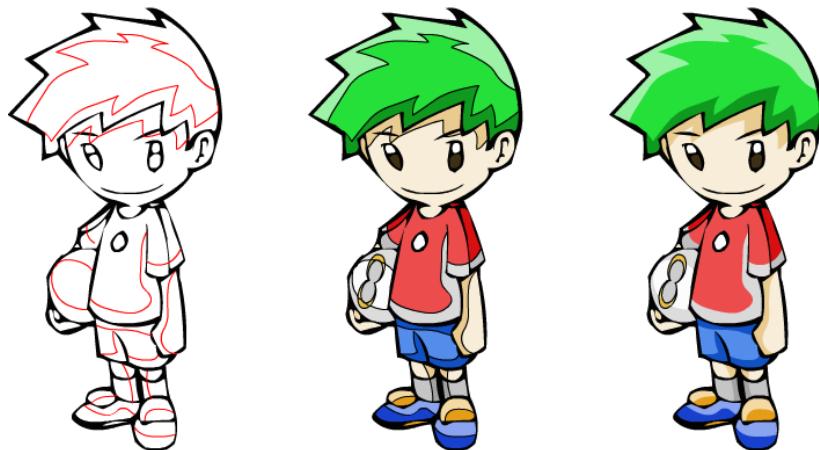
9. Perhatikan bahwa terdapat banyak kekurangan yang harus diperbaiki pada hasil trace tersebut. Perbaikilah dengan menggunakan **arrow tool**.

memperbaiki kesalahan hasil tracing

10. Setelah seluruh proses perbaikan selesai buatlah garis bantu dengan warna lain, yang nantinya digunakan untuk memberikan efek gelap terang pada gambar.

11. Selanjutnya berikan warna pada gambar tersebut. Perhatikan penggunaan warna gelap dan warna terang.

teknik pewarnaan karakter



12. Hapus seluruh garis bantu, seleksi gambar kemudian convert menjadi movieclip dengan nama “**karakter 1**”.
13. Simpan file dan movieclip siap dipakai.

### **Mengimpor File Siap Pakai**

Selain menggunakan drawing tool dan teknik tracing, kita juga dapat mengolah gambar dengan software lain seperti Adobe Photoshop, Image Ready, Fire work dan sebagainya. Format file yang paling sesuai untuk file gambar siap pakai adalah file bertipe **PNG** (baca: ping). File bertipe PNG memiliki keunggulan berupa transparency yang tidak dimiliki oleh file lain. (file bertipe GIF juga memiliki transparansi, tetapi file GIF memiliki kualitas yang kurang bagus, sehingga jarang dipakai).

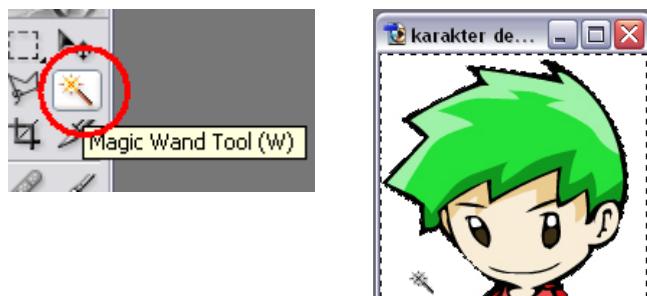
Untuk membuat file bertipe PNG pada program Adobe Photoshop 7 (catatan : pada saat buku ini ditulis program Photoshop terbaru adalah Photoshop CS 2, meskipun demikian proses pembuatan tetap sama), perhatikan contoh berikut:

1. Buatlah sebuah gambar secara manual pada kertas, tinta dan scan menjadi gambar bertipe JPEG. Apabila anda belum membuatnya, buka file gambar “**Karakter Desain Warna.JPG**” pada CD tutorial.
2. Apabila gambar belum diwarnai, tambahkan warna dan efek dengan menggunakan tool pada program photoshop.



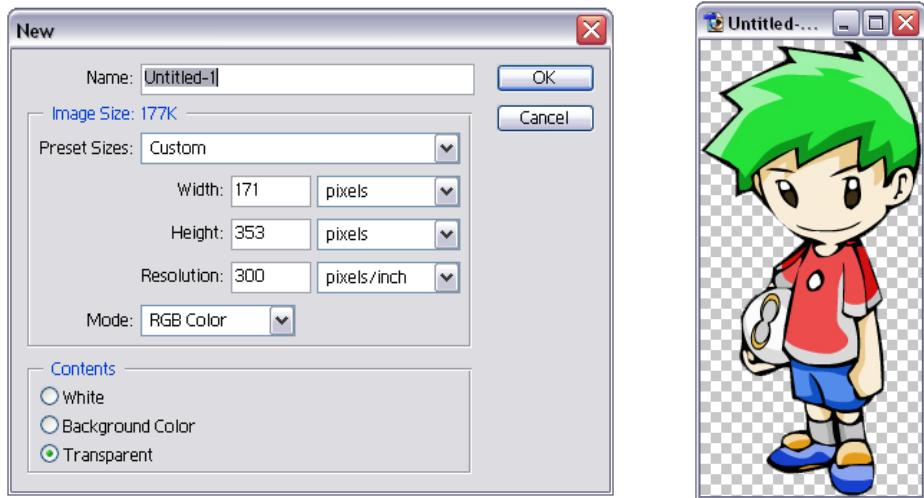
tool panel pada photoshop 7 dan hasil pewarnaannya

3. Perhatikan bahwa terdapat background berwarna putih pada sekeliling gambar. Background tersebut harus dihilangkan agar saat dipakai dalam movie flash warna putih tidak menutupi obyek yang ada di belakangnya.
4. Untuk menghilangkan warna putih di sekeliling gambar, gunakan **magic wand tool**, dan klik pada area yang berwarna putih. Seleksi (garis marque) berupa garis putus-putus akan terbentuk.



magic wand tool dan bentuk seleksi

5. Pilih menu **select>inverse** atau tekan tombol **Ctrl+Shift+I**, untuk menyeleksi bagian yang dipakai. Kemudian tekan tombol **Ctrl+C (copy)**.
6. Buat file baru dengan menekan menu **file>new**. Biarkan ukuran yang ada dan pilih **transparent** pada option **content**. Tekan **OK**, maka panel file baru akan terbentuk. Tekan **Ctrl+V** untuk mem-paste gambar yang sudah kita copy sebelumnya.



membuat file gambar dengan background transparan

7. kemudian simpan file dengan memilih menu file>save as. Ketikan nama file “**karakter desain**” dan pilihlah **PNG** pada tipe file. Klik **OK**, pilihlah **none** pada pilihan **PNG option**.



PNG option

8. Untuk memakai gambar tersebut, buka program flash dan pilih menu **file>import**.

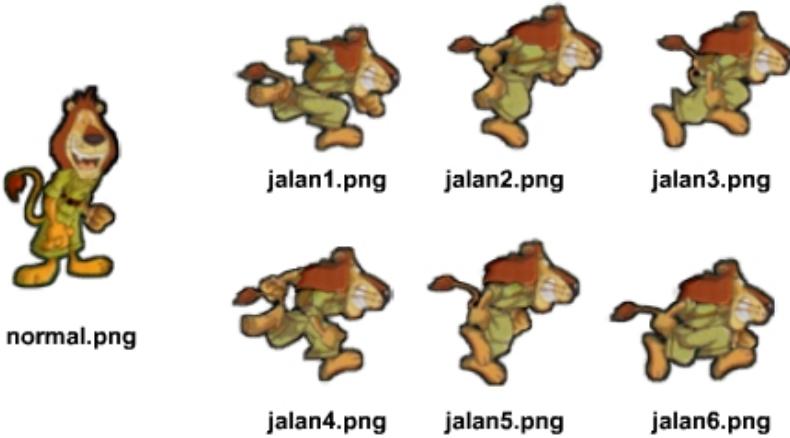
**Permasalahan :**

Buatlah sebuah karakter yang dapat berjalan atau berlari ke kanan dan kekiri, ketika tombol panah pada keyboard ditekan.

Untuk menyelesaikan permasalahan tersebut, kita membutuhkan sedikitnya 7 gambar karakter, yaitu 1 gambar karakter pada posisi **normal**, dan 6 gambar animasi frame-by-frame karakter yang sedang **berjalan** atau **berlari**. Perhatikan proses penyelesaian masalah tersebut :

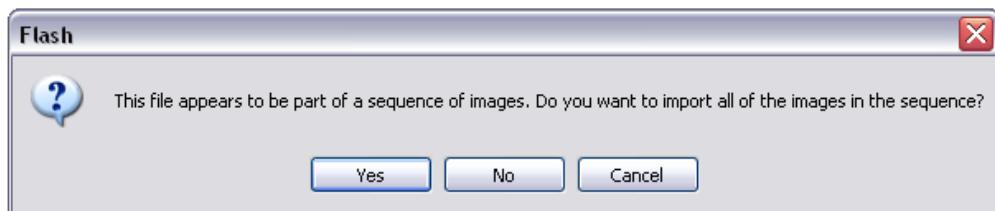
1. Buatlah gambar-gambar yang dibutuhkan untuk menyelesaikan masalah tersebut. Kemudian olah menjadi gambar bertipe **PNG** dengan background **transparan**. Perhatikan aturan pembuatan gambar karakter berlari secara frame-by-frame(direkomendasikan anda memahami 12 prinsip animasi, agar animasi frame by frame yang dihasilkan nantinya tampak menarik dan logis).

Untuk memberi nama file animasi secara frame by frame, gunakan angka yang berurutan. Contoh nama file yang baik untuk animasi frame-by-frame : **lari1.png**, **lari2.png**,..., **lari6.png**. Pemberian nama yang berurutan akan mempermudah proses pengolahan animasi pada Flash.



contoh file siap pakai pada game petualangan Paddle Pop  
(trademark karakter milik Walls : PT Unilever Indonesia)

2. Setelah file yang akan dipakai siap, buka program flash dan buatlah sebuah file dengan ukuran 800 x 600 px, 12 fps. Anda bisa menggunakan file “**normal.png**”, “**car jalan1.png**”, ..., “**car jalan6.png**” pada CD tutorial.
3. Klik menu **insert>new symbol..**, kemudian ketikan “**karakter**” pada nama dan pilih **movieclip** pada behaviour.
4. Klik **frame 1 layer 1**, kemudian pilih menu **file>import**. Pilih file ”**normal.png**” (posisi karakter berdiri) dan tekan **OK**. Maka pojok kiri atas gambar akan muncul tepat ditengah stage.
5. Klik **frame 2**, kemudian masukkan **blank keyframe** dengan menekan tombol **F7**. Klik menu **file>import..** Pilih file “**car jalan1.png**”, dan tekan **OK**. Jika anda menuliskan nama file dengan benar (menggunakan angka seperti pada contoh), sebuah dialog panel akan muncul. Klik **OK**, maka akan terbentuk animasi frame by frame secara otomatis.



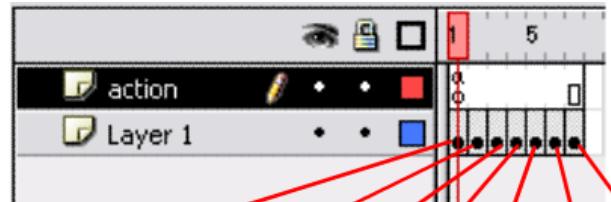
dialog panel ketika mengimpor file berurutan

6. Tambahkan sebuah layer baru diatas layer 1, dan ubah namanya menjadi **layer action**.
7. Klik **frame 1 layer action**, buka panel action, kemudian tambahkan action :

```
stop();
```

susunan layer dan obyek pada movieclip **karakter**

8. Keluar dari
9. Buka pane
10. Klik movie



stage.  
ript berikut :



```
onClipEvent (load) {
    langkah = 1;
    skala = 100;
    jaraklangkah = 20;
}

onClipEvent (enterFrame) {
    //langkah kekanan
    if (Key.isDown(Key.RIGHT)) {
        _xscale = skala;
        _yscale = skala;
        langkah++;
        _x += jaraklangkah;
        if (langkah>6) {
            langkah = 1;
        }
    }
}
```

```

        gotoAndStop(langkah+1);

    }

    //langkah kekiri
    if (Key.isDown(Key.LEFT)) {
        _xscale = -skala;
        _yscale = skala;
        langkah++;
        _x -= jaraklangkah;
        if (langkah>6) {
            langkah = 1;
        }
        gotoAndStop(langkah+1);
    }

    onClipEvent (keyUp) {
        //posisi kembali ke normal
        gotoAndStop(1);
    }
}

```

11. Simpan file dan jalankan movie.

Penjelasan program :

1. Pada movie event **load**, variabel yang akan dipakai diset dan ditentukan nilainya.
2. Ketika panah kanan ditekan (**if (Key.isDown(Key.RIGHT)) { }**), **skala** movieclip dijadikan positif (**\_xscale = skala;**), dan variabel **langkah** ditambah satu satuan (**langkah++**). variabel ini digunakan untuk menentukan langkah karakter. Selanjutnya posisi movieclip digeser ke kanan sejauh variabel **jaraklangkah** (**\_x += jaraklangkah**). dan frame aktif dipindah ke posisi yang sesuai dengan variabel **langkah** (**gotoAndStop(langkah+1)**), penambahan angka 1 dikarenakan animasi berlari dimulai dari frame 2.
3. Untuk mengembalikan posisi ke normal, saat tidak ada tombol ditekan, digunakan action **gotoAndStop(1);** pada movie event **keyUp**.

Catatan :

Ketika menjalankan movie tersebut, anda akan melihat sebuah kejanggalan (bug)

ketika anda menekan panah kanan dan diikuti dengan panah kiri. Hal tersebut dikarenakan posisi registration movieclip **karakter** tidak berada di tengah tetapi di pojok kiri atas. Untuk memperbaikinya, edit movieclip **karakter** dan geser posisi gambar di tiap-tiap frame ke tengah stage.

memperbaiki bug pada frame 1 movieclip karakter.

titik registrasi



sebelum diedit



sesudah diedit

## Background

Selain karakter, dalam game juga dibutuhkan sebuah background atau latar belakang. Membuat background dalam Flash dapat dilakukan dengan beberapa cara seperti pada pembuatan karakter. Akan tetapi sebelum menginjak pada bagian pembuatan

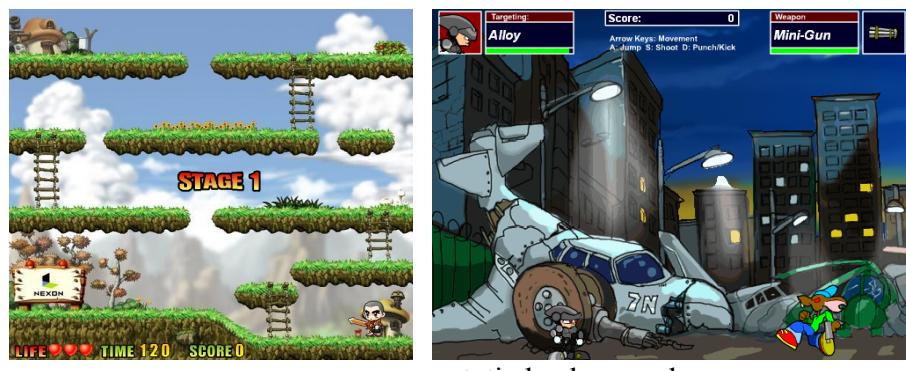
background, akan lebih baik jika kita mengenal terlebih dahulu jenis-jenis background dalam game.

Menurut saya, dalam buku ini background game flash dibedakan berdasarkan gerakkannya menjadi 2 jenis yaitu:

1. Static background.

Static background atau background diam sering kita jumpai pada game flash.

Selain mudah membuatnya, pada static background kita tidak membutuhkan script khusus untuk menggerakkannya. Perhatikan contoh dari game yang menggunakan static background berikut :



penggunaan static background

2. Scrolling Background.

Berbeda dengan static background yang diam, background bertipe scrolling background bergerak dari kanan kekiri, atas kebawah, atau bergerak bebas.

Tingkat kesulitan dari pembuatan scrolling background adalah pada pengelolahan script, karena tanpa script yang baik, movie dapat berjalan lambat. Perhatikan contoh dari game yang menggunakan scrolling background berikut :



Sedangkan background berdasarkan sudut pandangnya, saya membedakan menjadi 3 yaitu:

### 1. Pandangan datar

Pandangan datar hampir dapat kita temui pada sebagian besar game flash. Hal tersebut dikarenakan pembuatan background dengan pandangan datar jauh lebih mudah dibandingkan dengan dua pandangan yang lain. Perhatikan contoh dari game yang menggunakan background dengan pandangan datar berikut :



Background dengan pandangan datar

### 2. Pandangan isometri

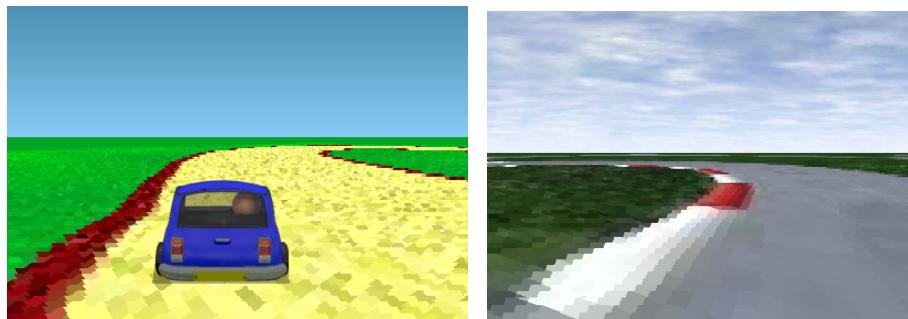
Yang dimaksud dengan pandangan isometri adalah pandangan burung (tampak atas) dengan sudut 45 derajat. Keunggulan dari pandangan isometri adalah dapat memberikan kesan kedalaman suatu obyek dalam game. . Perhatikan contoh dari game yang menggunakan background dengan pandangan isometri berikut :



game dengan pandangan isometri

### 3. Pandangan 3 Dimensi.

Dalam flash membuat background 3 dimensi (memiliki dimensi panjang lebar dan tinggi) sangat rumit, akan tetapi masih mungkin dilakukan untuk menghasilkan sebuah efek 3 Dimensi. Perhatikan contoh dari game yang menggunakan efek 3 dimensi pada background berikut :



efek 3D background

### **Static Background dengan Still Image (gambar diam)**

Untuk membuat sebuah static background saya membagi lagi menjadi 2 cara yaitu dengan menggunakan **still image (gambar diam)** dan menggunakan **sistem tiling (pengubinan)**. Membuat backgroud dengan teknik still image hampir sama dengan membuat karakter, yaitu bisa menggunakan drawing tool, teknik tracing atau mengimpor gambar jadi.

Berikut adalah contoh membuat static background dengan menggunakan **still image**:

1. Buatlah sebuah gambar background untuk sebuah game bertipe shooting, seperti pada gambar berikut (buka file “**background1.png**” ada CD tutorial), atau gambarlah sesuai dengan kreativitas anda :



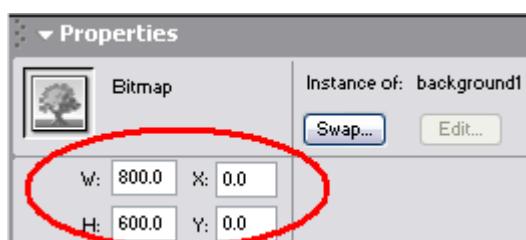
penggunaan gambar jadi sebagai background

2. Buka file latihan menggerakkan movieclip dengan mouse:tracking yang kita buat pada bab sebelumnya (buka file “**program34-mc\_gerakan\_mouse\_tracking.fla**“ pada CD tutorial), kemudian buka library. Kita akan menggunakan movieclip **burung**.
3. Setelah gambar siap, buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
4. Buatlah 2 buah layer baru dan ubah nama layer masing-masing **menjadi layer background, layer obyek dan layer action**.



susunan layer

5. Klik **frame 1 layer background**, kemudian klik menu **file>import**. Importlah gambar background yang sudah kita buat. Setelah gambar tampak pada stage, atur posisi dan ukuran gambar sampai sama dengan ukuran stage.



properties bitmap yang dipakai sebagai background

6. Klik tombol **lock** pada **layer background**, karena kita sudah tidak mengeditnya lagi.



lock layer

7. Klik **frame 1 layer obyek**, kemudian drag movieclip **burung** dari library file latihan mouse tracking yang sudah kita buka sebelumnya. Letakkan di sebelah kiri luar stage. Atur posisinya dan arah gerakkannya. Apabila burung menghadap ke kiri, balik posisi burung dengan menekan menu **modify>transform>flip horizontal**.



posisi burung terhadap stage

8. Klik movieclip **burung**, kemudian buka panel properties dan ketikan “**burung\_1**” pada instance name. Selanjutnya buka panel action dan ketikan script berikut :

```
onClipEvent (load) {  
    _y = 100+random(400);  
    xawal = _x;  
    kecepatan = 10;  
    naik = 0;  
    keluar = 0;  
    _visible = 0;  
}  
  
onClipEvent (enterFrame) {  
    //burung belum keluar  
    if (keluar == 0 and random(20) == 5) {  
        keluar = 1;  
        _visible = 1;  
    }  
    //burung keluar  
    if (keluar == 1) {
```

```

_x += kecepatan;
if (random(10) == 3) {
    //gerakan acak naik
    naik = 1;
} else if (random(10) == 4) {
    //gerakan acak turun
    naik = 2;
} else if (random(10) == 5) {
    // gerakan lurus
    naik = 0;
}
//menggerakkan naik.
if (naik == 1 and _y>50) {
    _y -= 5;
} else {
    //menggerakkan turun.
}
if (naik == 2 and _y<550) {
    _y += 5;
}
//keluar dari stage, maka kembalikan lagi ke posisi awal
if (_x>800) {
    _x = xawal;
    _y = 100+random(400);
    keluar = 0;
    _visible = 0;
}
}

```

9. Jalankan movie, maka akan didapati burung keluar secara acak dan bergerak secara acak. Tutup movie dan kembali ke stage utama.
10. Klik frame 1 layer obyek, buatlah sebuah gambar target sasaran seperti pada gambar berikut :



gambar sasaran pengganti kursor mouse

11. Seleksi gambar tersebut, kemudian convert menjadi **movieclip** dengan nama “**kursor**”.
12. Klik “**kursor**”, buka panel properties dan ketikan “**kursor**” pada instance name.

Buka panel action dan ketikan script :

```
onClipEvent (load) {  
    Mouse.hide();  
    startDrag(this, true);  
}  
  
onClipEvent (enterFrame) {  
    startDrag(this, true);  
}
```

13. Jalankan movie, anda akan mendapati kursor mouse sudah berubah bentuk. Akan tetapi burung belum dapat ditembak. Untuk dapat menembak burung, tutup movie dan lanjutkan proses pembuatan game.
14. Klik movieclip **burung** dan buka panel action.
15. Letakkan kursor mouse pada baris terakhir (dibelakang tanda “}” terakhir) kemudian tekan enter. Kemudian tambahkan action berikut :

```
onClipEvent (mouseDown) {  
    if (hitTest(_root.kursor)) {  
        _x = xawal;  
        _y = 100+random(400);  
        keluar = 0;  
        _visible = 0;  
    }  
}
```

16. Simpan dan jalankan movie. Untuk menambah jumlah burung, anda dapat melakukan **copy-paste** movieclip **burung** kemudian ubah instance name-nya dan ubah variabel **kecepatan** untuk menambah tingkat kesulitan.

### **Static Background dengan Sistem Tiling (pengubinan)**

Teknik pengubinan pada dasarnya sudah kita pelajari sebelumnya pada latihan penggunaan variabel bertipe array dan pemakaian action for bertingkat. Keuntungan penggunaan sistem tiling adalah kita dapat membuat background dengan berbagai alternatif hanya dengan mengubah bentuk data-nya saja. Perhatikan gambar dari game **Dig Dog** buatan saya berikut ini:



beberapa level pada game Dig Dog

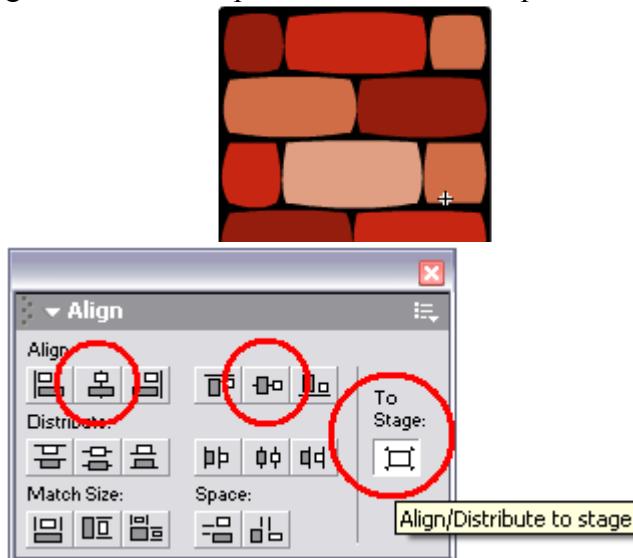
Pada game dig dog tersebut, inti permainan pada dasarnya selalu sama dari level ke level yaitu karakter utama yang diharuskan untuk menggali tulang dan mengalahkan musuh yang ada. Untuk membuat arena permainan (dalam hal ini adalah background) yang berbeda-beda, saya menggunakan sistem pengubinan dengan array 2 dimensi dan for bertingkat.

Perhatikan contoh berikut untuk memahami pembuatan background dengan sistem pengubinan:

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
2. Buatlah layer baru dan ubah nama masing-masing layer menjadi **layer background** dan **layer action**.
3. Buatlah sebuah symbol baru dengan memilih menu **insert>new symbol**. Ketikan “**ubin**” pada nama dan pilih **movieclip** pada behaviour.
4. Pada mode edit movieclip **ubin**, klik **frame 2** dan masukan **keyframe**.
5. Buatlah sebuah gambar balok (batu bata) berukuran 50 x 50 pixel seperti pada gambar berikut:

6.

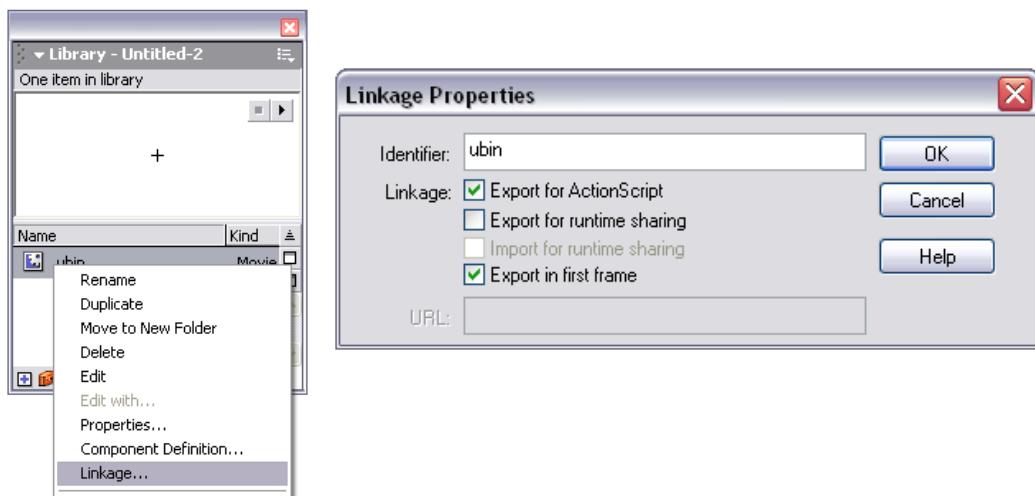
gambar batu bata pada frame 2 movieclip ubin



7. Seleksi gambar pada frame 2, kemudian buka panel **align** dengan memilih menu **window>align**. Atifkan tombol **Align/Distribute to Stage**, kemudian klik tombol **align vertical center** dan **align horisontal center** untuk memindah posisi gambar tepat pada tengah titik registrasi.

panel align

8. Keluar dari mode edit symbol dengan menekan tombol Ctrl+E.
9. Klik **frame 1 layer background**, kemudian buatlah gambar kotak dengan warna biru (#00CCFF) memenuhi stage.
10. Buka panel library. Klik kanan movieclip ubin dan tambahkan **linkage**. Pilih **export for actionscript** dan ketikan **ubin** pada identifier.



menambahkan linkage movieclip ubin

11. Klik **frame 1 layer action**, kemudian buka panel action dan ketikkan script berikut:

```
pola = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0],  
        [0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],  
        [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
        [0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0],  
        [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],  
        [0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0],  
        [0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]];  
  
function gambarPola(dataPola, xawal, yawal) {  
    kedalaman = 0;  
    lebar = dataPola[1].length;  
    tinggi = dataPola.length;  
    for (var i = 0; i<lebar; i++) {  
        for (var j = 0; j<tinggi; j++) {  
            attachMovie("ubin", "ubin"+i+"_"+j, kedalaman++,  
                       {_x:xawal+(i*50), _y:yawal+(j*50)});  
            this["ubin"+i+"_"+j].gotoAndStop(dataPola[j][i]+1);  
        }  
    }  
}  
  
gambarPola(pola, 25, 25);
```

12. Jalankan movie, anda akan mendapati background dengan pola yang bisa diubah dengan cepat.

Penjelasan program sama dengan penjelasan pada program array dan for bertingkat.

Catatan :

Untuk memahami lebih lanjut mengenai pembuatan static background dengan sistem

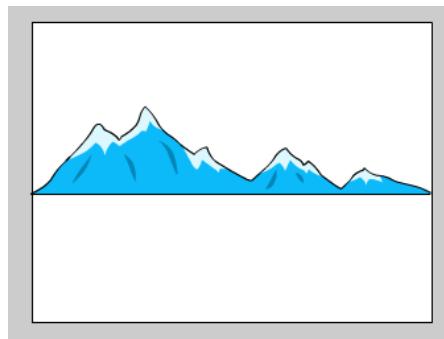
tiling (pengubinan), bacalah buku “Seri Pemrograman Game Flash : Tiling Game”.

### Scrolling Background dengan Motion Tween

Sebelum membuat sebuah scrollin background, harus kita pahami bahwa movie akan berjalan lambat jika kita menggerakkan banyak obyek pada waktu yang bersamaan. Untuk itu kita harus memahami teknik optimalisasi dalam membuat background bergerak agar game kita nantinya tidak berjalan lambat.

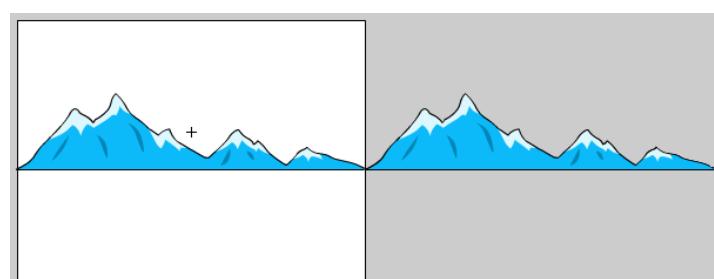
Untuk membuat sebuah scrolling background saya membagi lagi menjadi 3 cara yaitu dengan menggunakan **motion tween**, **action** dan menggunakan **sistem tiling (pengubinan)**. Perhatikan membuat background dengan teknik motion tween berikut:

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
2. Buat sebuah gambar gunung yang nantinya akan digunakan sebagai background. Perhatikan ukuran gambar tersebut, pastikan ukuran lebar gambar sama dengan ukuran stage yaitu 800 pixel.



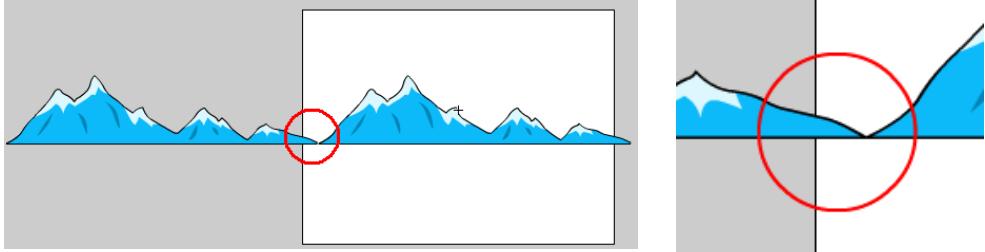
gambar background

3. Seleksi gambar dan convert menjadi **movieclip** dengan nama “**background**”.
4. Double klik movieclip **background** untuk mengeditnya.
5. Pada mode edit symbol, copy gambar dan paste-kan di sebelah kanan gambar sehingga menjadi tampak seperti pada gambar berikut:



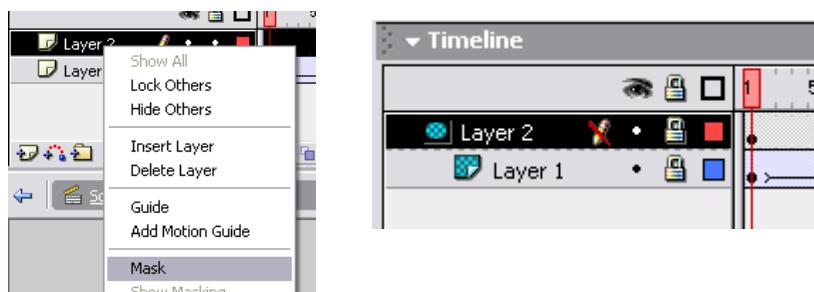
duplicasi gambar dan posisinya terhadap stage

6. Seleksi seluruh gambar kemudian convert kembali menjadi symbol movieclip dengan nama “**gunung**”.
7. Klik frame 40 masukan **keyframe**. Geser movieclip gunung ke kiri, sehingga tampak seperti pada gambar berikut:



posisi gambar pada frame 40

8. Klik kanan frame 20 kemudian pilih **create motion tween** untuk membuat animasi motion.
9. Keluar dari mode edit symbol dan jalankan movie. Kita akan mendapatkan background yang bergerak secara terus menerus tanpa terputus. Agar terlihat rapi bisa ditambahkan **masking**. Tutup movie
10. Untuk menambahkan masking, edit kembali movieclip “**background**”.
11. Tambahkan sebuah layer diatas layer 1, kemudian buatlah gambar kotak memenuhi stage.
12. Klik kanan **layer 2** dan pilih mask, maka gambar yang tampak akan terlihat rapi (semua berada di dalam stage).



menambahkan masking

13. Jalankan movie, maka tampilan background menjadi lebih baik dari sebelumnya.

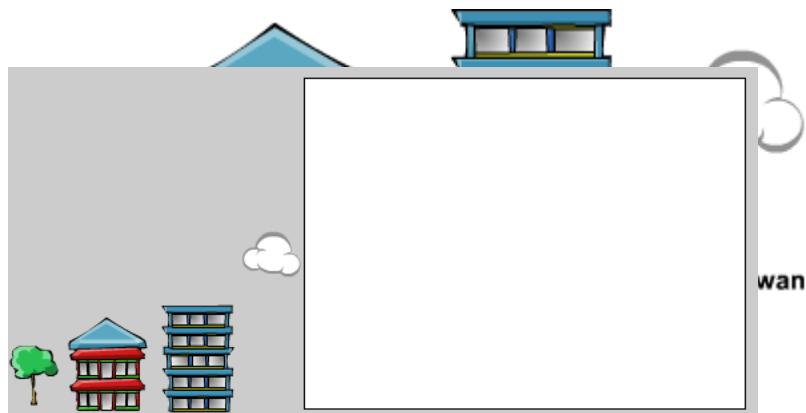
### **Scrolling Background dengan Action Sederhana**

Kelemahan background dengan teknik motion tween adalah background yang monoton, yang akan mempercepat kebosanan pemain. Untuk menghindari hal tersebut, kita dapat menggunakan action sederhana pada beberapa obyek yang akan

dijadikan background. Perhatikan contoh menggerakkan background dengan action berikut:

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
2. Buatlah 4 buah obyek berupa pohon, 2 gedung dan awan, kemudian convert masing-masing menjadi movieclip dengan nama “**pohon**”, “**gedung1**”, “**gedung2**” dan “**awan**”.

movieclip yang dipakai untuk background



3. Letakkan masing-masing movieclip di sebelah kanan luar stage.
4. Klik movieclip “pohon”, buka panel action dan ketikan action berikut :

```
onClipEvent (load) {  
    xawal = -100;  
    kecepatan = 20;  
    keluar = 0;  
    _visible = 0;  
}  
  
onClipEvent (enterFrame) {  
    // menampilkan background secara acak  
    if (random(30) == 5 and keluar == 0) {  
        keluar = 1;  
        _visible = 1;  
    }  
    // menggerakkan background  
    if (keluar == 1) {  
        _x += kecepatan;  
    }  
}
```

```

if (_x>800) {
    _x = xawal;
    keluar = 0;
    _visible = 0;
}
}

```

5. Copy action tersebut, kemudian paste-kan pada movieclip yang lain (pada movieclip “gedung1”, “gedung2”, dan “awan”).
6. Jalankan movie, maka akan kita dapat background scrolling secara acak.
7. Ubah bilangan random dan tambahkan lebih banyak obyek untuk memperbanyak variasi background.

### **Scrolling Background dengan Sistem Tiling**

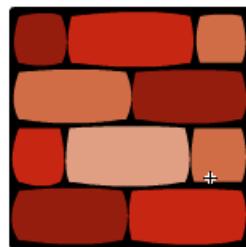
Pada beberapa game background dibuat dengan variasi yang lebih menarik dan keluar secara teratur, seperti pada game Super Mario, Sonic, dan game sejenis lainnya. Dengan membuat background yang selalu tetap dan teratur, sebuah game dapat dihafalkan dengan cepat oleh pemain, dan itulah nilai tambah sebuah game dari backgroundnya.

Untuk membuat background yang teratur dan keluar dengan urutan tertentu, kita dapat memanfaatkan sistem pengubinan (**tiling**). Akan tetapi ada beberapa hal yang harus kita perhatikan dalam membuat scrolling background dengan sistem tiling, yaitu :

1. Movie flash melambat jika kita menggerakkan banyak obyek dalam waktu yang sama.
2. Perhatikan ukuran movie flash. Jangan membuat movie dengan ukuran yang besar, karena dipastikan dapat memperlambat movie.
3. Untuk mengetahui ukuran movie yang optimal, setelah movie selesai dibuat, ujilah pada komputer yang lambat (komputer dengan spesifikasi yang buruk). Jika movie berjalan lambat, perkecilah ukuran movie. Namun jika movie berjalan normal, dipastikan movie yang anda buat bekerja baik pada komputer dengan spesifikasi yang lebih baik.

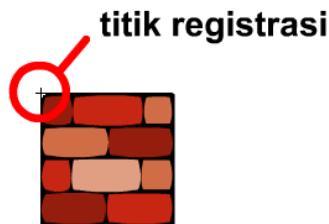
Perhatikan proses pembuatan background dengan sistem tiling berikut :

1. Buatlah movie dengan ukuran 200 x 150 pixel dan 25 fps.
2. Buatlah layer baru dan ubah nama masing-masing layer menjadi **layer background** dan **layer action**.
3. Buatlah sebuah symbol baru dengan memilih menu **insert>new symbol**. Ketikan “**ubin**” pada nama dan pilih **movieclip** pada behaviour.
4. Pada mode edit movieclip **ubin**, klik **frame 2** dan masukan **keyframe**.
5. Buatlah sebuah gambar balok (batu bata) berukuran 16 x 16 pixel seperti pada gambar berikut:



gambar batu bata pada frame 2 movieclip ubin

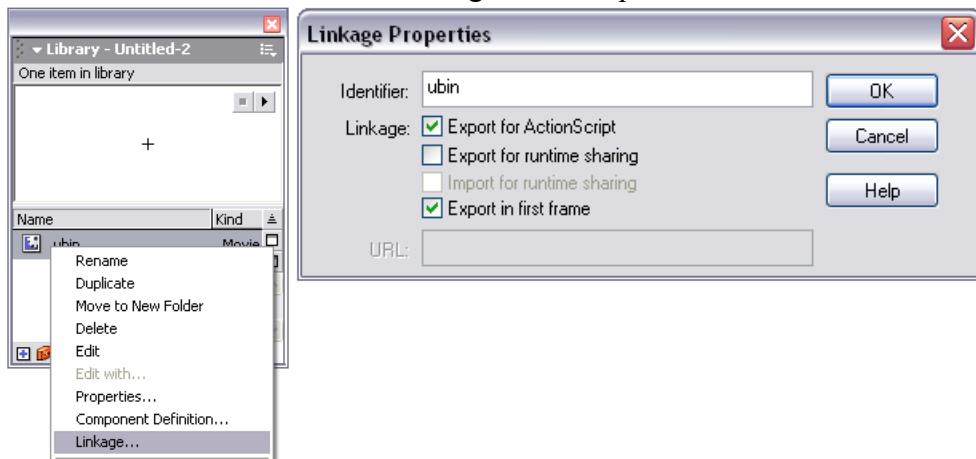
6. Seleksi gambar pada frame 2, kemudian buka panel **align** dengan memilih menu **window>align**. Atifkan tombol **Align/Distribute to Stage**, kemudian klik tombol **align left edge** dan **align top edge** untuk memindah posisi gambar tepat pada pojok kiri atas titik registrasi.



posisi titik registrasi movieclip ubin

7. Keluar dari mode edit symbol dengan menekan tombol Ctrl+E.
8. Klik **frame 1 layer background**, kemudian buatlah gambar kotak dengan warna biru (#00CCFF) memenuhi stage.
9. Buka panel library. Klik kanan movieclip ubin dan tambahkan **linkage**. Pilih **export for actionscript** dan ketikan **ubin** pada identifier.

menambahkan linkage movieclip ubin



10. Buatlah symbol baru dengan memilih menu **insert>new symbol**. Ketikan “**penampung**” pada nama dan pilih **movieclip** pada behaviour.
11. Klik kanan movieclip **penampung** pada library dan tambahkan **linkage**. Pilih **export for actionsript** dan ketikan **penampung** pada identifier
12. Klik **frame 1 layer action**, kemudian buka panel action dan ketikkan script berikut:

```
function gambarpola() {  
    for (i=0; i<tinggiTampak; i++) {  
        for (j=0; j<lebarTampak+1; j++) {  
            this.penampung.attachMovie("tile", "t_"+i+"_"+j, ++d);  
            this.penampung["t_"+i+"_"+j]._x = j*lebarUbin;  
            this.penampung["t_"+i+"_"+j]._y = i*tinggiUbin;  
  
            this.penampung["t_"+i+"_"+j].gotoAndStop(map1[i][j]+1);  
        }  
    }  
}  
  
function geser(arah) {  
    if (arah == "kanan" && lebarPeta>=j) {  
        this.penampung._x -= kecepatan;  
        xutama += kecepatan;  
        if (xutama>=lebarUbin) {  
            var iAkhir = i-tinggiTampak;  
        }  
    }  
}
```

```

        while (iAkhir<i) {
            var jAkhir = j;
            while (jAkhir<j+1) {
                this.penampung.attachMovie("tile","t_"+iAkhir+"_"+jAkhir, ++d);
                this.penampung["t_"+iAkhir+"_"+jAkhir]._x = jAkhir*lebarUbin;
                this.penampung["t_"+iAkhir+"_"+jAkhir]._y = iAkhir*tinggiUbin;
                this.penampung["t_"+iAkhir+"_"+jAkhir].gotoAndStop(map1[iAkhir]
[jAkhir]+1);
                removeMovieClip(this.penampung["t_"+iAkhir+"_"+(jAkhir-
lebarTampak-1)]);
                jAkhir++;
            }
            iAkhir++;
        }
        j++;
        xutama -= lebarUbin;
    }
}

if (arah == "kiri" && j-lebarTampak>0) {
    this.penampung._x += kecepatan;
    xutama -= kecepatan;
    if (xutama<=0) {
        j--;
        var iAkhir = i-tinggiTampak;
        while (iAkhir<i) {
            var jAkhir = j-lebarTampak-1;
            while (jAkhir<j-lebarTampak) {
                this.penampung.attachMovie("tile", "t_"+iAkhir+"_"+jAkhir, ++d);
                this.penampung["t_"+iAkhir+"_"+jAkhir]._x = jAkhir*lebarUbin;
                this.penampung["t_"+iAkhir+"_"+jAkhir]._y = iAkhir*tinggiUbin;
                this.penampung["t_"+iAkhir+"_"+jAkhir].gotoAndStop(map1[iAkhir]
[jAkhir]+1);
                removeMovieClip(this.penampung["t_"+iAkhir+"_"+(jAkhir+

```

```

lebarTampak+1)]);
                jAkhir++;
            }
            iAkhir++;
        }
        xutama +=lebarUbin;
    }
}

map1 = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0],  

[0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],  

[0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],  

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],  

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1]]  

lebarPeta = map1[0].length;  

tinggiPeta = map1.length;  

lebarTampak = 10;  

tinggiTampak = 8;  

lebarUbin = 16;  

tinggiUbin = 16;  

xawal = 16;  

yawal = 16;  

xutama = 0;  

yutama = 0;  

kecepatan = 3;  

this.attachmovie ("penampung", "penampung",1);  

this.penampung._x = xawal;  

this.penampung._y = yawal;  

gambarpola();

```

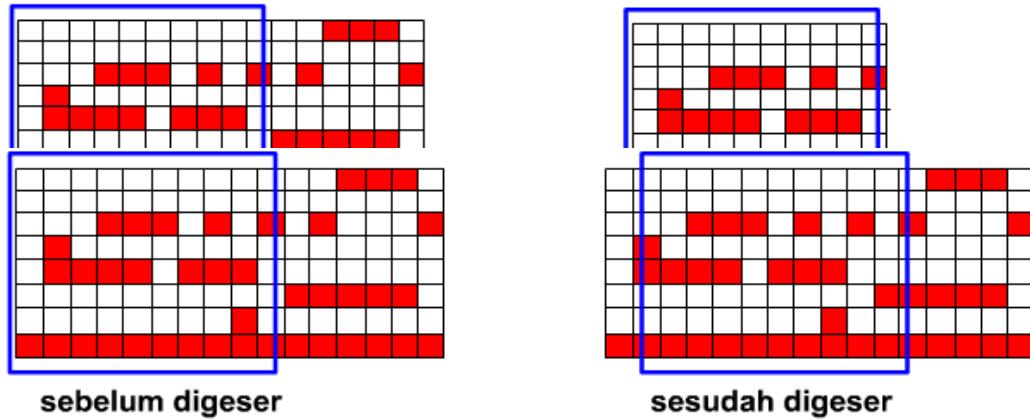
```
_root.onEnterFrame = function(){
    if (Key.isDown(Key.LEFT)){
        geser("kanan")
    }
    if (Key.isDown(Key.RIGHT)){
        geser("kiri")
    }
}
```

### 13. Jalankan movie.

Penjelasan program:

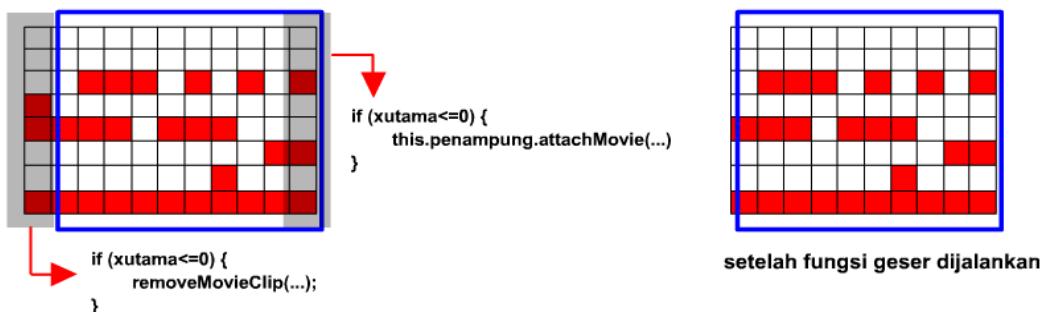
1. Pada awalnya kita tentukan dulu background yang akan dipakai dalam bentuk array 2 dimensi pada variabel map1. (**map1** = [.....];).
2. Selanjutnya variabel yang dipakai seperti (**tinggi** dan **lebar** peta, **tinggi** dan **lebar** daerah yang **tampak**, dan variabel yang lain) diset nilaiannya.
3. Untuk menampilkan background yang dapat digerakkan, dibutuhkan sebuah movieclip kosong untuk menampungnya (**this.attachMovie ("penampung", "penampung",1);**). Perhatikan bahwa movieclip **penampung** dibutuhkan ketika kita menggerakkan background dengan sistem tiling, sedangkan pada static background dengan sistem tiling kita tidak membutuhkan movieclip penampung.
4. Pada action tersebut terdapat 2 fungsi yaitu fungsi **gambarpola** dan fungsi **geser**. Fungsi **gambarpola** digunakan saat pertama kali movie dijalankan, untuk menampilkan gambar pada movie. Selanjutnya ketika terdapat tombol panah kiri atau panah kanan ditekan (**if (Key.isDown(Key.LEFT)){}**), maka fungsi **geser(arah)** dijalankan.
5. Perhatikan proses pada kedua fungsi tersebut.  
Pada fungsi **gambarpola** data bertipe array pada variabel **map1** divisualisasikan pada stage dengan tinggi dan lebar sesuai dengan variabel **lebarTampak** dan **tinggiTampak** dan diletakkan pada movieclip **penampung**.

tampilan stage akibat variabel lebarTampak dan tinggiTampak



Pada fungsi geser(arah), ketika variabel arah bernilai “**kiri**”, maka kordinat x dari movieclip **penampung** digeser ke kiri.

Bagian yang belum ditampilkan ditambahkan dengan action attachMovie  
**(this.penampung.attachMovie("tile", "t\_" +iAkhir+ "\_" +jAkhir, ++d);)** dan  
bagian yang tidak tampak akibat bergeser ke kiri di hilangkan dengan action  
removeMovieclip  
**(removeMovieClip(this.penampung["t\_" +iAkhir+ "\_" +(jAkhir+  
lebarTampak+1)])).**



bagian yang dihilangkan dan ditambah oleh fungsi geser

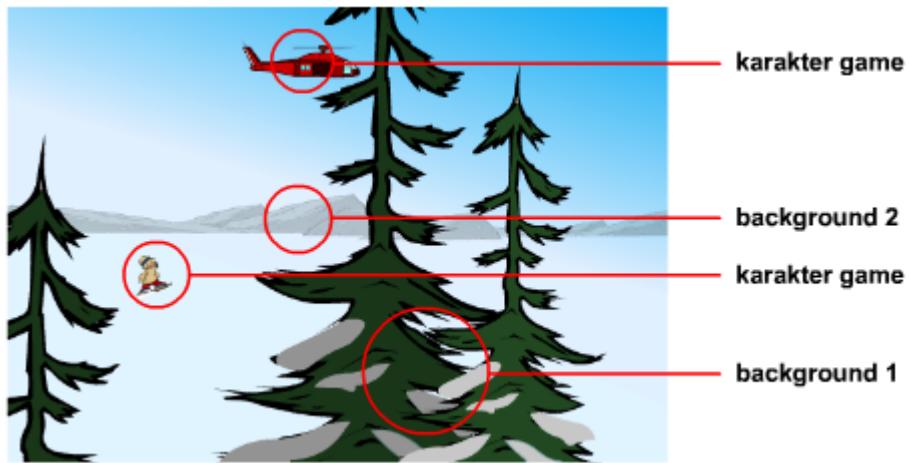
Catatan :

Untuk memahami lebih lanjut mengenai pembuatan scrolling background dengan sistem tiling pada flash game, bacalah buku “Seri Pemrograman Game Flash : Tiling Game”.

### Parallax Background

Yang dimaksud dengan parallax background adalah background berlapis dimana background yang lebih dekat dengan mata pemain bergerak lebih cepat daripada background yang lebih jauh dari mata pemain. Selain menambah kesan kedalaman, penambahan parallax background dapat menambah kesan realis suatu game.

Untuk membuat parallax background pada game, kita harus membuat minimal 2 lapis background dengan kecepatan yang berbeda. Perhatikan gambar berikut:



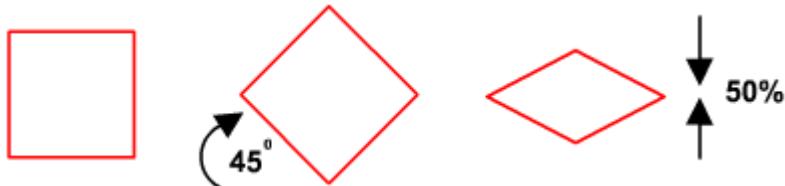
penggunaan parallax background pada game “snowboard” academy

Teknik pembuatan parallax background dapat dengan 3 cara yaitu dengan **motion**, dengan **action** sederhana dan dengan **sistem tiling**. Ketiga cara tersebut sama dengan tiga cara yang sudah dibahas sebelumnya, hanya saja proses diulang sebanyak 2 kali dengan variabel **kecepatan** yang berbeda-beda.

### Background dengan Pandangan Isometri

Background dengan pandangan isometri sering dipergunakan pada game-game PC maupun gam flash yang bertipe simulasi, RPG atau RTS. Perhatikan proses pembentukan background isometri berikut :

proses pembuatan background dengan pandangan isometri



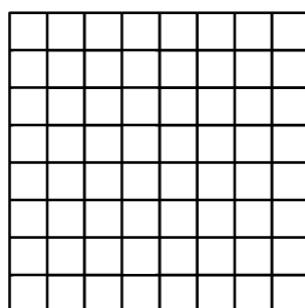
Pada dasarnya pandangan isometri diawali dengan bentuk kotak dengan pandangan datar. Kotak tersebut kemudian dirotasi 45 derajat dan dilakukan pen-skalaan sebanyak 50% secara vertikal.

Proses pembuatan background dengan pandangan isometri dapat dilakukan dengan 2 cara yaitu secara manual (menggambar pada stage) dan dengan menggunakan action

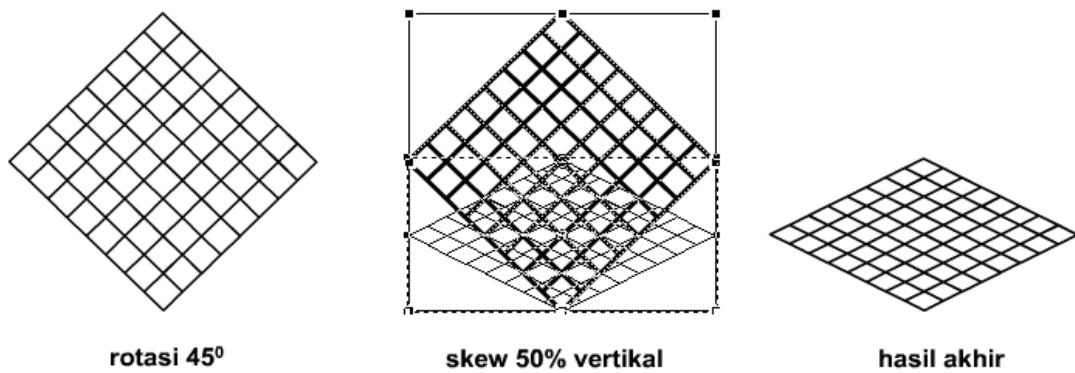
(menggunakan sitim tiling). Perhatikan proses pembuatan background isometri dengan cara manual (menggambar pada stage) berikut:

1. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel, dan 12 fps.
2. Buatlah sebuah layer baru dan ubah masing masing nama layer menjadi **layer pola** dan **layer background**.
3. Klik **frame1 layer pola**, kemudian buatlah sebuah gambar jaring-jaring persegi, seperti yang tampak pada gambar :

gambar jaring-jaring persegi

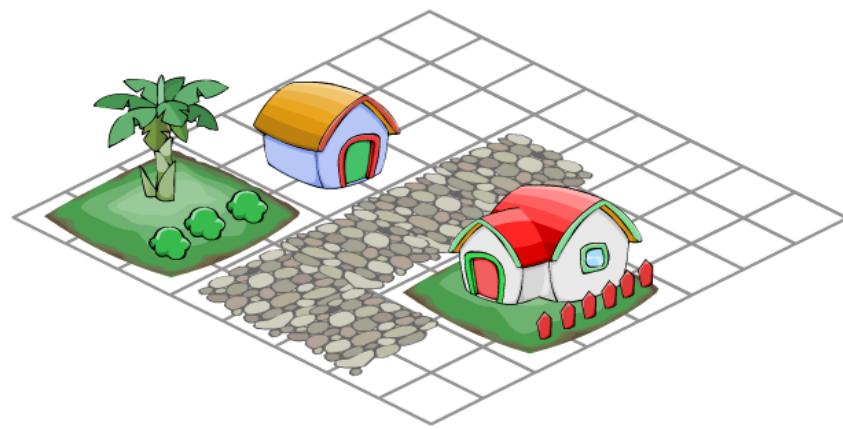


4. Seleksi gambar tersebut, kemudian dengan menggunakan transform tool, rotasikan sebesar 45 derajat
5. Kemudian lepaskan seleksi, seleksi kembali dan dengan menggunakan transform tool, skew (perkecil ukuran vertikalnya) menjadi setengah dari ukuran sebelumnya.



proses pembuatan background isometri secara manual

6. **Lock layer pola**, kemudian klik **frame 1 layer background** dan gambarlah background secara manual atau mengimpor gambar jadi. Perhatikan bahwa



peletakan background harus sesuai dengan pola yang telah dibuat. Perhatikan contoh berikut:

contoh pembuatan background isometri

7. Selesaikan seluruh background dan simpan file.

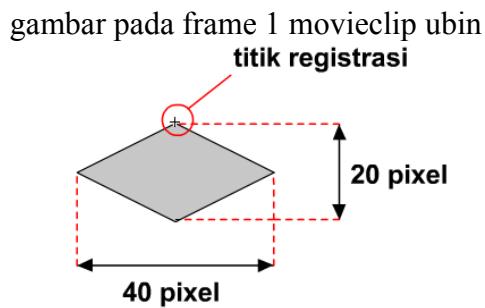
Catatan :

Membuat background isometri secara manual lebih mudah dibandingkan dengan membuat background isometri dengan teknik tiling, akan tetapi proses perhitungan pergerakan obyek nantinya akan jauh lebih sulit.

### Background Isometri dengan Teknik Tiling

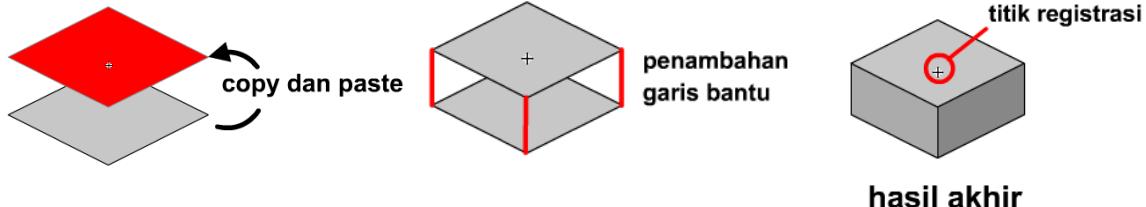
Menggunakan teknik tiling dengan pandangan isometri pada dasarnya sama dengan teknik tiling pada static background dengan pandangan datar, hanya saja perhitungan posisinya saja yang sedikit berbeda. Perhatikan proses berikut :

1. Buatlah sebuah file baru dengan ukuran 500 x 300 pixel, 25 fps.
2. Buatlah sebuah **movieclip** baru dengan memilih menu **insert>newsymbol**, ketikan **ubin** pada nama dan klik **OK**.
3. Pada mode edit symbol movieclip **ubin**, pada **frame 1** buatlah gambar kotak isometri seperti yang tampak pada gambar. Perhatikan penempatan titik registrasinya.



4. Klik **frame 2**, kemudian masukan **keyframe**. Tambahkan garis bantu untuk membentuk sebuah balok seperti tampak pada gambar :

proses menggambar obyek di frame 2



5. Keluar dari mode edit symbol.
6. Tambahkan **linkage** pada movieclip "**ubin**" dan ketikan "**tile**" pada identifier.

7. Buatlah sebuah **movieclip** baru dengan memilih menu **insert>new symbol** dan ketikan nama “**penampung**”.
8. Dalam mode edit movieclip “**penampung**”, jangan tambahkan apapun dan keluarlah dari mode edit symbl dengan menekan tombol Ctrl+E.
9. Tambahkan **linkage** pada movieclip “**penampung**” dan ketikan “**penampung**” pada identifier.
10. Klik **frame 1 layer 1**, kemudian buka panel action dan ketikan script berikut :

```

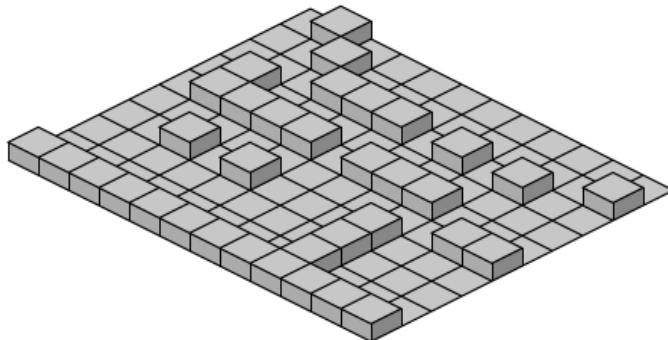
function gambarPola()
{
    lebar = map1[0].length;
    tinggi = map1.length;
    for (i = 0; i < tinggi; i++)
    {
        for (j = 0; j < lebar; j++)
        {
            this.penampung.attachMovie("tile", "t_" + i + "_" + j, ++kedalaman);
            this.penampung["t_" + i + "_" + j]._x = lebarUbin / 2 * (j - i);
            this.penampung["t_" + i + "_" + j]._y = tinggiUbin / 2 * (j + i);
            this.penampung["t_" + i + "_" + j].gotoAndStop(map1[i][j]+1);
        }
    }
}

map1 = [[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
         [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1 ],
         [0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0 ],
         [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
         [0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0 ],
         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 ],
         [0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0 ],
         [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
         [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 ],
         [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ]]
lebarUbin = 40;

```

```
tinggiUbin = 20;  
xawal = 230;  
yawal = 30;  
this.attachMovie("penampung", "penampung", 0);  
this.penampung._x = xawal;  
this.penampung._y = yawal;  
gambarPola();
```

11. Jalankan movie dan anda akan mendapati tampilan seperti pada gambar berikut:



contoh background isometri

Penjelasan program:

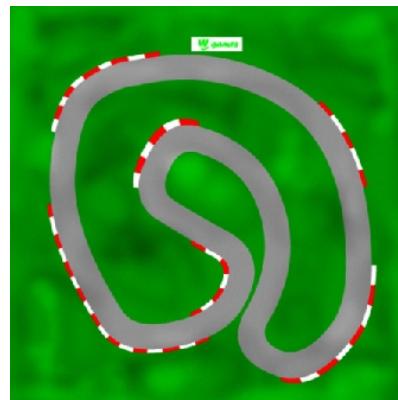
Pada dasarnya membuat background dengan pandangan isometri dengan sistem tiling sama dengan membuat background dengan pandangan datar, hanya saja pada peletakan posisi ubin dilakukan sedikit perubahan. Pada kordinat x posisi isometri menggunakan rumus **lebarUbin / 2 \* (baris - kolom)** sedangkan pada kordinat y digunakan rumus **tinggiUbin / 2 \* (baris + kolom)**.

### Background dengan Efek 3 Dimensi

Yang harus kita pahami sebelum membuat background dengan efek 3 Dimensi adalah bahwa flash adalah program yang berbasis 2 dimensi yaitu hanya memiliki kordinat x dan kordinat y. Sedangkan hal yang berbau 3 dimensi selalu memiliki sumbu yang ke 3 yaitu kordinat z. Hal kedua yang harus dipahami bahwa membuat background dengan efek 3 dimensi bukan berarti mengimport obyek 3 D dari luar Flash (seperti mengimport movie hasil dari program Swift 3D, 3D flash dan sebagainya) Tetapi membuat background dengan efek 3 dimensi adalah menciptakan kondisi seolah-olah terdapat sebuah sumbu z pada flash.

Untuk membuat efek 3 dimensi diperlukan sebuah bitmap sebagai pola background. Sebagai contoh jika kita akan membuat sebuah game bertipe racing, maka kita membutuhkan bitmap pola lintasan (sirkuit). Perhatikan contoh berikut:

1. Buatlah sebuah gambar bitmap pola sirkuit dengan ukuran 1000 x 1000 pixel menggunakan Adobe photoshop atau program sejenis lainnya seperti pada gambar: (atau gunakan file “**sircuit.jpg**” pada CD tutorial)



gambar **sircuit.jpg** yang akan dipakai sebagai background

2. Buka program Flash, buatlah file baru dengan ukuran 550 x 400 pixel dan 20 fps.
3. Import gambar yang sudah anda buat sebelumnya, dengan memilih menu **file>import**.
4. Seleksi gambar tersebut, kemudian convert menjadi movieclip dengan nama “**gerak**”.
5. Klik movieclip “**gerak**”, buka panel instance dan ketikan “**gerak**” pada instance name.
6. Klik movieclip “**gerak**”, kemudian convert kembali menjadi movieclip dengan nama “**gambar**”.
7. Klik movieclip “**gambar**”, buka panel instance dan ketikan “**gambar**” pada instance name.
8. Klik movieclip “**gambar**”, kemudian convert kembali menjadi movieclip dengan nama “**peta**”. Sehingga hierarki instance name dari movieclip “**peta**” adalah :

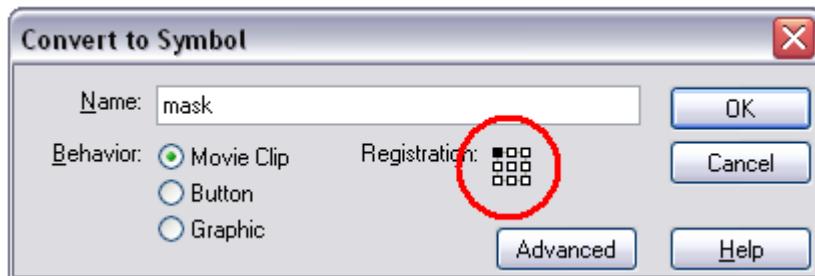


hirarki movieclip peta

9. Hapus movieclip “**peta**” dari stage, kemudian buka panel library.

10. Klik kanan movieclip “**peta**” pada library, dan tambahkan **linkage**. Pilih **export for action script** dan ketikan “**peta**” pada **identifier**.

11. Buatlah sebuah kotak dengan ukuran 550 x 3 pixel. Kemudian convert menjadi movieclip **mask**, pilih pojok kiri atas pada registration point.



titik registration movieclip “mask”

12. Klik kanan movieclip “**mask**” pada library, dan tambahkan **linkage**. Pilih **export for action script** dan ketikan “**mask**” pada **identifier**.

13. Klik **frame 1 layer 1**, kemudian buka panel action dan ketikan script berikut:

```
_quality = "LOW";
ams = 1;
xms = 3;
ms = 1;
skala = 10;
sudutKemiringan = 190;
for (i=1; i<=sudutKemiringan; i += ms) {
    total++;
    ams += xms/30;
    ms = Math.floor(ams);
    peta = attachMovie("peta", "peta"+total, i*3+1);
    peta.gambar._xscale = 100;
    peta.gambar._yscale = 100;
    peta._x = 275;
    peta._y = 400;
    peta._xscale = i*skala;
    peta._yscale = i*skala;
    peta.onEnterFrame = function() {
        this.gambar._rotation = _root.rotasi;
```

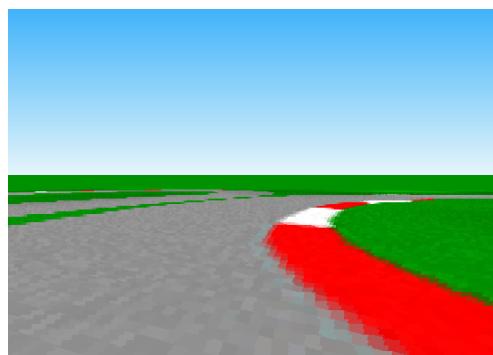
```

        this.gambar.gerak._y += _root.nilaiCos;
        this.gambar.gerak._x += _root.nilaiSin;
    };
    tutup = attachMovie("mask", "mask"+i, i*3+2);
    tutup._height = ms;
    tutup._y = 200+i;
    peta.setMask(tutup);
}

_root.onEnterFrame = function() {
    kecMax = 20;
    if (Key.isDown(Key.RIGHT)) {
        _root.rotasi -= 5;
    } else if (Key.isDown(Key.LEFT)) {
        _root.rotasi += 5;
    } else if (Key.isDown(Key.UP)) {
        _root.kec = 5;
    } else if (Key.isDown(Key.DOWN)) {
        _root.kec = -5;
    } else {
        _root.kec = 0;
    }
    _root.nilaiCos = _root.kec*Math.cos(_root.rotasi*Math.PI/180)/2;
    _root.nilaiSin = _root.kec*Math.sin(_root.rotasi*Math.PI/180)/2;
};

```

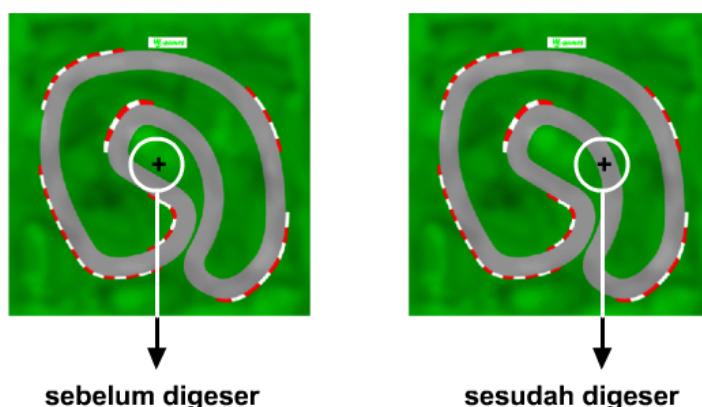
14. Jalankan movie. untuk mempercantik tampilan tambahkan sebuah layer dibawah layer 1 dan buatlah obyek berupa tanah datar dan langit.



## hasil program

Penjelasan program :

1. Pada awal script diset beberapa variabel yang akan dipakai dalam menggerakkan background. Selain itu juga terdapat properti **\_quality** yang diset “**LOW**” untuk mempercepat jalannya movie. Apabila anda ingin memperbaiki kualitas gambar, anda bisa mengeset nilai **\_quality** menjadi “**HIGH**”.
2. Proses pembuatan efek 3 dimensi pada dasarnya adalah menggambar ulang movieclip “**peta**” beberapa kali sesuai dengan variabel **sudutKemiringan** (**for (i=1; i<=sudutKemiringan; i += ms) {}**) dengan skala yang berbeda (**peta.\_xscale = i\*skala; peta.\_yscale = i\*skala;**).
3. Agar gambar terlihat rapi, maka dilakukan proses masking menggunakan action **peta.setMask(tutup)** dimana **tutup** berisi movieclip “**mask**” (**tutup = attachMovie("mask", "mask"+i, i\*3+2);**).
4. Untuk menggerakkan ke berbagai arah digunakan rumus sinus dan cosinus (**\_root.nilaiCos = \_root.kec\*Math.cos(\_root.rotasi\*Math.PI/180)/2;**) sebagai nilai perubahan kordinat x dan y movieclip “**gerak**”
5. Perubahan ke kanan dan ke kiri akibat fungsi sinus (**this.gambar.gerak.\_x += \_root.nilaiSin**), sedangkan perubahan ke depan dan ke belakang akibat fungsi cosinus (**this.gambar.gerak.\_y += \_root.nilaiCos**).
6. Jika anda menggunakan gambar “**sircuit.jpg**” pada CD tutorial, maka pada saat movie pertama kali dijalankan, anda akan berada pada daerah berwarna hijau. Jika anda ingin berada pada track (jalan), ubah posisi bitmap pada movieclip “**gerak**”. Letakkan registration poin pada jalan seperti pada gambar berikut:



perbaikan posisi registration poin pada movieclip “**gerak**”

## **Sound**

Selain game play(sistem permainan), karakter, dan background, suara dalam game juga dapat menambah nilai suatu game. Dalam game suara dibedakan menjadi 2 yaitu suara background (Back Ground Music) dan suara efek (Sound Effect). Kedua jenis sound tersebut dapat dibuat dengan 3 cara, yaitu : dengan cara manual, dengan action script attachSound, dan dengan mengimpor Sound.

## Mengatur Sound dengan Cara Manual

Yang dimaksud dengan cara manual menurut saya adalah dengan mengimpor file sound ke dalam library, kemudian men-drag-nya langsung ke stage. Keuntungan dari cara ini adalah pada sisi kemudahannya, akan tetapi untuk mengatur ulang sound sangat sulit untuk dilakukan. Perhatikan contoh berikut :

1. Siapkan sebuah file suara atau gunakan file “**bgm1.mp3**” pada folder suara di CD tutorial.
2. Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
3. Pilih menu **file>import to library..** dan pilih file suara yang akan diimpor, kemudian tekan OK.



bentuk symbol bertipe sound pada library

4. Untuk menggunakannya drag langsung dari library ke stage. Maka tampilan pada timeline akan berubah.



tampilan sound pada sebuah frame

5. Jalankan file, maka akan terdengar sound yang diimport sebanyak satu kali.
6. Tutup movie. Klik **frame 1** dan buka panel properties. Perhatikan pada bagian sound properties. Kita dapat memilih sound yang diinginkan atau menghilangkan sound pada roll menu **Sound**. Pada roll menu **effect** anda dapat memilih beberapa efek standard seperti fade in, fade out, pan left dan sebagainya. Atau anda dapat membuat efek sendiri dengan menekan tombol edit. Pada roll menu **sync** anda dapat memilih mode suara yang dijalankan :

event      berarti sound dapat dimainkan pada 1 frame saja, akan suara akan saling bertumpukan apabila frame yang aktif lebih dari 1 dan kurang

- dari panjang sound. Contoh movie memiliki 5 frame aktif sedangkan panjang lagu adalah 10 detik, maka suara akan bertumpukan.
- start berarti sound akan dimainkan sampai selesai, tanpa bertumpukan
  - stop untuk menghentikan sound
  - steam sound dimainkan sepanjang frame aktif. Untuk memainkan satu sound secara utuh, frame yang aktif harus sepanjang lamanya sound.

**Loop** menunjukkan berapa kali sebuah sound dimainkan.



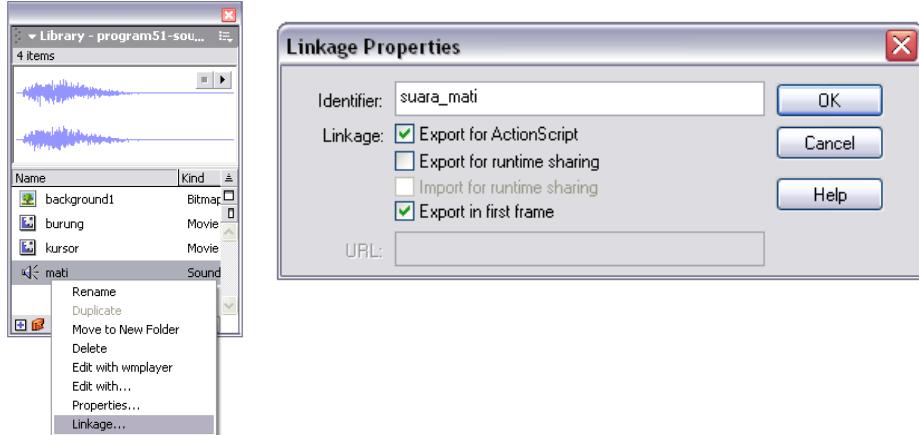
7. Klik **frame 10** kemudian masukan frame dengan menekan tombol **F5**. Ubahlah properties untuk lebih memahami arti sync dan effect.

### **Menambah Sound dengan attachSound();**

Dengan menambahkan sound secara manual terdapat kelemahan seperti sulit untuk mengatur volume, menghentikan sound, memulai sound pada waktu tertentu dan sebagainya. Untuk mengatasi masalah tersebut, flash menyediakan action **attachSound()**; untuk memanfaatkan action tersebut, kita harus menambahkan linkage pada symbol sound. Perhatikan contoh penggunaan action tersebut berikut ini:

1. Siapkan sebuah file suara atau gunakan file “**mati.mp3**” pada folder suara di CD tutorial.
2. Bukalah file latihan menggunakan static background yang telah kita buat sebelumnya, atau buka file **program42-backgroundStatic.fla** pada CD tutorial.
3. Pilih menu **file>save as**, dan ketikan nama file yang baru.
4. Pilih menu **file>import to library..** dan pilih file suara yang akan diimpor, kemudian tekan **OK**

5. Buka panel library, kemudian klik kanan pada library sound yang telah diimpor dan tambahkan **linkage**. Pilih **export for action script** dan ketikan **suara\_mati** pada **identifier**.



menambahkan linkage ada symbol sound

6. Klik movieclip “**burung**”, kemudian buka panel action dan tambahkan action-nya menjadi :

```

onClipEvent (load) {
    _y = 100+random(400);
    xawal = _x;
    kecepatan = 10;
    naik = 0;
    keluar = 0;
    _visible = 0;
    suara = new Sound();
    suara.attachSound("suara_mati");
}

onClipEvent (enterFrame) {
    //burung belum keluar
    if (keluar == 0 and random(20) == 5) {
        keluar = 1;
        _visible = 1;
    }
    //burung keluar
    if (keluar == 1) {
        _x += kecepatan;
        if (random(10) == 3) {
    
```

```

        //gerakan acak naik
        naik = 1;
    } else if (random(10) == 4) {
        //gerakan acak turun
        naik = 2;
    } else if (random(10) == 5) {
        // gerakan lurus
        naik = 0;
    }
    //menggerakkan naik.
    if (naik == 1 and _y>50) {
        _y -= 5;
    } else {
        //menggerakkan turun.
    }
    if (naik == 2 and _y<550) {
        _y += 5;
    }
    //keluar dari stage, maka kembalikan lagi ke posisi awal
    if (_x>800) {
        _x = xawal;
        _y = 100+random(400);
        keluar = 0;
        _visible = 0;
    }
}

onClipEvent (mouseDown) {
    if (hitTest(_root.kursor)) {
        _x = xawal;
        _y = 100+random(400);
        keluar = 0;
        _visible = 0;
        suara.start(0, 1);
    }
}

```

```
}
```

7. Apabila pada latihan sebelumnya anda membuat beberapa burung dengan variabel kecepatan dan nilai random yang berbeda, tambahkan juga action yang sama pada masing-masing movieclip burung tersebut.
8. Jalankan movie.

Penjelasan program:

1. Pada movie event load, dilakukan pengesetan awal sebuah obyek Sound dengan action **suara = new Sound();** dengan action tersebut berarti telah dibuat sebuah obyek baru bertipe sound.
2. Selanjutnya sound ditambahkan pada movie dengan action **suara.attachSound("suara\_mati");** "suara\_mati" merupakan linkage dari sound yang telah diimport.
3. Untuk menjalankan sound, yang dalam hal ini diletakkan pada saat burung tertembak adalah dengan menggunakan action **suara.start(0, 1);** dimana **0** merupakan detik dimulainya sound dan **1** adalah jumlah sound diainkan.

### **Mengimpor Sound dengan loadSound();**

Dalam flash,penggunaan sound yang banyak apalagi dengan durasi yang panjang dapat menyebabkan ukuran file membengkak. Kelemahan tersebut dapat kita tutupi dengan cara mengimpor sound dari luar movie. Akan tetapi untuk mengimpor sound dari luar, flash hanya menyupport file bertipe MP3. Perhatikan contoh berikut :

1. Siapkan sebuah file suara atau gunakan file "**bgm1.mp3**" pada folder suara di CD tutorial. Copy file tersebut ke folder dimana anda bekerja.
2. Lanjutkan file latihan attachSound atau buka file **program51-sound\_attachSound.fla** pada CD tutorial.
3. Klik movieclip "**burung**", buka panel action dan tambahkan pada blok movie event load menjadi :

```
onClipEvent (load) {  
    _y = 100+random(400);  
    xawal = _x;  
    kecepatan = 10;
```

```

naik = 0;
keluar = 0;
_visible = 0;
suara = new Sound();
suara.attachSound("suara_mati");
bgm = new Sound();
bgm.loadSound("bgm1.mp3", 0);
bgm.start(0.15, 10);
}

```

4. Simpan file dan jalankan movie.

Penjelasan program:

1. Untuk membuat suara background, kita harus membuat sebuah obyek baru bertipe sound dengan menggunakan action **bgm = new Sound();**
2. Untuk mengimpor sound, digunakan action (**bgm.loadSound(nama file yang akan diload, 0)**). nilai **0** berarti sound tidak streaming. (streaming berarti sound akan langsung dimulai dimainkan meskipun belum ter-load semuanya).
3. Untuk memainkan sound yang diimport, digunakan action **bgm.start(0.15, 10);** nilai **0.15** dipakai agar looping suara bgm tidak terputus, apabila dimasukkan nilai 0 akan terdengar suara yang terputus sesaathal tersebut dikarenakan suara **bgm1.mp3** yang diload dimulai pada detik ke 0.15. nilai **10** berarti suara yang diload akan dimainkan sebanyak 10 kali.

Catatan:

untuk mengimpor sound dari folder yang berbeda, tuliskan dengan jelas nama folder.

contoh : **bgm.loadSound("suara/bgm1.mp3", 0);**

untuk mengatur volume digunakan action **suara.setVolume(0-100);**

untuk megeset speaker yang aktif digunakan action **suara.setPan(-100 – 100);**

dimana –100 berarti speaker kiri yang aktif 0 berarti kedua speaker aktif dan 100

berarti speaker kanan yang aktif.

Untuk menghentikan sebuah sound dapat digunakan action **suara.stop();**

untuk menghentikan semua suara dapat digunakan action **stopAllSounds();**

## **Membuat Game Arcade**

Pada bab ini kita akan memcoba membuat sebuah game sederhana bertipe arcade. Apabila anda memahami proses pembuatan game ini, maka dapat dipastikan anda akan dapat membuat game yang lebih kompleks.

### **Story Board Game**

Judul game	Alien Attack
Jenis game	Arcade
Sistem kendali	Keyboard: arah panah untuk menggerakkan karakter dan

	spasi untuk menembak
Karakter utama	Pesawat jet
Musuh	Pesawat alien
Background Setting	Angkasa luar dengan teknik scrolling sederhana
Sistem permainan	Pemain harus menembak seluruh musuh yang ada. Pada waktu-waktu tertentu, pemain dapat memperoleh bonus berupa armor bonus yang akan mengaktifkan perisai pesawat, gun bonus yang akan menambah kemampuan senjata pesawat dan speed bonus yang akan mempercepat gerakan pesawat. Permainan berakhir jika pesawat meledak dan life point habis.

### **Memvisualisasikan Storyboard**

Setelah storyboard tersusun, tahapan berikutnya adalah memvisualisasikannya boleh secara manual drawing pada kertas atau secara digital sebagai beikut :



visualisasi dari story board

### Membuat Karakter dalam Game

Dari visualisasi tersebut, kita membutuhkan beberapa gambar karakter yaitu pesawat pemain, pesawat musuh, animasi ledakan, beberapa bonus, senjata dan background. Seluruh obyek tersebut dapat kita buat menggunakan drawing tool, atau mengimpor gambar siap pakai seperti pada penjelasan sebelumnya.

### Membuat Movieclip Ledakan

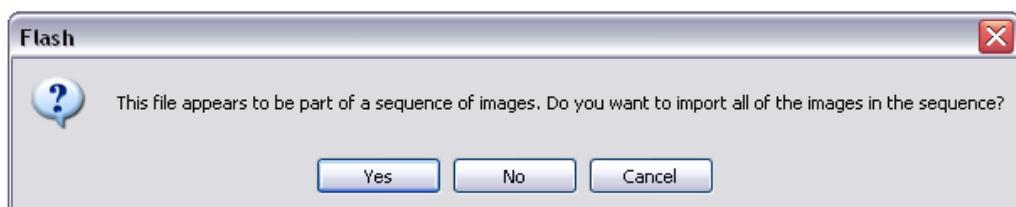
Ledakan hampir selalu dibutuhkan dalam game-game seperti yang kita buat sekarang. Pada dasarnya membuat efek ledakan yang paling efektif adalah dengan menggunakan teknik animasi frame by frame. Perhatikan proses pembuatan game “Alien Attack” berikut, yang dimulai dari pembuatan movieclip ledakan:

1. Buatlah sebuah file baru dengan ukuran 400 x 600 pixel dan 12 fps.
2. Buatlah 7 buah layer baru, dan ubah nama masing-masing layer menjadi **layer bg3, bg2, bg1, karakter, senjata, bonus, properti** dan **layer action**.



susunan layer pada game “alien attack”

3. Klik menu **insert>new symbol**. Ketikan “**ledakan**” pada nama dan pilih **movieclip** pada behaviour.
4. Pada mode edit symbol movieclip ledakan, klik **frame 1 layer 1**. Kemudian pilih menu **file>import**. Pilihlah file “**gambar\ledakan\image1.png**” pada CD tutorial.
5. Ketika muncul **sequence image dialog** (akibat dari mengimpor file dengan nama yang berurutan), pilih “**yes**” maka akan terbentuk animasi ledakan secara frame by frame.



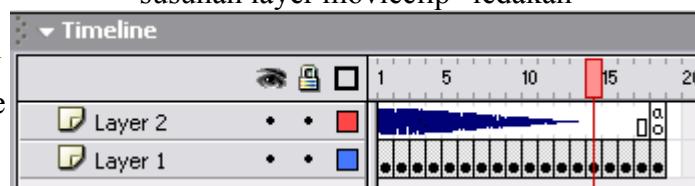
import sequence of images dialog

6. Perhatikan bahwa titik registrasi tidak terletak tepat di tengah ledakan. Anda dapat merubahnya tepat di tengah dengan align panel satu frame demi satu frame.
  7. Tambahkan sebuah layer baru diatas **layer 1**. Kemudian import file “**suara/ledakan.wav**” untuk menghasilkan sebuah efek suara ledakan.
  8. Klik **frame 18 layer 2**, kemudian masukan keyframe dengan menekan tombol F6.
- Buka panel action, dan ketikan action

```
stop();
```

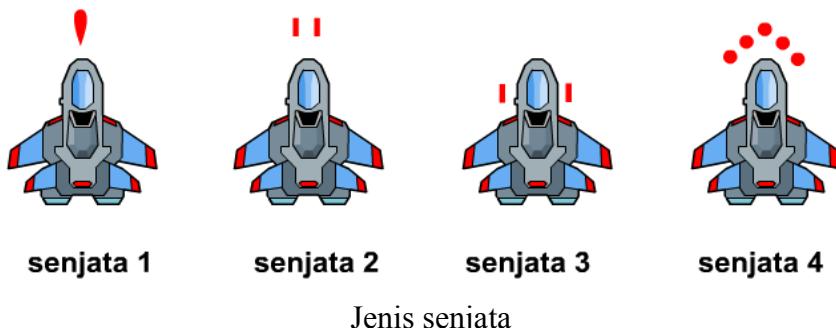
susunan layer movieclip “ledakan”

9. Keluar dari
10. Simpan file



## Membuat Movieclip Senjata Pemain

Dalam storyboard dijelaskan bahwa pesawat pemain memiliki senjata yang dapat meningkat kemampuannya ketika mendapatkan sebuah bonus. Hal tersebut berarti kita harus membuat beberapa jenis senjata dengan kemampuan yang berbeda. Sebagai contoh dalam game ini terdapat 4 tipe senjata, perhatikan gambar berikut :



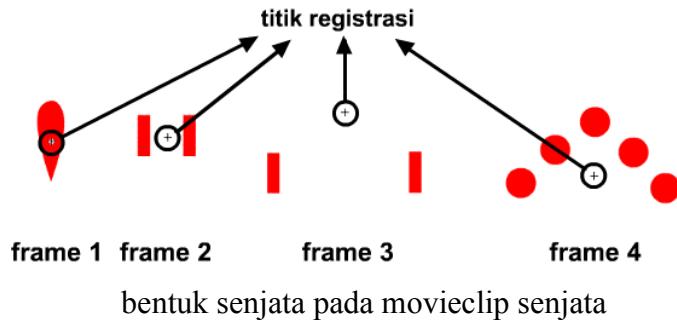
Untuk membuat empat jenis senjata tersebut, perhatikan langkah-langkah berikut:

1. Lanjutkan file “**alien attack.fla**”.
2. Buatlah sebuah symbol baru dengan menekan menu **insert>new symbol**. Ketikan nama “**laser**” dan pilih **movieclip** pada behaviour.
3. Pada mode edit symbol buatlah gambar senjata jenis pertama, seperti yang tampak pada gambar berikut :



gambar pada movieclip “laser”

4. Seleksi gambar senjata tersebut, kemudian convert kembali menjadi movieclip dengan nama “**senjata**”.
5. Klik movieclip “**senjata**”, kemudian buka panel properties dan ketikan “**Jenis**” pada instance name.
6. Double klik movieclip “**senjata**” untuk mengeditnya.
7. Klik **frame 2 layer 1**, kemudian masukan blank keyframe. Buatlah gambar jenis senjata yang kedua, seperti yang tampak pada gambar.
8. Selanjutnya buatlah gambar jenis senjata yang berbeda pada **frame 3** dan **frame 4**.
- 9.



10. Keluar dari mode edit symbol , klik **frame 1 layer senjata**, drag movieclip “laser” dari library ke stage dan simpan file.

### Membuat Movieclip Perisai

Dalam storyboard dijelaskan bahwa pesawat memiliki perisai ketika mendapatkan armor bonus. Dan ketika pesawat tertembak atau bertabrakan dengan musuh, maka perisai akan melindungi pesawat pemain. Dengan imajnasi kita visualisasikan bentuk perisai tersebut seperti pada gambar berikut :

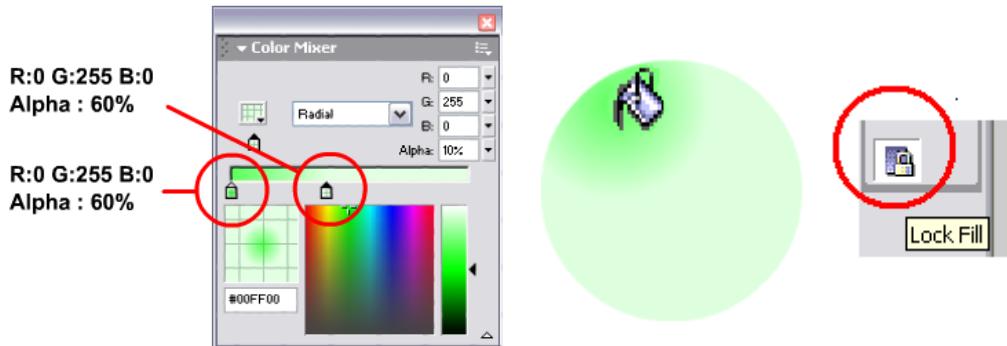


gambar perisai ketika aktif

untuk membuat perisai tersebut, perhatikan langkah-langkah di bawah ini :

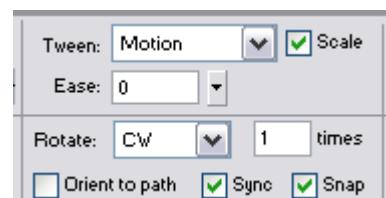
1. Lanjutkan fil Lanjutkan file “**alien attack.fla**”.
2. Buatlah sebuah symbol baru dengan menekan menu **insert>new symbol**. Ketikan nama “**perisai**” dan pilih **movieclip** pada behaviour.
3. Tambahkan 2 buah layer baru di atas layer 1 dan ubah namanya menjadi **layer suara** dan **layer action**.
4. Klik **frame 2 layer 1**, kemudian masukkan keyframe.
5. Pada **frame 2 layer 1**, buatlah sebuah lingkaran dengan menggunakan oval tool dengan ukuran sekitar 120x120 pixel. Hapuslah stoke (garis tepi)-nya.
6. Buka panel color mixer, pilih **radial** pada option fillstyle. Klik gradien pointer sebelah kiri dan ubah menjadi warna hijau (R : 0, G : 255, B : 0) dan ubah alpha value menjadi 60%. Klik gradien pointer sebelah kanan dan ubah menjadi warna hijau (R : 0, G : 255, B : 0) dan ubah alpha value menjadi 10%.Kemudian klik

tombol Paint Bucket Tool dan klik pada pojok kiri atas lingkaran. (matikan atau aktifkan lock fill jika tidak terbentuk radial fill yang diinginkan dan klik lagi paint bucket tool dan klik pada pojok kiri atas lingkaran)



membuat radial fill dalam flash

7. Seleksi lingkaran dan convert kembali menjadi movieclip dengan nama “plasma”.
8. Klik **frame 24 layer 1** kemudian masukkan keyframe.
9. Klik **frame 10**, kemudian buka panel properties. Pilih **motion** pada option tween dan pilih **CW** pada option rotate. Maka akan terbentuk animasi perisai yang berputar.



membuat animasi motion pada movieclip perisai

10. Klik **frame 2 layer suara**, kemudian impor file “suaraPerisai.wav” dari CD tutorial ke library. Kemudian drag sound tersebut ke stage.
11. Klik **frame 24 layer action**, masukkan keyframe. Kemudian buka panel action dan ketikan script:

```
gotoAndStop(1);
```

12. Keluar susunan layer movieclip perisai

	as	suara	perisai
•	•	•	
•	•	•	
•	•	•	

Membuat

Karakter utama dari game yang kita buat adalah sebuah pesawat luar angkasa yang akan meledak jika tertembak atau bertumbukan dengan musuh, atau dapat mengaktifkan perisai jika pesawat tersebut memiliki armor. Dari deskripsi tersebut

kita bisa membuat sebuah movieclip pesawat dengan langkah-langkah sebagai berikut:

1. Lanjutkan file “**alien attack.fla**”.
2. Klik **frame 1 layer karakter**. Kemudian buatlah sebuah gambar pesawat dengan menggunakan drawing tool seperti pada gambar atau sesuai dengan imajinasi anda (atau importlah sebuah file png yang telah anda buat sebelumnya).

gambar karakter pesawat



3. Seleksi gambar pesawat tersebut, kemudian convert menjadi **movieclip** dengan nama **pesawat**.
4. Double klik **pesawat** untuk mengeditnya. Pada mode edit symbol movieclip **pesawat**, tambahkan 2 layer di atas layer 1 dan ubah nama masing-masing layer menjadi **layer perisai** dan **layer action**.
5. Klik **frame 1 layer perisai**, kemudian drag movieclip “**perisai**” dari library ke stage. Letakkan tepat di tengah stage (perhatikan bahawa movieclip perisai berupa sebuah titik saja, hal ini dikarenakan pada frame 1 movieclip perisai tidak terdapat obyek apapun / blank keyframe). Tips dari saya : untuk mempermudah mengatur posisi dan ukuran perisai, klik movieclip perisai dan buka panel properties. Ubah symbol behaviour menjadi graphic, pilih **single frame** pada **option for graphic** dan ketikan **2** pada **first frame**. Atur posisi dan ukuran perisai, setelah dirasa cukup kembalikan behaviournya menjadi movieclip.



mengatur posisi dan ukuran perisai

6. Klik movieclip **perisai**, buka panel properties dan ketikan “**perisai**” pada instance name.

7. Klik **frame 2 layer 1**, kemudian masukan blank keyframe dengan menekan tombol F7.
8. Drag movieclip “**ledakan**” dari library ke stage. Atur posisi dan ukurannya terhadap pesawat.
9. Klik movieclip “**ledakan**”, buka panel properties dan ketikan “**ledakan**” pada instance name.
10. Klik **frame 1 layer action**, kemudian buka panel action dan ketikan script :

```
stop();
```

11. Keluar dari mode edit symbol, klik **pesawat** buka panel properties dan ketikan “**pesawat**” pada instance name kemudian simpan file.

### **Membuat Movieclip Senjata Pesawat Musuh**

Untuk meningkatkan kesulitan permainan, pesawat musuh juga dapat menembakkan senjata. Untuk membuatnya, perhatikan langkah berikut :

1. Klik **frame 1 layer senjata**, kemudian buatlah sebuah gambar senjata musuh menggunakan drawing tool (pada contoh ini senjata musuh menggunakan bentuk elips berwarna kuning (#FFFF00) )



gambar senjata musuh

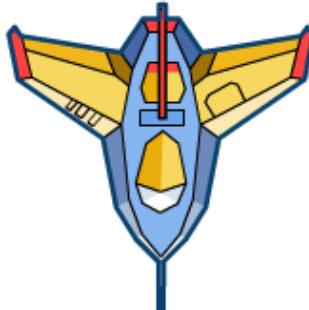
2. Seleksi gambar tersebut, kemudian convert menjadi **movieclip** dengan nama **senjataMusuh**.
3. Simpan file.

### **Membuat Movieclip Pesawat Musuh**

Sebelum membuat pesawat musuh kita tentukan terlebih dahulu jenis musuh dan kemampuannya. Dalam game ini dicontohkan terdapat 2 jenis musuh, akan tetapi anda bisa menambahkannya menjadi beberapa jenis lagi. Sedangkan kemampuan musuh adalah dapat menembak pemain pada saat-saat tertentu. Semakin tinggi level, semakin agresif musuh dalam bergerak dan menembak.

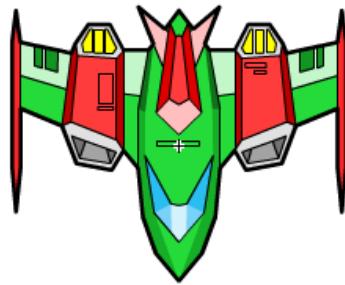
Untuk membuat movieclip pesawat musuh, perhatikan cara berikut :

1. Lanjutkan file “**alien attack.fla**”.
2. Klik **frame 1 layer karakter**. Kemudian buatlah sebuah gambar pesawat musuh dengan menggunakan drawing tool seperti pada gambar atau sesuai dengan imajinasi anda (atau importlah sebuah file png yang telah anda buat sebelumnya).



gambar musuh

3. Seleksi gambar pesawat tersebut, kemudian convert menjadi **movieclip** dengan nama **pesawatMusuh**.
4. Double klik **pesawatMusuh** untuk mengeditnya. Pada mode edit symbol movieclip **pesawatMusuh**, tambahkan layer baru di atas layer 1 dan ubah **layer action**.
5. Seleksi gambar musuh tersebut, kemudian convert kembali menjadi movieclip dengan nama “**jenis musuh**”.
6. Double klik movieclip **jenis musuh** untuk mengeditnya. Klik **frame 2**, kemudian masukan blank Keyframe.
7. Pada **frame 2** tersebut buatlah gambar musuh yang kedua seperti pada gambar atau sesuai dengan imajinasi anda atau mengimpor gambar yang sudah jadi.



musuh jenis kedua

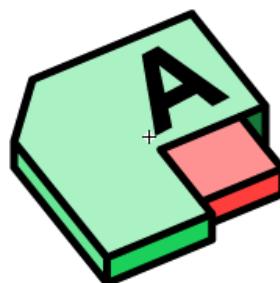
8. Jika anda ingin jenis musuh yang lain (lebih dari 2), buatlah gambar pesawat musuh selanjutnya pada frame 3 dan seterusnya.
9. Keluar dari mode edit symbol **jenis musuh** dan kembali ke mode edit symbol movieclip **pesawatMusuh**.

10. Klik movieclip **jenis musuh**, kemudian buka panel properties dan ketikan “**jenis**” pada instance name.
  11. Klik **frame 2 layer 1**, kemudian masukan blank keyframe dengan menekan tombol F7.
  12. Drag movieclip “**ledakan**” dari library ke stage. Atur posisi dan ukurannya terhadap pesawat musuh.
  13. Klik movieclip “**ledakan**”, buka panel properties dan ketikan “**ledakan**” pada instance name.
  14. Klik **frame 1 layer action**, buka panel action dan ketikan script:
- ```
stop();
```
15. Keluar dari mode edit symbol dan simpan file.

### **Membuat Movieclip Bonus**

Dalam storyboard dijelaskan bahwa terdapat bonus yang keluar secara acak. Bonus dibagi menjadi 4 yaitu bonus armor, bonus senjata, bonus kecepatan, dan bonus life point. Perhatikan proses pembuatan movieclip bonus berikut:

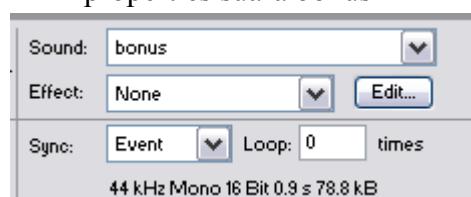
1. Klik **frame 1 layer bonus**, kemudian buatlah sebuah gambar seperti yang tampak pada gambar berikut atau sesuai dengan keinginan anda



gambar bonus armor

2. Seleksi gambar tersebut, kemudian convert menjadi movieclip dengan nama “**bonus armor**”.
3. Double klik movieclip “**bonus armor**”, kemudian pada mode edit symbol tambahkan sebuah layer diatas layer 1.
4. Klik **frame 2 layer 1**, masukan blank keyframe. Import file “**Bonus.wav**” dari CD tutorial ke library. Kemudian drag **suaraBonus** dari library ke stage.
5. Klik **frame 2 layer 1**, kemudian buka panel properties. Pilihlah **event** pada option **sync**.

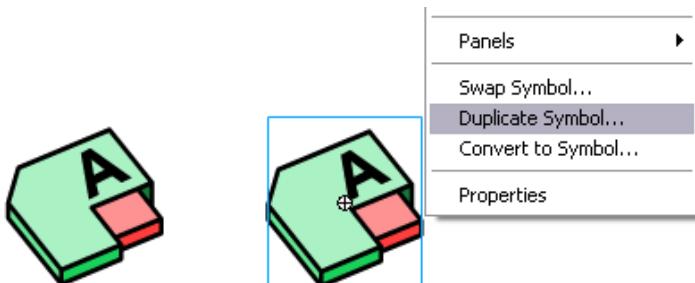
properties suara bonus



6. Klik **frame 1 layer 2**, kemudian buka panel action dan ketikan script :

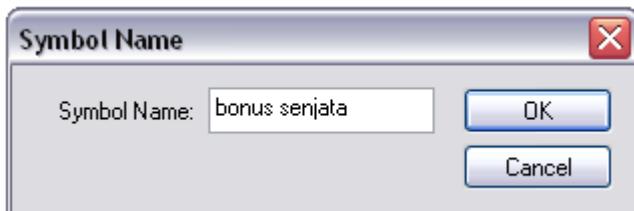
```
stop();
```

7. Copy movieclip “**bonus armor**”, kemudian paste-kan disampingnya.  
8. Klik kanan movieclip “**bonus armor**” hasil copy-paste pada langkah no 3, kemudian pilih **duplicate symbol**.



menduplikasi movieclip

9. Ketikan “**bonus senjata**” pada symbol name, maka akan terbentuk movieclip baru dengan nama “**bonus senjata**”.



dialog symbol name saat menduplikasi symbol

10. Double klik movieclip “**bonus senjata**”. Pada mode edit symbol, ubah huruf “A” pada gambar menjadi huruf “G” (untuk Gun). Kemudian keluar dari mode edit symbol.  
11. Lakukan langkah no 3 sampai no 6 untuk membuat movieclip “**bonus kecepatan**” (S), dan “**bonus lifepoint**” (L).



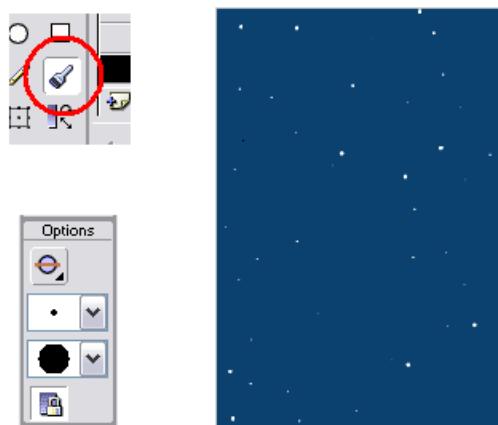
**bonus senjata    bonus speed    bonus life**  
movieclip bonus

12. Simpan file.

## Membuat BackGround

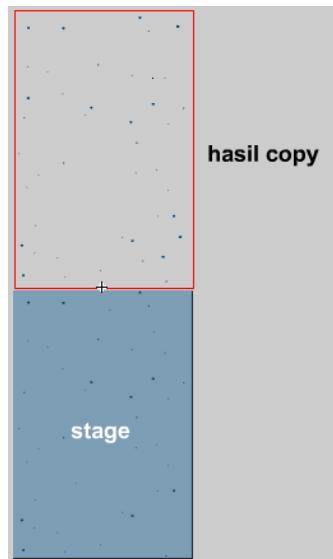
Pada susunan layer terdapat 3 buah layer untuk background yaitu layer bg3 untuk background paling bawah, layer bg2 untuk background bintang yang scrolling dengan kecepatan lambat dan layer bg1 untuk background bintang yang scrolling dengan kecepatan yang lebih cepat. Untuk membuat background perhatikan langkah berikut:

1. Klik **frame 1 layer bg3**, kemudian buatlah kotak dengan warna biru gelap (#0D4A77) dengan ukuran yang sama dengan ukuran stage, yaitu 400 x 600 pixel.
2. Klik ikon **lock layer bg3**, kemudian klik **frame 1 layer bg2**. dengan menggunakan brush tool buatlah gambar titik-titik berwarna putih dengan ukuran yang berbeda dan dengan posisi yang acak memenuhi stage.



gambar pada frame 1 layer bg2

3. Seleksi seluruh gambar pada **frame 1 layer bg2**, kemudian convert menjadi **movieclip “background2”**.
4. Double klik movieclip **“background2”**, kemudian seleksi seluruh titik yang ada. Copy titik-titik tersebut kemudian paste-kan tepat di atasnya. Perhatikan gambar berikut:



pengaturan movieclip background2

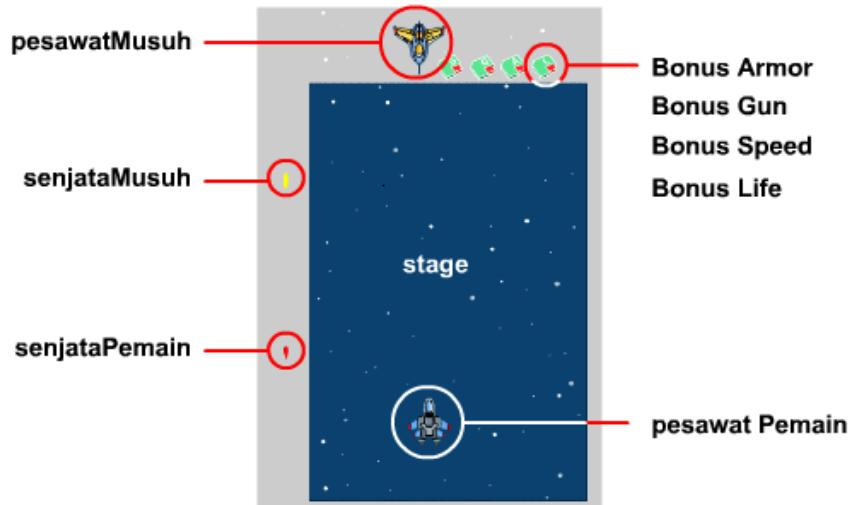
5. Keluar dari mode edit symbol.
6. Klik frame 1 layer bg1. Dengan cara yang sama (langkah no 2 sampai 4), buatlah movieclip "**background1**" dengan ukuran titik yang lebih besar dan posisi yang berbeda.
7. Simpan file.

### Menata Obyek pada Stage

Pada bab memulai pembuatan game, terdapat beberapa tahapan yang akan mempermudah kita dalam membuat game. 4 tahapan pertama sudah kita lakukan yaitu menyusun storyboard, memvisualisasikan storyboard, menggambar karakter dan menggambar background. Tahapan selanjutnya adalah tahapan programming.

Tahapan ini dibagi menjadi beberapa tahap lagi, dan tahapan yang pertama adalah menata obyek pada stage.

Jika anda melakukan proses pembuatan karakter dan pembuatan background dengan benar, maka tugas anda pada tahapan ini relatif mudah yaitu mengatur posisi obyek seperti yang tampak pada gambar berikut:



posisi obyek terhadap stage

### Menambahkan Instance Name

Tahapan selanjutnya adalah menambahkan instance name pada obyek. Perhatikan nama movieclip dan nama instance yang ditambahkan berikut ini:

| movieclip                | instance name |
|--------------------------|---------------|
| pesawat (pesawat pemain) | pesawat       |
| pesawatMusuh             | musuh         |
| laser (senjata pemain)   | laser         |
| senjataMusuh             | laserMusuh    |

### Menambahkan Action

Tahapan yang paling rumit dalam game adalah menyusun action. Apabila kita masih dalam tahap awal belajar action script, maka cara yang tepat untuk melalui tahapan ini adalah dengan menuliskan sedikit script terlebih dahulu, kemudian menguji kinerja action. Bila tidak terjadi kesalahan, maka action ditambahkan lagi dan diuji kembali. Proses tersebut dilakukan secara terus menerus sampai didapatkan detail game.

### Menggerakkan Pesawat Pemain dengan Keyboard

Langkah pertama untuk memulai proses penambahan action pada suatu game yang paling mudah adalah memulai dari karakter utama. Pada game yang akan kita buat, karakter utama adalah pesawat yang dapat digerakkan ke empat arah menggunakan Keyboard. Maka proses penambahan actionnya adalah sebagai berikut :

1. Klik movieclip **pesawat**, kemudian buka panel action dan ketikan script berikut:

```

onClipEvent (load) {
    Mouse.hide();
    xt = _x;
    yt = _y;
}

onClipEvent (enterFrame) {
    //keyboard ditekan
    if (Key.isDown(Key.LEFT) and xt>0) {
        xt -= 30;
    }
    if (Key.isDown(Key.RIGHT) and xt<400) {
        xt += 30;
    }
    if (Key.isDown(Key.UP) and yt>0) {
        yt -= 30;
    }
    if (Key.isDown(Key.DOWN) and yt<600) {
        yt += 30;
    }
    // tracking
    _x = _x+(xt-_x)/_root.kecepatanPesawat;
    _y = _y+(yt-_y)/_root.kecepatanPesawat;
}

```

2. Klik **frame 1 layer action**, kemudian buka panel action dan ketikan script berikut:

```
kecepatanPesawat = 30;
```

3. Simpan file dan jalankan movie. Cobalah untuk menekan tombol panah pada keyboard. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

### **Menembakkan Senjata ketika Tombol Spasi Ditekan**

Langkah selanjutnya adalah menambahkan script agar pesawat dapat menembakkan laser. Selain itu untuk menambah tingkat kesulitan, terdapat sebuah variabel energi yang mana energi tersebut akan berkurang ketika pemain menembakkan senjatanya. Untuk membuatnya kita harus menambahkan script pada 2 obyek yaitu pada movieclip laser dan movieclip pesawat. Perhatikan langkah berikut :

1. Klik movieclip **pesawat**, kemudian buka panel action dan tambahkan script menjadi:

```
onClipEvent (load) {  
    Mouse.hide();  
    xt = _x;  
    yt = _y;  
    jumlahsenjata = 0;  
    energi = 100;  
    kedalamanSenjata = 100;  
}  
  
onClipEvent (enterFrame) {  
    // menambah energi untuk senjata  
    energi++;  
    if (energi>=100) {  
        energi = 100;  
    }  
    //keyboard ditekan  
    if (Key.isDown(Key.LEFT) and xt>0) {  
        xt -= 30;  
    }  
    if (Key.isDown(Key.RIGHT) and xt<400) {  
        xt += 30;  
    }  
    if (Key.isDown(Key.UP) and yt>0) {  
        yt -= 30;  
    }  
    if (Key.isDown(Key.DOWN) and yt<600) {  
        yt += 30;  
    }  
    if (Key.isDown(Key.SPACE)) {  
        //menembak  
        if (energi>=5+2*_root.jenisSenjata and _currentframe == 1) {  
            energi -= 5+2*_root.jenisSenjata;  
            jumlahsenjata++;  
            duplicateMovieClip(_root.laser, "laser"+jumlahsenjata,
```

```

kedalamanSenjata++);

        if (kedalamanSenjata>200) {
            kedalamanSenjata = 100;
            jumlahsenjata = 0;
        }
    }

}

// tracking

_x = _x+(xt-_x)/_root.kecepatanPesawat;
_y = _y+(yt-_y)/_root.kecepatanPesawat;
}

```

2. Klik movieclip **laser**, kemudian buka panel action dan ketikan script berikut:

```

onClipEvent (load) {

    _x = _root.pesawat._x;
    _y = _root.pesawat._y-25;
    jenis.gotoAndStop(_root.jenisSenjata);
}

onClipEvent (enterFrame) {

    if (_name<>"laser") {

        _y -= _root.kecepatanSenjata;
        if (_y<-10) {
            removeMovieClip(this);
        }
    } else {
        _x = -100;
    }
}

```

3. Klik **frame 1 layer action**, kemudian buka panel action dan tambahkan script script menjadi:

```

kecepatanPesawat = 30;
jenisSenjata = 1;

kecepatanSenjata = 10;

```

4. Simpan file dan jalankan movie. Cobalah untuk menekan tombol spasi pada keyboard. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

Penjelasan action :

1. Pada movieclip **pesawat** ditambahkan variabel **jumlahSenjata** yang digunakan untuk menentukan nama dari masing-masing laser yang diduplikasi, variabel **energi** yang digunakan untuk membatasi jumlah laser yang keluar secara bersamaan, dan variabel **kedalamanSenjata** yang menentukan **\_depth** atau kedalaman dari masing-masing duplikasi movieclip laser. (ingat bahwa 1 kedalaman hanya boleh ditempati oleh 1 obyek).
2. Menembakkan laser hanya jika pemain menekan tombol spasi (**if(Key.isDown(Key.SPACE))**) dan ketika pesawat belum meledak atau saat berada pada frame 1 dan energi yang ada cukup untuk menembak (**(if (energi>=5+2\*\_root.jenisSenjata and \_currentframe == 1)**). Bila ketiga kondisi tersebut dipenuhi maka movieclip **laser** diduplikasi.
3. Pada movieclip **laser**, ketika diduplikasi akibat action **duplicateMovieclip(...)**, maka posisi pertama muncul dari movieclip laser adalah tepat di atas movieclip pesawat, dan jenis senjatanya diatur sesuai dengan variabel **jenisSenjata**. (lihat blok movie event **load**).
4. Perintah **if (\_name<>"laser")** berarti movieclip **laser** yang sesungguhnya (bukan hasil duplikasi) tidak akan ditampilkan pada stage.
5. Selanjutnya movieclip **laser** digerakkan ke atas dan dihapus jika keluar dari stage.

## Menggerakkan Musuh

Langkah berikutnya adalah menggerakkan musuh. Sedangkan untuk memperbanyak musuh, cara yang paling efektif adalah dengan menggunakan action **duplicateMovieclip**. Perhatikan proses berikut:

1. Klik movieclip **pesawatMusuh**, kemudian buka panel action ketikan script:

```
onClipEvent (load) {  
    jenis.gotoAndStop(1+random(_root.jenisMusuh));  
    kecepatan = 10+_root.level;  
    _y = -50;  
    _x = 100+random(200);  
}  
  
onClipEvent (enterFrame) {  
    if (_name<>"musuh") {
```

```

_y += kecepatan;
if (_y>650) {
    removeMovieClip(this);
}
//gerakan kekanan dan kekiri acak
if (random(45-_root.level*2)) == 3 or (random(10) == 2 and
_root.pesawat._x>this._x)) {
    arah = 1;
}
if (random(45-_root.level*2)) == 4 or (random(10) == 5 and
_root.pesawat._x<this._x)) {
    arah = 2;
}
if (random(40) == 4) {
    arah = 0;
}
//gerakan acak
if (arah == 1) {
    _x += 5;
} else if (arah == 2) {
    _x -= 5;
}
}

```

2. Klik **frame 1 layer action**, kemudian buka panel action dan tambahkan script menjadi:

```

kecepatanPesawat = 30;
jenisSenjata = 1;
kecepatanSenjata = 10;
level = 1;
jenisMusuh = 2;
musuhMati = 0;
jumlahMusuhKeluar = 0;
kedalamanMusuh = 500;
_root.onEnterFrame = function() {

```

```

// menampilkan musuh secara acak
if (musuhMati<10+level*15 and random(50-_root.level*2) == 5) {
    jumlahMusuhKeluar++;
    if (jumlahMusuhKeluar>=10) {
        jumlahMusuhKeluar = 0;
    }
    duplicateMovieClip(_root.musuh, "musuh"+jumlahMusuhKeluar,
    kedalamanMusuh++);
}

// naik level
if (musuhMati>=10+level*10 and level<20) {
    musuhMati = 0;
    jumlahMusuhKeluar = 0;
    kedalamanMusuh = 500;
    level++;
}
};


```

3. Simpan file dan jalankan movie. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

### **Musuh Menembakkan Senjata**

Untuk membuat musuh dapat menembakkan senjata, kita butuh melakukan beberapa penambahan script pada movieclip senjataMusuh, pesawatMusuh dan pada frame 1 layer action. Perubahan tersebut adalah sebagai berikut:

1. Klik movieclip **senjataMusuh**, kemudian buka panel action dan ketikan script:

```

onClipEvent (enterFrame) {
    if (_name<>"laserMusuh") {
        _y += 15;
        if (_y>610) {
            removeMovieClip(this);
        }
    }
}

```

2. Selanjutnya klik movieclip **pesawatMusuh**, kemudian buka panel action dan tambahkan script script menjadi:

```

.....
//gerakan acak
if (arah == 1) {
    _x += 5;
} else if (arah == 2) {
    _x -= 5;
}
//menembak
if (_currentframe == 1 and random(30-_root.level) == 4) {
    _root.jumlahSenjataMusuh++;
    if (_root.jumlahSenjataMusuh>100) {
        _root.jumlahSenjataMusuh = 0;
    }
    duplicateMovieClip(_root.laserMusuh,
"laserMusuh"+_root.jumlahSenjataMusuh, _root.kedalamanSenjataMusuh++);
    _root["laserMusuh"+_root.jumlahSenjataMusuh]._x = _x;
    _root["laserMusuh"+_root.jumlahSenjataMusuh]._y =
_y+20;
}
}

```

3. Selanjutnya klik **frame 1 layer action**, kemudian buka panel action dan tambahkan script script menjadi:

```

.....
jumlahMusuhKeluar = 0;
kedalamanMusuh = 500;
jumlahSenjataMusuh = 0;
kedalamanSenjataMusuh = 2000;
_root.onEnterFrame = function() {

    // menampilkan musuh secara acak
    if (musuhMati<10+level*15 and random(50-_root.level*2) == 5) {

.....

```

4. Simpan file dan jalankan movie. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya

## Mendeteksi Tumbukan

Sebelum menambahkan action untuk tumbukan, maka kita harus mencari seluruh peluang tumbukan yang terjadi dalam game. Peluang-peluang tumbukan tersebut antara lain:

1. Tumbukan antara **pesawat** dengan **pesawatMusuh**.
2. Tumbukan antara **laser** dengan **pesawatMusuh**.
3. Tumbukan antara **laserMusuh** dengan **pesawat**.

## Mendeteksi Tumbukan antara pesawat dengan pesawatMusuh

Ketika pesawat menabrak pesawatMusuh, maka peluang yang terjadi adalah pesawatMusuh meledak, dan pesawat meledak apabila tidak memiliki armor atau pesawat mengaktifkan armor dan tidak meledak. Untuk membuatnya perhatikan langkah berikut:

1. Klik movieclip **pesawat**, kemudian buka panel action dan tambahkan script menjadi:

```
....  
// tracking  
  
_x = _x+(xt-_x)/_root.kecepatanPesawat;  
_y = _y+(yt-_y)/_root.kecepatanPesawat;  
// setelah meledak  
  
if (ledakan._currentframe == 18 and mati == 0) {  
    //masih memiliki lifepoint  
    if (_root.nyawa>0) {  
        _root.nyawa--;  
        _root.jenisSenjata = 1;  
        _root.armor = 0;  
        energi = 100;  
        jumlahsenjata = 0;  
        gotoAndStop(1);  
    } else {  
        //gameover  
    }  
}
```

```
}
```

2. Selanjutnya klik movieclip **pesawatMusuh**, kemudian buka panel action dan tambahkan script script menjadi:

```
.....  
  
//menembak  
  
if (_currentframe == 1 and random(30-_root.level) == 4) {  
    _root.jumlahSenjataMusuh++;  
    if (_root.jumlahSenjataMusuh>100) {  
        _root.jumlahSenjataMusuh = 0;  
    }  
    _root.suaraLaserMusuh.start(0, 1);  
    duplicateMovieClip(_root.laserMusuh,  
"laserMusuh"+_root.jumlahSenjataMusuh, _root.kedalamanSenjataMusuh++);  
    _root["laserMusuh"+_root.jumlahSenjataMusuh]._x = _x;  
    _root["laserMusuh"+_root.jumlahSenjataMusuh]._y =  
    _y+20;  
}  
  
//tumbukan dengan pesawat  
  
if (hitTest(_root.pesawat) and _currentframe == 1 and  
_root.pesawat._currentframe == 1 and _root.pesawat.perisai._currentframe == 1)  
{  
    if (_root.armor>0) {  
        // pesawat punya armor  
        _root.armor--;  
        _root.pesawat.perisai.gotoAndPlay(2);  
        gotoAndStop(2);  
    } else {  
        // pesawat meledak  
        gotoAndStop(2);  
        _root.pesawat.gotoAndStop(2);  
    }  
}  
  
// setelah meledak
```

```

        if (ledakan._currentframe == 18) {
            gotoAndStop(1);
            removeMovieClip(this);
            _root.musuhMati++;
            _root.score += 100;
        }
    }
}

```

- Selanjutnya klik **frame 1 layer action**, kemudian buka panel action dan tambahkan script script menjadi:

```

.....
jumlahSenjataMusuh = 0;
kedalamanSenjataMusuh = 2000;
score = 0;
armor = 2;
nyawa = 2;
_root.onEnterFrame = function() {
    // menampilkan musuh secara acak
    if (musuhMati<10+level*15 and random(50-_root.level*2) == 5) {
.....

```

- Simpan file dan jalankan movie. Tabrakkan pesawat dengan musuh dan lihatlah apakah proses tumbukan benar. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya

### **Mendeteksi Tumbukan antara laser dengan pesawatMusuh**

Tumbukan selanjutnya yang harus dideteksi adalah tumbukan antara laser dan pesawatMusuh. Untuk membuatnya perhatikan langkah-langkah berikut:

- Klik movieclip **laser**, kemudian buka panel action dan tambahkan script menjadi:

```

.....
if (_y<-10) {
    removeMovieClip(this);
}
//tumbukan dengan pesawat Musuh
for (i=0; this.i<10; i++) {
    if ((hitTest(_root["musuh"]) or hitTest(_root["musuh"+i]))

```

```

        and _root["musuh"+i]._currentframe == 1) {
            _root["musuh"+i].gotoAndStop(2);
            removeMovieClip(this);
        }
    } else {
        _x = -100;
    }
}

```

2. Simpan file dan jalankan movie. Cobalah menembak musuh. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

### **Mendeteksi Tumbukan antara laserMusuh dengan pesawat**

Tumbukan yang ketiga adalah tumbukan antara laserMusuh dan pesawat. Untuk membuatnya perhatikan langkah-langkah berikut:

1. Klik movieclip **senjataMusuh** (movieclip dengan instance name **laserMusuh**), kemudian buka panel action dan tambahkan script menjadi:

```

onClipEvent (enterFrame) {
    if (_name<>"laserMusuh") {
        _y += 15;
        if (_y>610) {
            removeMovieClip(this);
        }
    }
    //tumbukan dengan pesawat
    if (hitTest(_root.pesawat) and _root.pesawat._currentframe == 1
and _root.pesawat.perisai._currentframe == 1) {
        if (_root.armor>0) {
            // pesawat punya armor
            _root.armor--;
            _root.pesawat.perisai.gotoAndPlay(2);
            removeMovieClip(this);
        } else {
            // pesawat meledak
            removeMovieClip(this);
        }
    }
}

```

```

        _root.pesawat.gotoAndStop(2);
    }
}
}
}

```

2. Simpan file dan jalankan movie. Cobalah menahan tembakan musuh. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

## Mengeluarkan Bonus

Setelah proses perhitungan tumbukan di atas, game pada dasarnya sudah dapat dimainkan. Akan tetapi terdapat beberapa elemen lagi yang perlu ditambahkan, salah satunya adalah bonus. Untuk menambahkan bonus perhatikan proses berikut:

1. Klik movieclip bonus armor, kemudian buka panel action dan ketikan script berikut:

```

onClipEvent (load) {
    keluar = 0;
    ya = _y;
    _visible = 1;
    kecepatan = 10+random(10);
}

onClipEvent (enterFrame) {
    // merandom waktu keluar
    if (keluar == 0 and random(500) == 47) {
        keluar = 1;
        _visible = 1;
        _x = 50+random(250);
    }
    if (keluar == 1) {
        _y += 10;
        if (_y>620) {
            _y = ya;
            _visible = 0;
            keluar = 0;
        }
    }
}

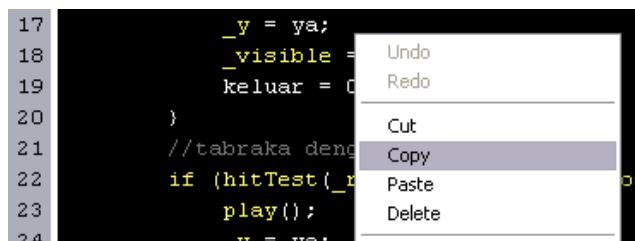
```

```

//tabrakan dengan pemain
if (hitTest(_root.pesawat) and _root.pesawat._currentframe == 1) {
    play();
    _y = ya;
    _visible = 0;
    keluar = 0;
    //dapat bonus
    _root.armor++;
}
}

```

2. Seleksi seluruh action tersebut, kemudian klik kanan dan pilih copy.



mengopy action

3. Klik movieclip **bonus life**, selanjutnya buka panel action dan pastekan action yang sudah anda copy pada langkah no 2.
4. Ubah baris

```
//dapat bonus
_root.armor++;
menjadi
```

```
//dapat bonus
_root.nyawa++;
```

5. Klik movieclip **bonus speed**, selanjutnya buka panel action dan pastekan action yang sudah anda copy pada langkah no 2.

6. Ubah baris

```
//dapat bonus
_root.armor++;
menjadi
```

```
//dapat bonus  
if (_root.kecepatanPesawat>10) {  
    _root.kecepatanPesawat -= 5;  
}
```

7. Klik movieclip **bonus gun**, selanjutnya buka panel action dan pastekan action yang sudah anda copy pada langkah no 2.
8. Ubah baris

```
//dapat bonus  
_root.armor++;  
menjadi
```

```
//dapat bonus  
if (_root.jenisSenjata<4) {  
    _root.jenisSenjata++;  
}  
  
_root.pesawat.energi = 100;
```

9. Simpan file dan jalankan movie. Tunggulah bonus muncul dan tabrakkan pesawat ke bonus, untuk saat ini efek yang terlihat hanya ketika anda mendapatkan bonus gun, sedangkan bonus yang lain tidak terlihat efeknya. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

## Menggerakkan Background

Elemen lain yang perlu ditambahkan adalah gerakan background. Untuk membuatnya perhatikan proses berikut :

1. Klik movieclip “background1”, kemudian buka panel action dan ketikan script berikut:

```
onClipEvent (load) {  
    ya = _y;  
}  
  
onClipEvent (enterFrame) {  
    _y += 10;  
    if (_y>600) {  
        _y = ya;  
    }  
}
```

2. Seleksi seluruh action, klik kanan dan pilih copy.

3. Klik movieclip “background2”, kemudian pastekan action yang sudah anda copy pada langkah no 2.
4. Ubah baris

```
_y += 10;  
menjadi
```

```
_y += 5;
```

10. Simpan file dan jalankan movie. Perhatikan pergerakan background. Apabila tidak terjadi kesalahan lanjutkan dengan proses selanjutnya.

## Menambahkan Sound

Pada bab Sound dijelaskan bahwa elemen sound memegang peranan penting dalam suatu game, karena dapat meningkatkan kerealitasan sebuah game. Setelah kita menyelesaikan tahap-demi tahap di atas, pada dasarnya kita sudah membuat 80% dari total game. Akan tetapi suara yang belum ada adalah suara senjata pemain dan senjata lawan serta suara background (BGM).

Untuk menambahkan suara-suara tersebut, siapkan terlebih dahulu file suara yang dibutuhkan. Dalam contoh ini file suara yang digunakan adalah file “suaraLaser.wav”, “suaraLaser2.wav” dan “bgm2.mp3”. Proses penambahan suara-suara adalah sebagai berikut:

1. Copy file “**bgm2.mp3**” dari CD tutorial ke folder tempat anda bekerja.
2. Importlah file “**“suaraLaser.wav”** dan “**“suaraLaser2.wav”**” dari CD tutorial ke library.
3. Klik kanan “**“suaraLaser.wav”**”, kemudian tambahkan linkage dengan identifier “**“suaraLaser”**”.
4. Klik kanan “**“suaraLaser2.wav”**”, kemudian tambahkan linkage dengan identifier “**“suaraLaser2”**”.
5. Klik **frame 1 layer action**, kemudian buka panel action dan tambahkan script menjadi:

```
....  
score = 0;  
armor = 2;  
nyawa = 2;  
//suara
```

```

bgm = new Sound();
bgm.loadSound("bgm2.mp3", 0);
bgm.setVolume(50);
bgm.start(0, 99);
suaraLaser = new Sound();
suaraLaser.attachSound("suaraLaser2");
suaraLaserMusuh = new Sound();
suaraLaserMusuh.attachSound("suaraLaser");
_root.onEnterFrame = function() {
    // menampilkan musuh secara acak
    if (musuhMati<10+level*15 and random(50-_root.level*2) == 5) {
        .....

```

6. Selanjutnya klik movieclip **pesawat**, buka panel action dan tambahkan script menjadi:

```

.....
if (Key.isDown(Key.SPACE)) {
    //menembak
    if (energi>=5+2*_root.jenisSenjata and _currentframe == 1) {
        _root.suaraLaser.start(0, 1);
        energi -= 5+2*_root.jenisSenjata;
        jumlahsenjata++;
    }
.....

```

7. Kemudian klik movieclip **pesawatMusuh**, buka panel action dan tambahkan script menjadi:

```

.....
//menembak
if (_currentframe == 1 and random(30-_root.level) == 4) {
    _root.suaraLaserMusuh.start(0, 1);
    _root.jumlahSenjataMusuh++;
    if (_root.jumlahSenjataMusuh>100) {
        .....

```

8. Simpan file dan jalankan movie.

Proses pembuatan game Allien attack saat ini sudah memasuki tahapan sekitar 90 % sedangkan tahap 10% selanjutnya dibahas pada bab **Proses Finishing**.

## **Finishing Game**

Setelah anda menyelesaikan action utama dari sebuah game, sebagai contoh anda telah melakukan proses pembuatan game Allien Attack pada bab sebelumnya, tahapan selanjutnya adalah menyelesaikan detail dari game tersebut atau saya istilahkan sebagai proses finishing game. Hal yang harus dilakukan dalam tahap finishing antara lain adalah :

- Menambahkan properti
- Menambahkan cover
- Menambahkan menu utama
- Membuat preloader

### **Menambahkan Properti Game**

Yang dimaksud dengan properti game disini adalah keterangan mengenai variabel-variabel yang harus diketahui oleh pemain. Sebagai contoh variabel yang harus diketahui pada game allien attack adalah variabel level, score, energi senjata, kondisi armor dan jumlah nyawa yang dimiliki.



contoh properti pada game Allien Attack

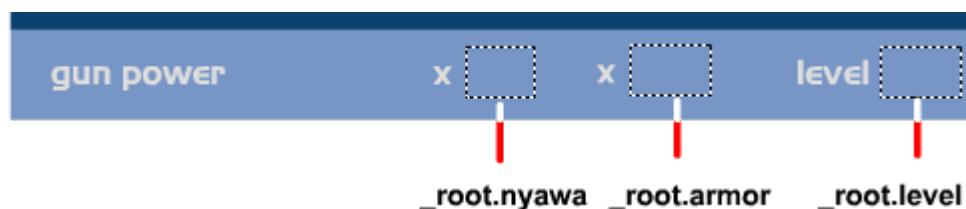
Untuk menambahkan properti game, perhatikan contoh berikut:

1. Lanjutkan file “**alien attack.fla**” yang telah anda buat sebelumnya.
2. Klik **frame 1 layer properti**. Buatlah sebuah kotak yang akan digunakan sebagai background text (lihat gambar diatas) dengan warna lain yang lebih terang (contoh : #809CCA).
3. Kemudian buatlah 4 buah **static text** “**gun power**”, “**x**”, “**x**” dan “**level**”. Atur posisinya seperti pada gambar berikut:



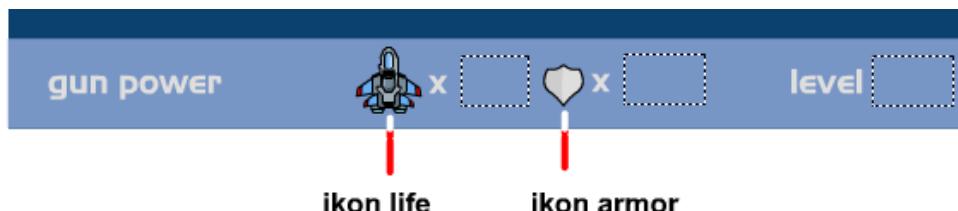
posisi static text

4. Selanjutnya buatlah 3 buah **dynamic text** dengan variabel “**\_root.nyawa**”, “**\_root.armor**” dan “**\_root.level**”. Perhatikan peletakkannya :



posisi dynamic text

5. Kemudian drag movieclip **pesawat**, dan buatlah gambar **perisai**(sebagai ikon dari armor) perhatikan posisinya seperti pada gambar berikut:



peletakan ikon life dan ikon armor

6. Untuk membuat properti gun power, buatlah sebuah kotak berwarna merah seperti pada gambar berikut :



indikator gun power

- Seleksi **fill** kotak tersebut, kemudian convert menjadi **movieclip** dengan nama “**gun power**” . Perhatikan pada pilihan registration pilih tengah kiri.

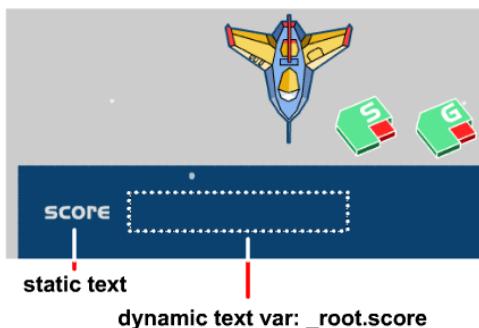


posisi registrasi movieclip “gin power”

- Klik movieclip “**gun power**” tersebut, kemudian buka panel action dan ketikan script:

```
onClipEvent (enterFrame) {
    _xscale = _root.pesawat.energi;
}
```

- Untuk membuat tampilan score buatlah sebuah **static text** “**score**” pada pojok kiri atas stage dan sebuah **dynamic text** disampingnya dengan variabel “**\_root.score**”. Perhatikan gambar berikut :



posisi score pada stage

- Jalankan movie, maka pesawat atau pesawat musuh akan tampak di atas ketika melewati tampilan properti. Hal ini dikarenakan kedalaman (**\_depth**) dari obyek pada layer properti masih di bawah movieclip yang lain, meskipun layer properti berada diatas layer obyek yang lain.
- Untuk memperbaikinya, tutup movie kemudian seleksi seluruh obyek pada frame 1 layer properti, dan convert menjadi movieclip dengan nama “**properti**”. Kembalikan posisi titik registasi ke tengah.
- Klik movieclip **properti**, kemudian buka panel properti dan ketikan “**properti**” pada instance name.
- Klik movieclip **properti**, kemudian buka panel action dan ketikan script berikut :

```
onClipEvent (load) {  
    this.swapDepths(9000);  
}
```

14. Simpan dan jalankan movie.

### Menambahkan Cover

Cover dalam suatu game adalah tampilan yang muncul saat pertama kali file game dijalankan. Penambahan cover yang menarik akan membuat sebuah game menjadi lebih mudah dikenal oleh orang lain (ingat bahwa banyak orang yang menilai bahwa sebuah kesan pertama adalah yang paling diingat) untuk itu buatlah tampilan cover yang sebaik-baiknya. Perhatikan contoh berikut :



cover depan dari game “dig dog”

Untuk membuat tampilan cover yang menarik, anda dapat memanfaatkan software grafis seperti Adobe Photoshop, Firework dan software sejenis lainnya atau anda dapat membuat secara langsung pada stage dengan menggunakan drawing tool. Yang perlu diperhatikan adalah buatlah ukuran gambar sesuai dengan ukuran stage.

### Pemilihan Gambar Ilustrasi untuk Cover

Sebuah desain grafis dapat dibentuk oleh beberapa elemen yaitu warna, gambar ilustrasi, dan typografi. Khusus untuk gambar ilustrasi yang akan kita pakai sebagai cover, harus mewakili isi dari game. Akan lebih baik jika kita menampilkan ilustrasi dengan gaya yang melebih-lebihkan sehingga pemain akan tertarik untuk memainkannya.

Untuk game yang kita buat (alien attack), kita dapat menampilkan karakter pesawat sebagai cover. Selain itu penggayaan desain juga diperlukan dalam sebuah cover game. Perhatikan dua contoh cover untuk game alien attack berikut :



dua desain dengan penggayaan yang berbeda

Pada dua desain tersebut, kemungkinan sebagian besar orang akan menyukai desain sebelah kiri. Akan tetapi membuatnya memang lebih sulit dari pada desain yang kedua. Oleh karena itu pada perusahaan developer game yang sesungguhnya bagian grafis game dibuat oleh tenaga profesional khusus untuk grafis. Meskipun demikian kita yang baru belajar menjadi seorang developer game, harus mampu juga memilih atau membuat sebuah tampilan grafis yang bagus untuk game kita.

### **Pemilihan Typografi untuk Cover**

Pemilihan typografi atau pemakaian text juga menjadi sebuah point penting untuk sebuah desain cover game. Menurut teori typografi, pemilihan bentuk huruf sangat mewakili isi dari suatu desain. Sebagai contoh perhatikan beberapa alternatif huruf untuk game alien attack kita berikut ini:

contoh typografi untuk coer game alien attack

**alien  
attack**

**ALIEN  
ATTACK**

**alien  
attack**

**alien  
attack**

**ALIEN  
ATTACK**

**4LIEEN  
ATTACK**

huruf terpilih dan modifikasinya

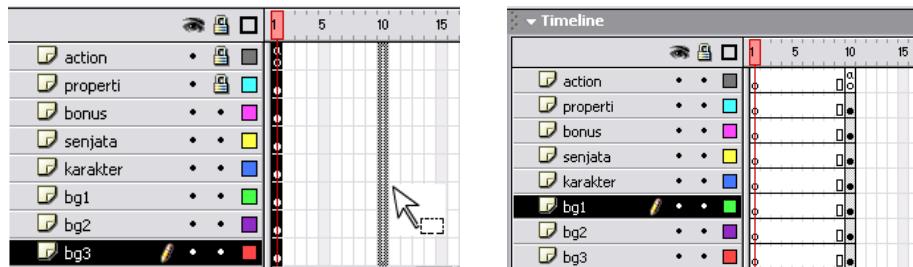
Setelah huruf dimodifikasi, proses selanjutnya adalah menambahkan typografi pada desain yang anda buat. Perhatikan gambar berikut:



peletakan typografi pada desain cover

Setelah file gambar cover game selesai dibuat, simpan file dalam format jpg dan import ke dalam game yang kita buat. Perhatikan contoh contoh berikut:

1. Lanjutkan file “alien attack.fla” yang telah anda buat sebelumnya.
2. Seleksi **frame 1 seluruh layer** (dengan cara : klik frame 1 layer action, kemudian tahan tombol Shift dan klik frame 1 layer bg3).
3. Geser ke **frame 10**. Perhatikan gambar berikut:



menggeser frame dan hasil penggeseran frame

4. Klik **frame 2 layer bg3**, kemudian masukan keyframe.
5. Import file “cover final.jpg” dari CD tutorial ke stage (import file gambar untuk cover buatan anda sendiri), kemudian atur posisinya tepat menutupi stage.
6. Klik **frame 2 layer bg2**, kemudian masukan keyframe.
7. Buatlah sebuah **static text** “**press any key**”, dan letakkan di bagian bawah stage. Kemudian convert teks tersebut menjadi **movieclip** dengan nama “**press**”.

8. Double klik movieclip “press”. Pada mode edit symbol klik **frame 10 layer 1**, kemudian masukan blank keyframe.
9. Klik **frame 20 layer 1**, kemudian masukan frame (F5), sehingga terbentuk animasi teks berkedip. Keluar dari mode edit symbol.
10. Klik movieclip “press”, buka panel action dan ketikan script:

```
onClipEvent (enterFrame) {
    if (Key.isDown()) {
        _root.gotoAndStop("menuUtama");
    }
}
```

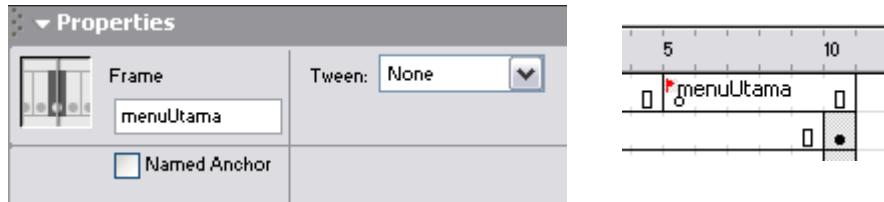
“menuUtama” adalah frame label yang belum kita buat.

11. Klik **frame 2 layer action**, kemudian tambahkan keyframe. Buka panel action dan ketikan script :

```
stop();
```

12. Buatlah sebuah layer baru diatas layer properties dan ubah namanya menjadi **layer label**.

13. Klik **frame 5 layer label**, kemudian masukan keyframe. Buka panel properties dan ketikan “menuUtama” pada frame label.



penambahan frame label

14. Klik **frame 10 layer label**, kemudian masukan keyframe. Buka panel properties dan ketikan “main” pada frame label.

15. Simpan file.

## Menambahkan Menu Utama

Menu utama disini adalah sebuah tampilan yang memiliki beberapa tombol akses untuk memulai game sampai dengan keluar dari game. Biasanya pada menu utama terdapat beberapa tombol yaitu : “start game”, “load game”, “high score”, “credit” dan “quit”.

Perhatikan contoh menambahkan menu utama pada game alien attack berikut :

1. Lanjutkan file “**alien attack.fla**” yang telah anda buat sebelumnya.
2. Klik **frame 5 layer bg2**, kemudian masukan blank keyframe.
3. Buatlah 4 buah **static text** yaitu “**start game**”, “**high score**”, “**game credit**” dan “**quit**” dengan posisi vertikal.



posisi static text

4. Seleksi masing-masing text dan convert menjadi **button** dengan nama “**startGame**”, “**highScore**”, “**gameCredit**” dan “**quit**”.
5. Klik tombol “**startGame**”, kemudian buka panel action dan ketikan scrip berikut:

```
on (release) {  
    gotoAndStop("main");  
}
```

6. Klik tombol “**startGame**”, kemudian buka panel action dan ketikan scrip berikut:

```
on (release) {  
    gotoAndStop("highScore");  
}
```

7. Klik tombol “**gameCredit**”, kemudian buka panel action dan ketikan scrip berikut:

```
on (release) {  
    gotoAndStop("credit");  
}
```

8. Klik tombol “**quit**”, kemudian buka panel action dan ketikan scrip berikut:

```
on (release) {  
    fscommand("quit");  
}
```

9. simpan file.

### Menambahkan Halaman Game Credit

Halaman game credit biasanya berisi tentang informasi pembuat game. Sebagai contoh perhatikan proses penambahan halaman game credit berikut:

1. Lanjutkan file “**alien attack.fla**” yang telah anda buat sebelumnya.
2. Klik **frame 6 layer bg2**, kemudian masukan blank keyframe.
3. Buatlah **static text** yang berisi tentang diri anda, sebagai contoh : “Alien Attack created by Wandah Games. ([www.wandah.com](http://www.wandah.com))”.
4. Selanjutnya klik **frame 6 layer label** dan masukan keyframe. Buka panel properties dan ketikan “**credit**” pada frame label.
5. Klik frame **6 layer bg1** dan masukan keyframe. Buatlah sebuah tombol “**back**”.
6. Klik tombol “**back**”, buka panel action dan ketikan script:

```
on (release) {  
    gotoAndStop("menuUtama");  
}
```

7. Simpan file dan jalankan movie.



halaman Game Credit

### Menambahkan Text Game Over

Jalankan game alien attack yang telah anda tambahkan properti, dan apabila lifepoint 0 dan pesawat meledak, maka permainan akan terhenti. Hal tersebut dikarenakan kita belum membuat tampilan game over. Untuk membuatnya perhatikan langkah-langkah berikut:

1. Lanjutkan file “**alien attack.fla**” yang telah anda buat sebelumnya.

2. Buatlah sebuah **movieclip** baru dengan memilih menu insert>new symbol.. ketikan “**gameOver**” pada nama symbol.
3. Pada mode edit symbol buatlah sebuah **static text** “GAME OVER”.
4. Seleksi text tersebut, kemudian convert menjadi **movieclip** “**teks game over**”.
5. Klik movieclip “**teks game over**”, buka panel properties. Kemudian tambahkan efek alpha 0%.
6. Klik **frame 25 layer 1**, kemudian masukan keyframe.
7. Klik movieclip “**teks game over**”, buka panel properties. Kemudian tambahkan efek alpha 100%.
8. Klik kanan **frame 20** dan pilih Create Motion Tween.
9. Tambahkan 2 buah layer di atas layer 1, dan ubah namanya menjadi **layer tombol** dan **layer action**.
10. Klik **frame 25 layer tombol**, kemudian tambahkan keyframe.
11. Kemudian buatlah 2 buah tombol baru yaitu tombol “**play again**” dan tombol “**submit score**”. Letakkan dibawah movieclip “teks game over”.



obyek pada frame 25

12. Klik tombol “**submit score**”, kemudian buka panel action dan ketikan scrip berikut:

```
on (release) {
    removeMovieClip(this);
    //bersihkan layar
    stopAllSounds();
    for (i=0; i<200; i++) {
        removeMovieClip(_root["musuh"+i]);
        removeMovieClip(_root["laserMusuh"+i]);
        removeMovieClip(_root["laser"+i]);
    }
    removeMovieClip("properti");
    _root.gotoAndStop("submit");
}
```

13. Klik tombol “play again”, kemudian buka panel action dan ketikan scrip berikut:

```
on (release) {  
    removeMovieClip(this);  
    //bersihkan layar  
    stopAllSounds();  
    for (i=0; i<200; i++) {  
        removeMovieClip(_root["musuh"+i]);  
        removeMovieClip(_root["laserMusuh"+i]);  
        removeMovieClip(_root["laser"+i]);  
    }  
    removeMovieClip("properti");  
    _root.gotoAndStop("menuUtama");  
}
```

14. Klik **frame 25 layer action**, kemudian tambahkan keyframe.

15. Buka panel action dan ketikan action:

```
stop();
```

16. Tekan Ctrl+E untuk keluar dari mode edit symbol.

17. Tambahkan linkage pada movieclip “gameOver” dan ketikan “gameover” pada identifier.

18. Klik movieclip **pesawat**, kemudian buka panel action dan tambahkan script menjadi:

```
....  
//masih memiliki lifepoint  
if (_root.nyawa>0) {  
    _root.nyawa--;  
    _root.jenisSenjata = 1;  
    _root.armor = 0;  
    energi = 100;  
    jumlahsenjata = 0;  
    gotoAndStop(1);  
} else {  
    //gameover
```

```
mati = 1;
```

```
    _root.attachMovie("gameover", "gameover", 5000, {_x:200, _y:300});  
}  
}  
}
```

19. Simpan dan jalankan movie. Cobalah untuk mengalah secepatnya untuk menguji text game over.

### Menambahkan Text Next Level

Jalankan game alien attack dan capailah score 2000 secepatnya, maka pada properti level kita akan melihat perubahan dari level 1 ke level 2. Kelemahan dari game ini adalah efek transisi level yang belum ada. Untuk itu perlu ditambahkan sebuah transisi, perhatikan proses berikut:

1. Lanjutkan file “**alien attack.fla**” yang telah anda buat sebelumnya.
2. Buatlah sebuah **movieclip** baru dengan memilih menu insert>new symbol.. ketikan “**nextLevel**” pada nama symbol.
3. Pada mode edit symbol buatlah sebuah **static text** “next level”.
4. Seleksi text tersebut, kemudian convert menjadi **movieclip** “**teks next level**”.
5. Klik movieclip “**teks next level**”, buka panel properties. Kemudian tambahkan efek alpha 0%.
6. Klik **frame 40 layer 1**, kemudian masukan keyframe.
7. Klik movieclip “**teks next level**”, buka panel properties. Kemudian tambahkan efek alpha 100%.
8. Klik kanan **frame 20** dan pilih Create Motion Tween.
9. Tambahkan layer baru di atas layer 1, dan ubah namanya menjadi **layer action**.
10. Klik **frame 40 layer action**, kemudian tambahkan keyframe.
11. Buka panel action dan ketikan action:  

```
removeMovieClip(this);
```
12. Tekan Ctrl+E utuk keluar dari mode edit symbol.
13. Tambahkan linkage pada movieclip “**nextLevel**” dan ketikan “**nextLevel**” pada identifier.
14. Klik **frame 1 layer action**, kemudian buka panel action dan tambahkan script menjadi:  

```
....  
// naik level
```

```
if (musuhMati>=10+level*10 and level<20) {  
    musuhMati = 0;  
    jumlahMusuhKeluar = 0;  
    kedalamanMusuh = 500;  
    level++;  
    attachMovie("nextLevel", "nextLevel", 5001, {_x:200, _y:300});  
}  
};
```

15. Simpan file dan jalankan movie.

Catatan:

Pada dasarnya game Alien Attack tersebut pada dasarnya sudah selesai dibuat, akan tetapi terdapat beberapa hal yang akan dijelaskan pada bab selanjutnya, yaitu penambahan preloader, dan sistem highscore.

## **Publishing**

Sebuah game flash dapat dibedakan menjadi dua berdasarkan cara memainkannya, yaitu secara **online** melalui sebuah situs internet dan secara **offline** ketika dimainkan pada PC. Kedua cara tersebut memiliki sedikit perbedaan yaitu :

- Pada online publishing game ditampilkan pada sebuah internet browser, sedangkan pada offline publishing game dijalankan oleh Windows Projector.
- Pada online publishing, kita membutuhkan sebuah preloader sedangkan pada offline publishing game tidak membutuhkan preloader.

- Pada online publishing kita dapat menyimpan file (seperti menyimpan score) dengan bantuan PHP sedangkan pada offline publishing kita kesulitan dalam menyimpan file seperti score, data pemain dan sebagainya (meskipun demikian menyimpan file secara offline masih bisa dilakukan).

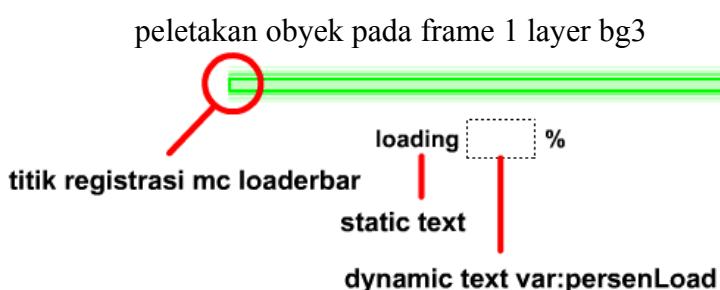
## Preloader

Movie flash secara default pada sebuah internet browser dijalankan secara streaming. Artinya file yang telah didownload akan segera ditampilkan pada browser. Ketika game kita tampilkan pada suatu situs internet, maka game tidak dapat dimainkan secara langsung akan tetapi game ditampilkan secara perlahan satu obyek demi satu obyek tergantung kecepatan modem. Sehingga tampilan akan terlihat kacau (kurang rapi).

Untuk mengatasi hal tersebut, kita membutuhkan sebuah preloader yang akan menunggu hingga seluruh file game terdownload. Dengan menambahkan preloader pemain diharuskan menunggu beberapa saat sampai game siap dimainkan. Perhatikan proses penambahan preloader pada game alien attack berikut.

1. Lanjutkan file “**alien attack.fla**” yang telah kita buat pada bab-bab sebelumnya.
2. Klik **frame 1 layer bg3**. Kemudian buatlah sebuah persegi panjang. (persegi panjang tersebut akan kita pake sebagai indikator proses loading).
3. Seleksi kotak dan convert menjadi **movieclip** dengan nama “**loader bar**”. Pilih tengah kiri pada registration.

Buatlah **static text** “loading:        %” dan **dynamic text** dengan variabel “**persenLoad**”. Perhatikan peletakannya.



4. Klik movieclip “**loaderbar**”, buka panel action dan ketikan script:

```
onClipEvent (load) {
    bytes = 0;
```

```

totalbytes = 0;
}

onClipEvent (enterFrame) {
    bytes = Math.round(_parent.getBytesLoaded()/1024);
    totalbytes = Math.round(_parent.getBytesTotal()/1024);
    _xscale = Math.round((bytes/totalbytes)*100);
    _root.persenload = _xscale;
    if (bytes>=totalbytes) {
        _root.gotoAndStop(2);
    }
}

```

5. Klik **frame 1 layer action**, kemudian buka panel action dan ketikan script:

```

stop();
persenload = 0;

```

6. Simpan dan jalankan movie. Anda tidak akan melihat proses loading, karena ketika dijalankan secara offline komputer seolah-olah telah mendownload seluruh file yang ada.

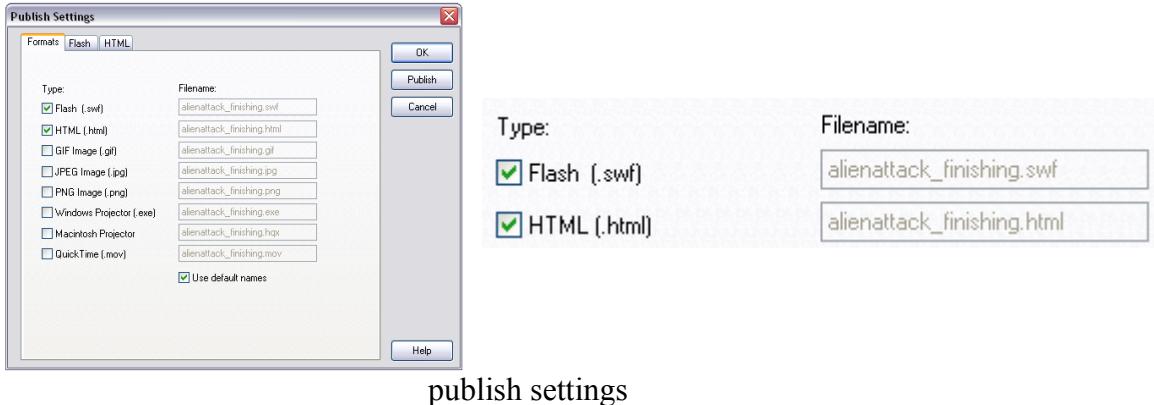
Penjelasan program:

1. Pada saat diload terdapat 2 variabel yaitu variabel **bytes** yang digunakan untuk menentukan jumlah bytes yang telah didownload, dan variabel **totalBytes** yang menunjukkan ukuran total file yang akan didownload.
2. Selanjutnya variabel bytes dihitung dengan action (**bytes = Math.round(\_parent.getBytesLoaded()/1024);**). **Math.round** digunakan untuk membulatkan variabel bytes. **getBytesLoaded()** digunakan untuk menghitung jumlah file yang telah didownload, dan **1024** diperoleh dari perhitungan 1 Kb = 1024 byte.
3. Agar proses loading terlihat, digunakan movieclip “**loaderbar**” yang kondisi skala x (lebar-nya) ditentukan oleh jumlah file yang telah didownload (**\_xscale = Math.round((bytes/totalbytes)\*100);**) dan % angka yang ditampilkan pada dynamic text dengan action (**\_root.persenload = \_xscale;**).
4. Setelah file terdownload seluruhnya, maka game dimulai (**if (bytes>=totalbytes) {...}.**).

## Online Publishing

Agar nantinya game kita dapat dimainkan secara online, maka kita harus mempublish game menjadi 2 tipe file yaitu swf dan html. Perhatikan cara mempublis file berikut:

1. Lanjutkan file “**alien attack.fla**” yang telah kita buat pada bab-bab sebelumnya.
2. Klik menu file>publish settings. Pilih type **flash** dan **HTML**.



publish settings

3. Klik **Publish**, dan proses publishing akan berjalan. Setelah selesai, maka akan terbentuk 2 file yaitu file bertipe swf dan file bertipe html

Perhatikan bahwa ketika anda menjalankan file HTML, movie akan ditampilkan secara default pada pojok kiri atas browser. Untuk membuat tampilan yang lebih menarik, anda membutuhkan program web developer semacam dreamweaver.

## Offline Publishing

Untuk kebutuhan game yang dimainkan offline, pilih type Windows Projector dan klik publish. File yang terbentuk adalah file bertipe exe, sehingga file tersebut dapat dijalankan pada seluruh computer meskipun flash player belum terinstall.



mempublish dengan type exe

## **High Score**

Sebuah game flash online tidak akan lengkap apabila tidak memiliki fitur high score. Dengan penambahan fitur ini, pemain akan tertantang untuk menjadi yang terbaik sehingga akan memainkan game beberapa kali.

Berbeda dengan bahas pemrograman lain seperti Visual C++, Visual Basic, atau Delphi yang dilengkapi dengan script operasi file, Flash tidak memiliki script untuk mengubah atau menyimpan data. Untungnya kelemahan tersebut dapat diatasi dengan menggabungkan flash dengan program lain, salah satunya adalah PHP.

## Menginstal PHP

Sebelum memasuki proses pembuatan high score kita harus mengubah sedikit proses kerja kita. Pada bab sebelumnya dijelaskan bahwa highscore dengan PHP tidak bisa dijalankan secara offline. Disisi lain proses kerja kita selama ini adalah secara offline, untuk itu dalam pembuatan highscore kita harus membuat komputer kita “seolah-olah” online dengan cara menjalankan program php.

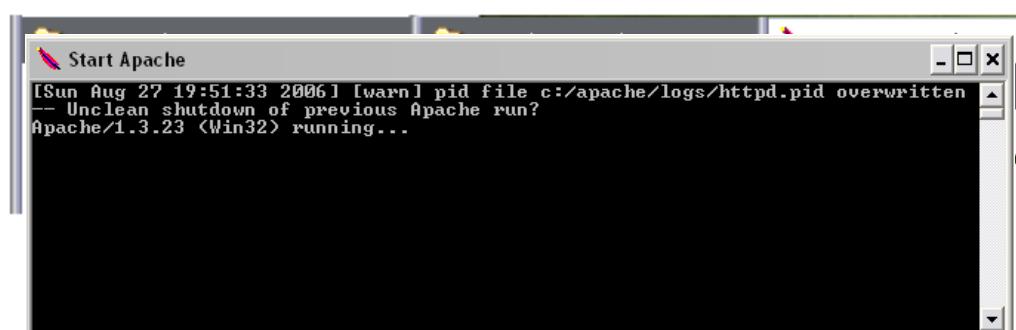
Untuk menginstal PHP anda membutuhkan file phpriad.exe yang mana pada contoh di buku ini saya menggunakan file phpriad versi 2.2.1. Untuk menginstalnya, jalankan file **phpriad2-2-1.exe**.



ikon file phpriad2-2-1

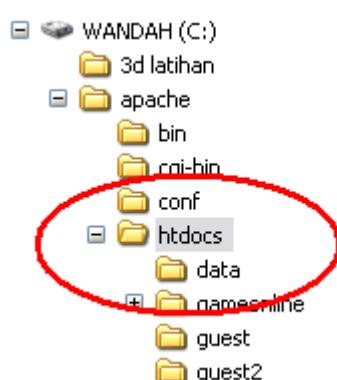
Setelah selesai menginstal, pada start menu akan terdapat aplikasi baru yaitu PHP triad. Agar komputer kita “seolah-olah” online, terdapat 3 hal yang harus dilakukan yaitu:

1. Jalankan program PHP triad dengan menekan windows **start menu>all program>PHP triad>Apache console> start Apache**.



menjalankan Apache  
tampilan ketika Apace dijalankan

2. Seluruh proses kerja anda harus berada pada **folder C:\Apache\htdocs**.



folder tempat kita bekerja

- Untuk menjalankan program publis kedalam format HTML, buka internet browser kemudian ketikan “**http://localhost/nama\_file\_yang\_akan\_dijalankan.html**“ pada alamat browser. (syarat : flash player harus sudah terinstall pada internet browser anda).

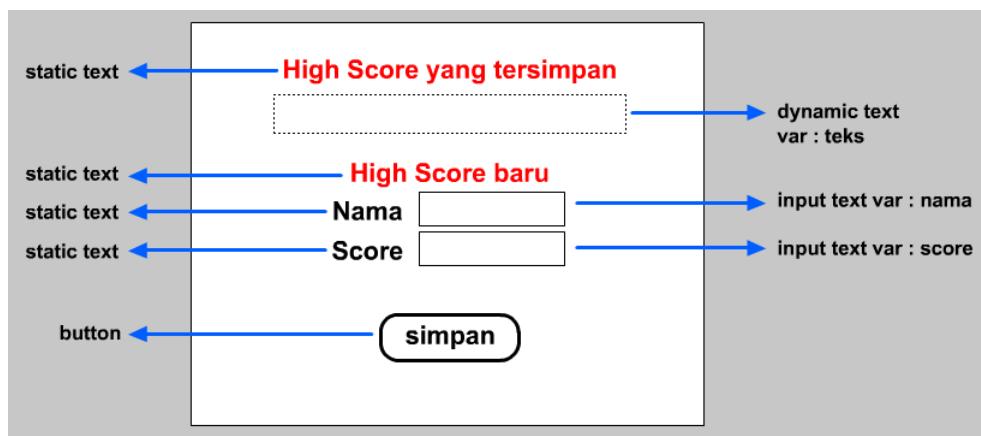


menjalankan file html

### Menyimpan Data Ke dalam File TXT

Pada contoh berikut, kita akan mencoba membuat program sederhana untuk menyimpan sebuah variabel kedalam sebuah file teks (\*.txt). Latihan ini sangat berguna untuk memahami proses penyimpanan highscore pada suatu game flash. Perhatikan langkah berikut :

- Buatlah sebuah file baru dengan ukuran 800 x 600 pixel dan 12 fps.
- Buatlah 2 buah layer baru dan ubah nama masing-masing layer **menjadi layer teks, layer label dan layer action**.
- Klik **frame 1 layer teks** kemudian buatlah obyek seperti pada gambar berikut :



obyek pada frame 1 layer text

- Klik **frame 1 layer label**, kemudian buka pane properties dan ketikan “**loading**” pada frame label.
- Klik **frame 1 layer action**, buka panel action dan ketikan script :

```
stop();
acak = Math.random()*1000000;
```

```

dataKu = new LoadVars();
dataKu.load("data.txt?"+acak, dataKu, "POST");
dataKu.onLoad = function() {
    teks = dataKu.namaPemain+" score "+dataKu.scorePemain;
}

```

6. Klik tombol “simpan”, buka panel action dan ketikan script:

```

on (release) {
    gotoAndStop("simpan");
}

```

7. Klik **frame 10 layer label**, kemudian masukkan keyframe. Buka panel properties dan ketikan “simpan” pada frame label.

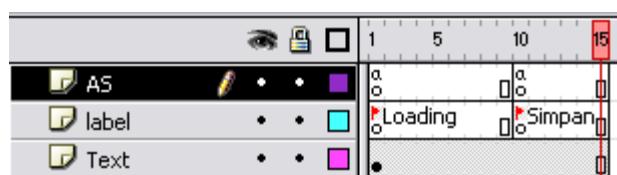
8. Klik **frame 10 layer action**, kemudian maukan keyframe. Buka panel action dan ketikan script :

```

stop();
acak = Math.random()*1000000;
dataKu = new LoadVars();
dataKu.load("simpan.php?"+acak+"&nama="+nama+"&score="+score, dataKu,
"POST");
dataKu.onLoad = function() {
    gotoAndPlay("Loading");
}

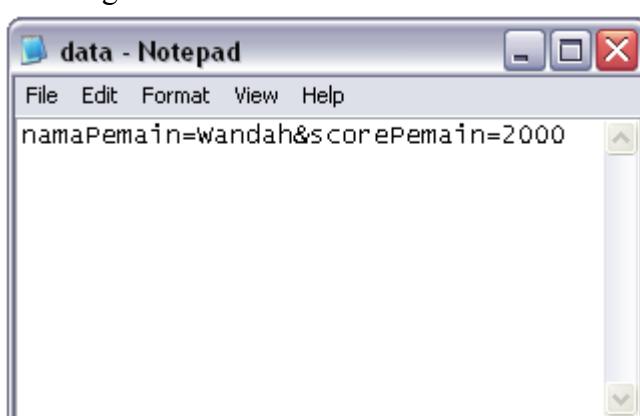
```

9. Simpan file pada folder “**C:/apache/htdocs**” dengan nama “**highScore.fla**”, dn publish menjadi type swf dan type HTML.



susunan layer file highScore.fla

10. Selanjutnya buka program **Notepad**. Ketikan “**namaPemain=Wandah&scorePemain=2000**” dan simpan pada folder **C:/apache/htdocs** dengan nama “**data.txt**”.



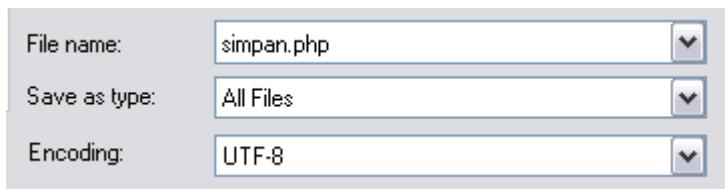
file data.txt

11. Buat file baru pada Notepad dengan memilih menu **file>new**.

12. Selanjutnya ketikan script PHP berikut:

```
<?
$data_anda="namaPemain=".$nama."&scorePemain=".$score;
$file01=fopen("data.txt",w);
fputs($file01,$data_anda);
fclose($file01);
?>
```

13. Simpan simpan pada folder **C:/apache/htdocs** dengan nama “**simpan.php**” .



menyimpan file dengan format selain TXT pada Notepad

14. Jalankan program Apache dengan menekan windows **start menu>all program>PHP triad>Apache console> start Apache**.

15. Buka internet browser (sebagai contoh saya menggunakan Internet Explorer), kemudian ketikan alamat “**http://localhost/highScore.html**”. Jika proses benar, maka anda dapat memasukan nama baru dan score baru kemudian menyimpannya.

Catatan :

Selain menggunakan Notepad, anda juga dapat menggunakan program web builder semacam Dreamweaver.

Ketika anda telah meng-upload file swf dan file HTML tersebut kedalam sebuah situs, maka penulisan alamat tanpa diikuti teks “**localhost**”. Sebagai contoh setelah file tersebut saya upload ke situs saya, maka untuk mengaktifkannya saya tinggal menuliskan alamat “**www.wandah.com/highscore.html**”

Jika anda bekerja pada folder baru di dalam folder C:/apache/htdocs, maka penulisan alamat harus diikuti dengan nama folder tersebut. Contoh : saya menyimpan file kerja saya pada folder **C:/apache/htdocs/latihanHighscore**, maka alamat yang saya ketikan pada browser adalah

**“<http://localhost/latihanHighscore/highScore.html>”.**

Penjelasan program:

Pada file highscore.fla

1. Baris **acak = Math.random()\*1000000;** akan menghasilkan sebuah bilangan acak. Bilangan ini dipakai seolah-olah sebagai sebuah nomor pendaftaran pengguna suatu situs. Mengingat banyaknya pengguna internet pada waktu yang bersamaan, kemungkinan file yang kita publish didownload bersamaan sangat besar. Untuk itu dibutuhkan sebuah identitas yang membedakan antara pengguna satu dan pengguna yang lain.
2. baris **dataKu.load("data.txt?"+acak, dataKu, "POST");** akan membuka file **data.txt** yang telah kita buat dan membaca variabel yang ada didalamnya.
3. Selanjutnya variabel yang telah diload ditampilkan pada dynamic text dengan action “**teks = dataKu.namaPemain+" score "+dataKu.scorePemain;**”.
4. Ketika tombol “**simpan**” ditekan maka action pada frame berlabel **simpan** dijalankan.
5. Untuk menyimpan file digunakan action  
**“dataKu.load("simpan.php?"+acak+"&nama="+nama+"&score="+score, dataKu, "POST");”**. Baris “**simpan.php?"+acak** akan menjalankan file **simpan.php** yang kita buat sebelumnya. Baris “**+"&nama="+nama+"&score="+score**” disebut sebagai parameter crossing dari flash ke PHP.

Pada file data.txt

Teks “**namaPemain=Wandah**” merupakan deklarasi sebuah variabel bernama **namaPemain** dan bernilai **Wandah**. Selanjutnya untuk membuat variabel baru harus dipisah dengan tanda “**&**”.

Pada file simpan.php

1. Baris **<? dan ?>** merupakan tag identitas dari sebuah program php.
2. Pada baris **\$data\_anda="namaPemain=".\$nama."&scorePemain=".\$score;** Karakter “**\$**” dalam php digunakan untuk mendeklarasikan sebuah variabel. Sedangkan karakter “**.**” pada baris tersebut berarti sebuah penambahan.

| Akibat                                                                                            | dari | action |
|---------------------------------------------------------------------------------------------------|------|--------|
| “ <b>dataKu.load("simpan.php?"+acak+"&amp;nama="+nama+"&amp;score="+score,</b>                    |      |        |
| <b>dataKu, "POST");</b> ” pada flash, sebagai contoh bila kita mengetikan nama “ <b>Rosy</b> ”    |      |        |
| dan nilai “ <b>2500</b> ” pada saat movie dijalankan maka nilai variabel <b>data_anda</b> berubah |      |        |
| menjadi “ <b>namaPemain=Rosy&amp;scorePemain=2500</b> ”;                                          |      |        |
| 3. Baris <b>\$file01=fopen("data.txt",w);</b> berarti kita mengakses file <b>data.txt</b> dengan  |      |        |
| perintah menulis file( <b>w</b> ).                                                                |      |        |
| 4. Baris <b>fputs(\$file01,\$data_anda);</b> akan menambahkan nilai dari variabel                 |      |        |
| <b>data_anda</b> kedalam file <b>data.txt</b> .                                                   |      |        |
| 5. Setelah selesai melakukan operasi file, dalam setiap pemrograman file harus                    |      |        |
| ditutup kembali. Pada php proses tersebut menggunakan script <b>fclose(\$file01);</b> .           |      |        |

Dengan memahami proses file tersebut, anda akan dapat mengaplikasikannya kedalam sebuah game.

### **Menyimpan Data Ke dalam File XML**

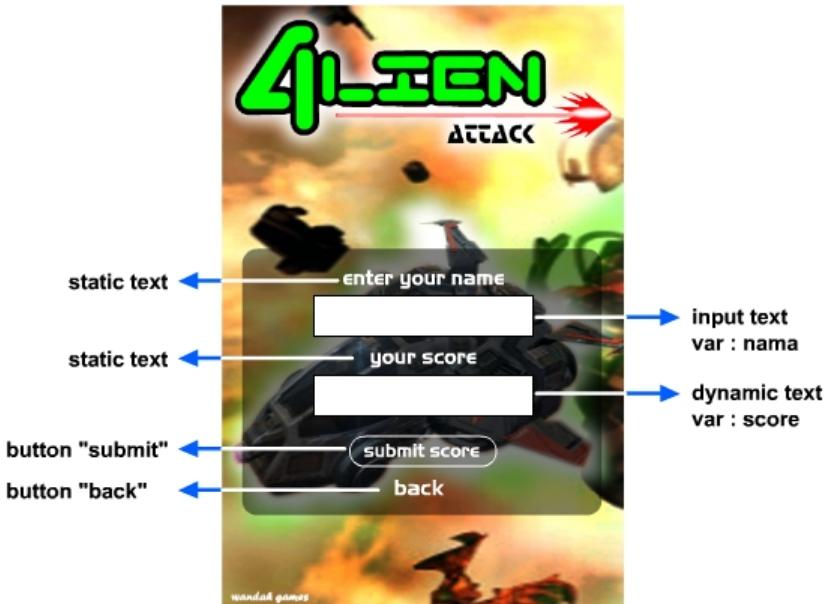
Kelemahan dari program sebelumnya (highScore.fla) adalah ketika kita menggunakan banyak variabel -sebagai contoh kita menyimpan 10 data pemain dengan nilai terbesar, maka kita harus mengakses 20 variabel yaitu 10 variabel nama dan 10 variabel score- kesulitan akan kita temui. Untuk itu diperlukan sebuah cara mengolah dan menyimpan data dengan praktis.

mySQL adalah salah satu penyimpan database yang sering dipakai. Akan tetapi menggunakan penggabungan antara Flash dan mySQL bagi pemula cukup rumit. Untuk itu pada buku ini saya menjelaskan prose pengelolahan database yang lebih mudah dengan menggunakan file bertipe **XML**. Mengelola data dengan XML lebih mudah karena Flash memiliki action yang mensupport file bertipe XML ini.

Perhatikan contoh berikut :

1. Copy file “**alien attack.fla**” yang telah anda buat sebelumnya ke folder **C:/apache/htdocs**. Kemudian buka file tersebut. Kita akan menambahkan sebuah fitur highscore 10 pemain teratas.
2. Klik **frame 30 seluruh layer**, kemudian tambahkan frame (tekan F5).
3. Klik **frame 15 seluruh layer**, kemudian masukan blank keyframe (tekan F7).

4. Klik **frame 15 layer bg3**, kemudian drag bitmap “**cover final.jpg**” dari library ke stage. Letakkan tepat menutupi stage.
5. Klik **frame 15 layer bg2**, kemudian buatlah gambar kotak berwarna gelap di tengah stage (lihat gambar). Kotak ini kita pakai sebagai background untuk text agar terlihat jelas.
6. Klik frame **15 layer bg1**, kemudian buatlah obyek seperti pada gambar berikut :
7. buatlah obyek seperti pada gambar berikut :



obyek pada frame 15

8. Klik **button “submit”**, kemudian buka panel properties dan ketikan “**submit**” pada instance name.
9. Klik button “**back**”, kemudian buka panel action dan ketikan script:

```
on (release) {
    gotoAndStop("menuUtama");
}
```

10. Klik **frame 15 layer label**, kemudian buka panel properties dan ketikan “**submit**” pada frame label.

11. Klik **frame 15 layer action**, kemudian buka panel action dan ketikan script :

```
stop();
dataXML1 = new XML();
dataXML1.ignoreWhite = true;
penampungXML = new XML();
acak = Math.random()*100000;
```

```

dataXML1.load("datapemain.xml?uniq="+acak);
dataXML1.onLoad = function() {
    this.contentType = "text/xml";
};

penampungXML.onLoad = function() {
    this.contentType = "text/xml";
};

submit.onRelease = function() {
    acak = Math.round(Math.random()*10000);
    dataXML1.load("datapemain.xml?uniq="+acak);
    //menyimpan data

    dataXML1.firstChild.appendChild(dataXML1.createElement("datapemain"));

    dataXML1.firstChild.lastChild.appendChild(dataXML1.createElement("nama"));

    dataXML1.firstChild.lastChild.lastChild.appendChild(dataXML1.createTextNode(
        nama));
}

dataXML1.firstChild.lastChild.appendChild(dataXML1.createElement("score"));

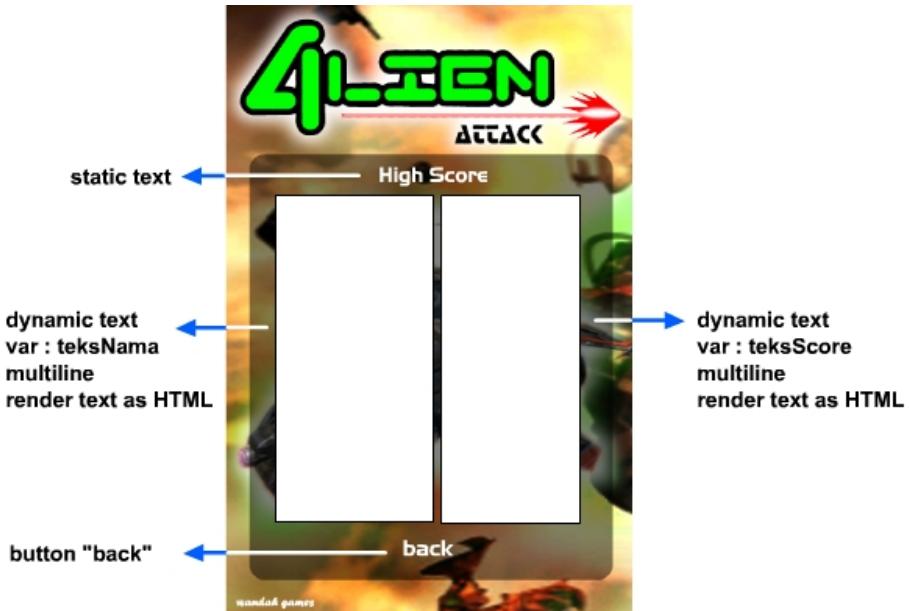
dataXML1.firstChild.lastChild.lastChild.appendChild(dataXML1.createTextNode(
    score));
//akses php
dataXML1.sendAndLoad("XML.php", penampungXML);
gotoAndStop("highScore");
};

```

12. Selanjutnya klik **frame 20 layer bg2**, dan masukan blank keyframe.

13. Buatlah gambar kotak berwarna gelap di tengah stage (lihat gambar).

14. Klik **frame 20 layer bg1**, dan masukan blank keyframe. Buatlah obyek seperti pada gambar berikut:



obyek pada frame 20

15. Klik **frame 20 layer label**, kemudian masukan keyframe. Buka panel properties dan ketikan “highScore” pada frame label.

16. Klik **frame 20 layer action**, kemudian masukan keyframe. Buka panel action dan ketikan script:

```

stop();

dataXML2 = new XML();
dataXML2.ignoreWhite = true;
acak = Math.random()*100000;
dataXML2.load("datapemain.xml?uniq="+acak);
dataXML2.onLoad = function() {
    // baca data file
    var jumlahPemain = dataXML2.firstChild.childNodes.length;
    scorePemain = [];
    namaPemain = [];
    xnama = "";
    for (i=0; i<jumlahPemain; i++) {
        scorePemain[i] = Number(dataXML2.firstChild.childNodes[i].childNodes[1].firstChild.nodeValue);
        namaPemain[i] =
    }
}

```

```

dataXML2.firstChild.childNodes[i].childNodes[0].firstChild.nodeValue;
}

//hitung highscore
for (i=0; i<=jumlahPemain-1; i++) {
    minIndex = i;
    for (j=i+1; j<=jumlahPemain; j++) {
        if (scorePemain[j]>scorePemain[minIndex]) {
            minIndex = j;
        }
    }
    if (minIndex>i) {
        k = scorePemain[i];
        scorePemain[i] = scorePemain[minIndex];
        scorePemain[minIndex] = k;
        xnama = namaPemain[i];
        namaPemain[i] = namaPemain[minIndex];
        namaPemain[minIndex] = xnama;
    }
}
// tampilkan 10 highscore pada dynamic text
for (i=0; i<10; i++) {
    teksNama += namaPemain[i]+\n";
    teksScore += scorePemain[i]+\n";
}
};

```

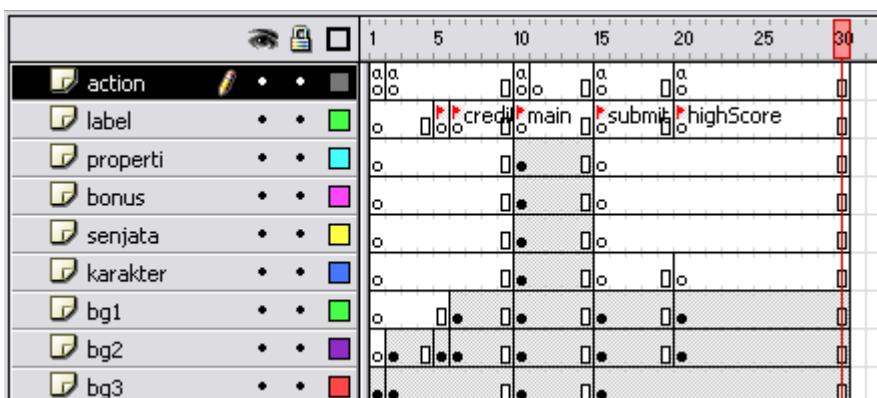
17. Klik button “**back**”, kemudian buka panel action dan ketikan script:

```

on (release) {
    gotoAndStop("menuUtama");
}

```

18. Simpan file dan publish menjadi type swf dan HTML.



susunan akhir file alien attack.fla

19. Buka program **Notepad**. Ketikan script XML berikut:

```
<datapemain>
  <datapemain>
    <nama>Wandah</nama>
    <score>1000</score>
  </datapemain>
</datapemain>
```

20. Simpan dengan nama “**dataPemain.xml**” pada folder **C:/apache/htdocs**.

21. Pada Notepad, buatlah file baru dan ketikan script PHP berikut:

```
<?php
$file = fopen("datapemain.xml", "w+") or die("file tidak ada");
$xmlString = $HTTP_RAW_POST_DATA;
if(!fwrite($file, $xmlString)){
  print "kesalahan operasi file";
}
print $xmlString."\n";
fclose($file);
?>
```

22. Simpan dengan nama “**XML.php**” pada folder **C:/apache/htdocs**.

23. Jalankan program Apache, kemudan buka program internet browser dan ketikan “**http://localhost/alien attack.html**”. Jalankan game, setelah game berakhir tekan tombol submit score, maka akan terlihat daftar high score.

Penjelasan program:

Pada file dataPemain.XML

Pada dasarnya sebuah file XML memiliki format penulisan yang sama seperti HTML, hanya saja pada type XML kita dapat mendefinisikan sendiri tag-tag yang dipakai.

Perhatikan file tersebut :

The diagram illustrates the memory structure of the XML document. It shows the XML code on the left and its corresponding memory representation on the right. The nodes are represented by gray boxes, and their memory addresses are shown in red text. A red circle highlights the node value '1200' under the second child node of the second main node, which has an address of firstchild.childnode[1].childnode[0]. A red arrow points from the word 'nodeValue' at the bottom to this circled node value.

<datapemain>	firstchild
<datapemain>	firstchild.childnode[0]
<nama>Wandah</nama>	firstchild.childnode[0].childnode[0]
<score>1000</score>	firstchild.childnode[0].childnode[1]
</datapemain>	
<datapemain>	firstchild.childnode[1]
<nama>Tiok</nama>	firstchild.childnode[1].childnode[0]
<score>1200</score>	firstchild.childnode[1].childnode[1]
</datapemain>	
</datapemain>	

## struktur file bertipe xml

Pada file Alien Attack.fla

Pada frame 15 layer, action digunakan sebagai alat penyimpan :

1. baris **dataXML1 = new XML();** digunakan untuk mendefinisikan sebuah obyek baru bertipe **XML**.
2. **penampungXML = new XML();** obyek **penampungXML** digunakan untuk menyimpan data XML sementara sebelum disimpan kedalam sebuah file.
3. Baris  
**dataXML1.firstChild.lastChild.appendChild(dataXML1.createElement("na  
ma"));** digunakan untuk membuat sebuah **tag** baru dalam sebuah file XML.
4. Baris  
**dataXML1.firstChild.lastChild.lastChild.appendChild(dataXML1.createTextNode  
Node(nama));** digunakan untuk memasukan sebuah nilai **nodeValue**.
5. Setelah semua tag dibuat, selanjutnya data dikirim untuk diproses oleh program PHP dengan menggunakan perintah **dataXML1.sendAndLoad("XML.php",  
penampungXML);**

Pada frame 20 layer, action digunakan sebagai alat membaca data dan mengurutkan data dari yang terbesar :

1. Baris **dataXML2.load("datapemain.xml?uniq="+acak);** digunakan untuk meng-load file **dataPemain.xml**.
2. Selanjutnya data file dibaca dan disimpan dalam variabel bertipe array **namaPemain** dan **scorePemain**. (baris dibawah komentar // **baca data file**)
3. Selanjutnya dilakukan pengurutan data dari yang tertinggi dengan menggunakan **operasi for bertingkat** (prinsip sederhananya adalah jika data ke i+1 lebih besar dari data ke i maka data ditukar nilainya).
4. High score ditampilkan sebanyak 10 data pada sebuah dynamic text. Akibat pemilihan option **render text as HTML** pada dynamic text, script pada baris **teksNama += namaPemain[i]+\n";** berarti variabel teksNama ditambah **namaPemain[i]** dan ditambah **enter** (baris baru)-sehingga data akan tersusun rapi ke arah bawah.

Pada file XML.php

1. Sebelum melakukan operasi file, file **dataPemain.xml** dibuka terlebih dahulu dengan perintah **\$file = fopen("datapemain.xml", "w+");**
2. Baris **\$xmlString = \$HTTP\_RAW\_POST\_DATA;** merupakan deklarasi variabel **xmlString** yang nilainya sama dengan data yang tersimpan pada variabel **penampungXML** (di dalam file **alien attack.fla frame 15 layer action**).
3. Selanjutnya data disimpan dengan action **print \$xmlString."\n";**

Dan setelah data disimpan, file ditutup kembali dengan action **fclose(\$file);**.

Catatan :

Ketika bekerja dengan PHP, kita sering kali melakukan kesalahan penulisan yang mengakibatkan terhapusnya data didalam file yang kita akses. Sebagai contoh ketika kita game alien attack kita coba mainkan dan menekan tombol submit score, isi file “dataPemain.xml” tiba tiba hilang (ditandai dengan ukuran file 0 Kb-lihatlah pada windows eksplorer). Untuk itu sebelum mencoba game, buka windows eksplorer, copy terlebih dahulu file “dataPemain.xml” kemudian pastekan sehingga terbentuk file “copy of dataPemain.xml”. Langkah ini bertujuan agar kita tidak membuat file berulang kali.



mengcopy file yang rawan terhapus

Menyimpan file baik menjadi file bertipe TXT maupun XML hanya bekerja pada sebuah computer yang online (setelah movie diupload ke sebuah situs). Ketika file tersebut dijalankan pada sebuah Windows Projector (secara offline), operasi file tidak dapat dilakukan, sehingga data tidak akan bisa berubah.

## **Mensubmit Game Buatan Sendiri ke Situs Internet**

Saat ini situs yang menampilkan flash game sangat banyak. Dari data terakhir saat buku ini ditulis, jika kita memasukan keyword “flash game” ke sebuah search engine semacam google.co.id akan didapatkan hasil lebih dari 10.800.000. Dari jumlah yang banyak itu sudah dapat dipastikan terdapat beberapa situs flash yang terkenal. Situs-situs tersebut antara lain:

1. [www.miniclip.com](http://www.miniclip.com)
2. [www.flashgame.net](http://www.flashgame.net)
3. [www.flashgames247.com.](http://www.flashgames247.com)
4. [www.freeonlinegames.com](http://www.freeonlinegames.com)
5. [www.orisinal.com](http://www.orisinal.com)
6. [www.gotoandplay.it](http://www.gotoandplay.it)
7. [www.wandah.com](http://www.wandah.com)
8. [www.games.babaflash.com](http://www.games.babaflash.com)
9. [www.arkadium.net](http://www.arkadium.net)
10. [www.adobe.com/developer/exchange.html](http://www.adobe.com/developer/exchange.html) dan sebagainya.

Dari beberapa situs tersebut kita dapat mengirimkan game final karya kita. Beberapa diantaranya memberikan loyalti dengan kisaran antara 50 US\$ samapai 200 US\$. Sedangkan beberapa karya lainnya bersedia menampilkan profil developernya-hal seperti inipun juga sangat menguntungkan, karena berdasarkan sedikit pengalaman saya-anda bisa mengumpulkan 100US\$ perbulan hanya dengan menyisihkan 2 sampai 3 jam per hari -(6 hari dalam 1 minggu- karena minggu itu hari libur)- waktu anda didepan komputer. Itulah inti dari buku ini saya tulis, selain **membuat game itu menyenangkan** ternyata membuat game **juga menguntungkan**. Selain itu saat ini

game flash juga dapat dimainkan dalam sebuah mobile phone, sehingga peluang pasar jauh lebih besar.

Untuk lebih mendalami game flash berbagai type bacalah buku “Game programming series : board game”, “seri action game”, “seri arcade game”, “seri Racing Game”, “seri Tiling Game”, “seri Fighting”, “seri RPG”, “seri RTS”, “seri Flashlite : Flash Mobile Game” dan “Memainkan Game Flash pada PS2 dan XBOX” yang merupakan kelanjutan dari buku ini.

### **FSCommand**

Fscommand pada flash memiliki 5 perintah yang paling sering dipakai dalam game, yaitu:

1. **fscommand(“showmenu”, “false”);** digunakan untuk menghilangkan tampilan menu. Akan tetapi ketika klik kanan mouse ditekan tampilan menu masih muncul yaitu setting dan about.
2. **fscommand(“fullscreen”, “true”);** digunakan untuk menampilkan moviesecara penuh atau tetap pada ukurannya jika nilai true diubah menjadi false.
3. **fscommand(“allowscale”, “true”);** digunakan agar movie mengikuti perubahan ukuran projector.
4. **fscommand(“trapallkeys”, “true”);** digunakan untuk mengunci tombol keyboard. Akibat perintah ini tombol keyboard hanya bisa diakses oleh action Key.
5. **fscommand(“quit”);** digunakan untuk keluar dari aplikasi.

Untuk mengaplikasikannya klik frame 1 layer action di setiap program yang anda buat, kemudain ketikan action fscommand yang anda butuhkan.Selain 5 bentuk tersebut, action fscommand juga dapat dimanfaatkan sebagai jembatan penghubung antara program flash dengan program lain seperti java, mdm zinc dan sebagainya.

### **Mencegah SWF decompiler**

Ketika software SWF decompiler dipasarkan (salah satu yang terbaik adalah Sothink SWF decompiler) banyak developer game flash kecewa, karena karyanya akan lebih mudah dibajak oleh orang lain (istilah yang populer adalah re-skinning). Pada forum-

forum flash, para developer mengeluh dan berusaha untuk mencari solusi mencegah decompiler.

Bagaimanapun juga file swf sangat mudah di decompiler menjadi file fla. Hal tersebut tidak bisa kita cegah- meskipun demikian terdapat beberapa software yang dapat melindungi karya anda dari decompiler semacam **swf locker**. Meskipun demikian cara yang terbaik untuk melindungi karya kita dari decompiler adalah dengan mempublish file menjadi bertipe exe dengan menggunakan software **swf to projector**. Akan tetapi game kita tidak akan bisa dimainkan secara online lagi- walaupun demikian game masih dapat dijual dengan harga yang lebih tinggi. Cara tersebut adalah tren yang dilakukan oleh pada developer flash game pada saat buku ini ditulis.

Sofware swf to projector yang saya rekomendasikan adalah **MDM Zinc** dan **MProjector**. Atau jika anda tidak memiliki keduanya anda bisa menggunakan program editor **ResHacker**. Dengan salah satu software tersebut kita dapat melakukan beberapa hal yaitu:

1. Melindungi karya dari decompiler.
2. Mengubah ikon flash player dengan ikon buatan sendiri.
3. Menghilangkan seluruh tampilan menu saat tombol klik kanan mouse ditekan.
4. Mengubah versi file dan menyembunyikan text Macromedia Flash / Adobe Flash Player.
5. Menyimpan file-meskipun game dijalankan offline (fitur ini terdapat pada program MDM Zinc).

### **Menggunakan SWF decompiler**

Meskipun decompiler dianggap sebagai musibah oleh beberapa developer, namun decompiler juga dianggap sebagai durian jatuh bagi developer pemula. Kita dapat mengambil game buatan orang lain dari sebuah situs internet menggunakan software khusus (contoh : flashSavingPlugin) kemudian men-decompilernya.

Akan tetapi tujuan dari mendecompiler bukanlah untuk proses reskinning (mencuri/memplagiasi karya orang lain)-tetapi tujuan yang sebenarnya adalah

mempelajari pola kerja orang lain dalam membuat sebuah game. Kita dapat membandingkan penggunaan action yang efektif dan yang kurang efektif.

buku ini hanya sebuah awal bagi saya untuk membagi sedikit pengalaman dalam membuat sebuah game flash. Untuk pengembangan yang selanjutnya sangat tergantung kepada ketekunan anda dan jam terbang anda dalam menghadapi sebuah logika pemrogramman game. Selamat Mencoba!

Hubungi saya pada email [wandah@wandah.com](mailto:wandah@wandah.com) apabila terdapat pertanyaan yang berkaitan dengan flash programming.