PENGEMBANGAN GAME HELI SHOTTER MENGGUNAKAN ACTION SCRIPT BERBASIS MACROMEDIA FLASH

SKRIPSI

Disusun Oleh :

TARMIN

NIM. 0612516

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2011

LEMBAR PERSETUJUAN

PENGEMBANGAN GAME HELI SHOTTER MENGGUNAKAN ACTION SCRIPT BERBASIS MACROMEDIA FLASH

SKRIPSI

Disusun dan diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai gelar Sarjana Teknik Teknik Komputer Dan Informatika Strata Satu (S-1)

Disusun Oleh :

TARMIN

NIM. 0612516

Mengetahui,

Ketua Program Studi Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT.
NIP.Y.101.8800.189

Diperiksa dan Disetujui,

Pembimbing I	Pembimbing II
M.Ibrahim Ashari, ST, MT.	**Ahmad Faisol, ST.**
NIP.P. 1030100358	**NIP.P.1031000431**

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

KATA PENGANTAR

Dengan mengucap syukur kehadirat Tuhan Yang Maha Esa yang dengan segala Kasih dan Anugerah–Nya, telah memberikan kekuatan, kesabaran, bimbingan dan perlindungan sehingga penulis dapat menyelesaikan laporan skripsi dengan judul:

" PENGEMBANGAN GAME HELI SHOTTER MENGGUNAKAN ACTION SCRIPT BERBASIS MACROMEDIA FLASH "

Pembuatan skripsi ini disusun guna memenuhi syarat akhir kelulusan pendidikan jenjang Strata I di Institut Teknologi Nasional Malang. Dalam penyusunan skripsi ini penulis banyak mendapat bantuan baik moril maupun materiil, saran dan dorongan semangat dari berbagai pihak, untuk itu penulis mengucapkan terima kasih kepada :

1. Bapak Ir. Soeparno Djiwo, MT selaku rektor ITN Malang
2. Bapak Ir. Sidik Noertjahjono, MT selaku Dekan Fakultas Teknologi Industri.
3. Bapak Ir. Yusuf Ismail Nakhoda, MT selaku Ketua Jurusan Teknik Elektro S-1 ITN Malang.
4. Bapak Ibrahim Ashari, ST, MT selaku Dosen Pembimbing I.
5. Bapak Ahmad Faisol, ST selaku Dosen Pembimbing II.
6. Bapak dan Ibuku tercinta yang selalu memberikan semangat untuk selalu meraih cita-cita dengan ilmu dan amal sholeh.
7. Teman – teman yang memberikan support dan nasihat untuk segera menyelesaikan skripsi.
8. Dan semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini yang tidak bisa penulis sebutkan satu persatu.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat diharapkan. Akhir kata, penulis mohon maaf kepada semua pihak bilamana selama penyusunan skripsi ini penyusun membuat kesalahan secara tidak sengaja dan semoga skripsi ini dapat bermanfaat bagi kita semua.

Malang, oktober 2011

Penulis

Abstrak

PENGEMBANGAN GAME HELI SHOTTER MENGGUNAKAN ACTION SCRIPT BERBASIS MACROMEDIA FLASH

TARMIN, NIM 0612516

Dosen Pembimbing : M.Ibrahim Ashari, ST, MT dan Ahmad Faisol, ST,

Bermain *game* bukanlah sesuatu yang asing, saat ini menjadi alternatif hiburan bagi tua, muda, pria maupun wanita. Hingga kini telah banyak mesin-mesin pemutar *game* atau consule yang mampu menampilkan *game* yang memukau, Perkembangan *game* memang begitu pesat seiring dengan perkembangan teknologi,dan inovasi *game,* salah satu permainan *game* komputer yang cukup menarik adalah tembak-menembak, salah satunya adalah game Heli shotter begitu nama istilahnya. Game Heli Shotter yang pembuatannya mengunakan Action script pada Macromedia flash.

Action Script adalah bahasa pemrograman yang dipakai oleh softwere flash untuk mengendalikan object-object ataupun movie yang terdapat dalam flash. Sebenarnya flash juga bisa tidak mengunakan Action script dalam pemakaianya, tapi kalau menginginkan adanya interaktifitas yang lebih kompleks maka Action script ini dibutuhkan. *Macromedia Flash 8,* merupakan *software* yang dirancang untuk membuat animasi berbasis vektor dengan hasil yang mempunyai ukuran yang kecil. Awalnya *software* ini memang diarahkan untuk membuat animasi atau aplikasi berbasis internet (*online*). Tetapi pada perkembangannya banyak digunakan untuk membuat animasi atau aplikasi yang bukan berbasis internet (*offline*). Dengan *Actionscript 2.0* yang dibawanya, Flash 8 dapat digunakan untuk mengembangkan *game* atau bahan ajar seperti kuis atau simulasi.

Kata kunci: *perkembangan Game, Action script, Macromedia Flash 8*

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bermain *game* bukanlah sesuatu yang asing, saat ini menjadi alternatif hiburan bagi tua, muda, pria maupun wanita. Hingga kini telah banyak mesin-mesin pemutar *game* atau consule yang mampu menampilkan *game* yang memukau, Perkembangan *game* memang begitu pesat seiring dengan perkembangan teknologi,dan inovasi *game*, salah satu permainan *game* komputer yang cukup menarik adalah tembak-menembak, salah satunya adalah game Heli shotter begitu nama istilahnya. Game ini bercerita tentang pertempuran sebuah Helikopter. Dimana setiap pemain berusaha menembak kapal-kapal dan menghancurkan yang menjadi musuhnya dengan cara mengclik *keyboard* arah panah untuk,dan spasi utuk menembak,shift untuk menyimpan. Berdasarkan uraian di atas, penulis bermaksud untuk mengembangkan suatu Game Heli Shotter yang pembuatannya mengunakan Action script pada Macromedia flash.

Action Script ^{2} adalah bahasa pemrograman yang dipakai oleh softwere flash untuk mengendalikan object-object ataupun movie yang terdapat dalam flash. Sebenarnya flash juga bisa tidak mengunakan Action script dalam pemakaianya, tapi kalau menginginkan adanya interaktifitas yang lebih kompleks maka Action script ini dibutuhkan. Oleh karena itu, penulis mengambil topik tugas akhir dengan judul

"Pengembangan Game Heli Shotter Menggunakan Action script Berbasis Macromedia Flash".

1.2 Perumusan Masalah

Berdasarkan latar belakang pemilihan judul, maka yang menjadi permasalahan adalah bagaimana membuat dan mengembangan Game Heli Shotter Menggunakan Action script berbasis Macromedia Flash.

1.3 Tujuan dan Manfaat Penulisan

Tujuan penyusunan tugas skripsi ini adalah untuk mengembangan Game Heli Shotter Menggunakan Action script Berbasis Macromedia Flash. Dan Manfaat dari penyusunan tugas skripsi ini yaitu:

1. Game dapat dijadikan sebagai sarana hiburan yang cukup menarik.
2. Sebagai pembelajaran pembuatan game mengunakan Action script

1.4 Pembatasan Masalah

Keterbatasan ruang lingkup permasalahan dalam mengembangan Game Heli Shotter menggunakan Macromedia Flash ini antara lain :

1. *Input* dibatasi hanya pada *keyboard* sedangkan *Output* perangkat lunak pada layar monitor.
2. Jumlah pemain adalah sebanyak 1 orang.
3. Pemain bisa memilih 1 Helikopter
4. Tingkat kesulitan Normal

1.5 Metodologi Penyelesaian Masalah

Langkah – langkah pengembangan game heli shotter ini antara lain

a. Menganalisa bagaimana sebaiknyan game *Heli shotter* berjalankan.

b. Mempelajari cara permainan dari game *Heli shotter*.

c. Merancang *interface* untuk game *Heli shotter*.

d. Merancang dan mengembangkan game *Heli shotter* dengan mengunakan *Macromedia Flash 8*.

1.6. Sistematika Pembahasan

Penyusunan laporan Skripsi ini menggunakan kerangka pembahasan yang terbentuk dalam susunan bab, yang dapat dijelaskan sebagai berikut :

Bab I : Pendahuluan

Bab ini merupakan dasar penyusunan laporan Skripsi yang di dalamnya berisi tentang Latar Belakang Masalah, Rumusan Masalah, Tujuan Skripsi, Batasan Masalah, Metodologi Pengembangan Sistem, dan Sistematika Pembahasan Skripsi.

Bab II : Dasar Teori

Bab ini berisi tentang teori-teori yang dipakai dalam penyusunan Skripsi ini.

Bab III: Analisis dan Perancangan

Bab ini berisi tentang tahap analisis yaitu identifikasi dan analisis masalah dan analisis kebutuhan sistem untuk menyelesaikan masalah yang dihadapi berdasarkan teori pada Bab II dan Bab III. Bab ini juga berisi hasil pengembangan yaitu proses kelanjutan dari tahap analisis meliputi proses akusisi pengetahuan.

Bab IV : Implementasi dan Pengujian

Bab ini berisi tentang implementasi hasil pengembangan pada Bab IV dan penyesuaian kebutuhan sistem serta meliputi hasil pengujian sistem.

Bab V : Penutup

Bab ini berisi tentang kesimpulan dan saran dari hasil penyusunan laporan Skripsi yang telah disusun.

BAB II

LANDASAN TEORI

2.1. Sejarah Perkembangan *Game*[8]

Sejarah teknologi *game* komputer secara langsung berhubungan dengan perkembangan komputer itu sendiri. Komputer dengan kecepatan processor tinggi, grafis yang lebih mendekati realita, dan media penyimpanan yang lebih besar sebenarnya dimaksudkan untuk memenuhi kebutuhan dalam bermain *games.*

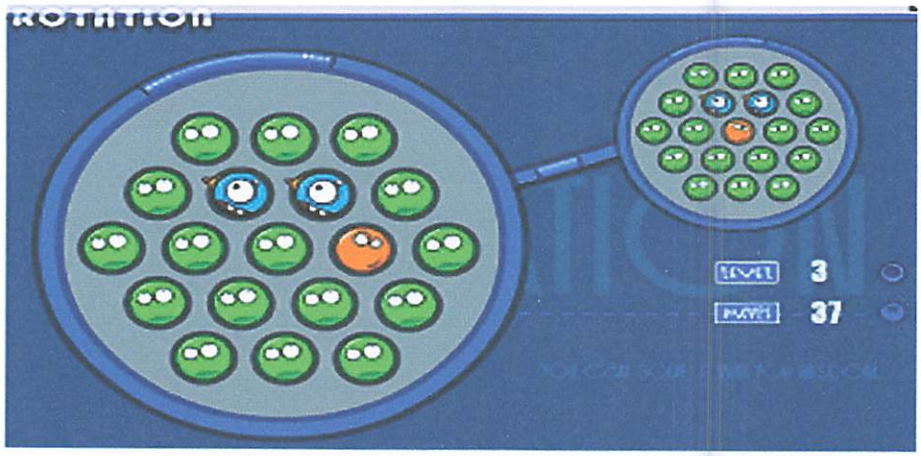
Game adalah sebuah permainan interactive yang membutuhkan komputer untuk bermain. Program komputer menerima input dari si pemain melalui pengendali dan menampilkan lingkungan buatan melalui TV atau layar monitor.

2.2. Jenis *Game*[3]

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 *game play* (cara memainkannya), game dapat diklasifikasikan menjadi beberapa jenis. Dalam buku ini klasifikasi game flash adalah sebagai berikut :

2.2.1 Board *Game*[3]

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



Gambar 2-1 Game "rotation" bertipe board game

2.2.2 Arcade [3]

Arcade games adalah semua permainan yang mudah dimengerti, menyenangkan dan grafiknya bagus walau biasanya sederhana. Pengertian mudah dimengerti dan menyenangkan dikarenakan permainan ini hanyalah berkisar pada hal-hal yang disenangi umum seperti pukul memukul, tembak menembak, tusuk menusuk, kejar mengejar dan semua yang mudah dan menyenangkan. Yang termasuk kedalam permainan jenis ini adalah Prince of Persia, Street Fighter, Golden Axe, Grand Prix, Robocop.

Gambar 2-2 Game "ballistic biscuit" game arcade

2.2.3. Action [3]

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, mocil, petualangan paddle pop, alloy tease exile dan sebagainya.

Gambar 2-3 Game "mocil"game bertipe action

2.2.4. Shooting [3]

Game bertipe *shooting* sebagian besar menggunakan *mouse* sebagai alat pengendalinya. 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.

Gambar 2-4 Game "terorist" game bertipe shooting

2.3 Karakter dalam Game [3]

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.

2.3.1. Gaya Realis Detail [3]

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.

Gambar 2-11 game bergaya Realis Detail

2.3.2. Gaya Realis Sederhana [3]

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.*

Gambar 2-12 karakter bergaya Realis Sederhana

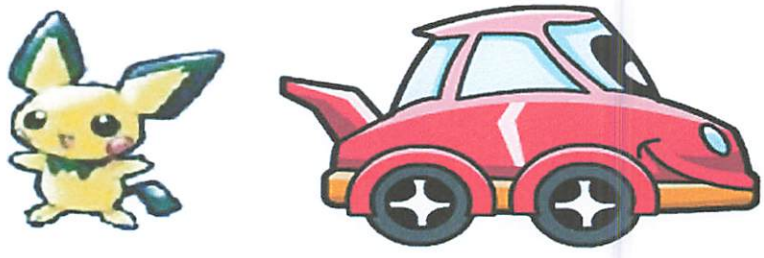
2.3.3. Gaya Kartun [3]

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.

Gambar 2-13 karakter bergaya kartun

2.3.4. Gaya Imajinasi [3]

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.*



Gambar 2-14 karakter bergaya imajinasi

2.4 Background Game [3]

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*.

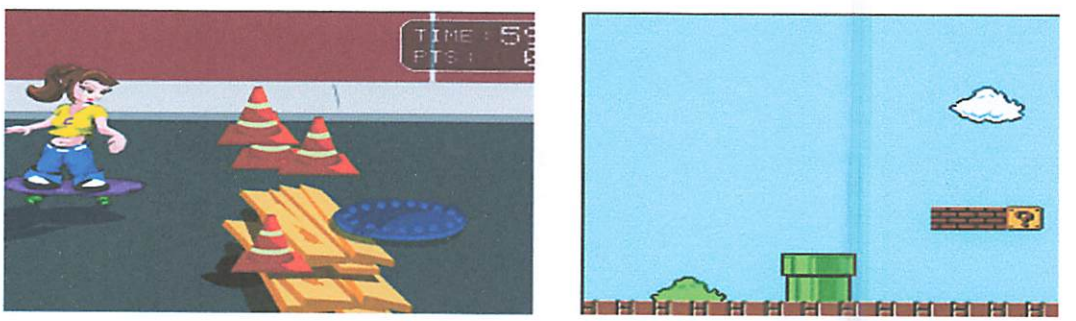
2.4.1. Static Background. [3]

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 :

Gambar 2-15 penggunaan static background

2.4.2. Scrolling Background.[3]

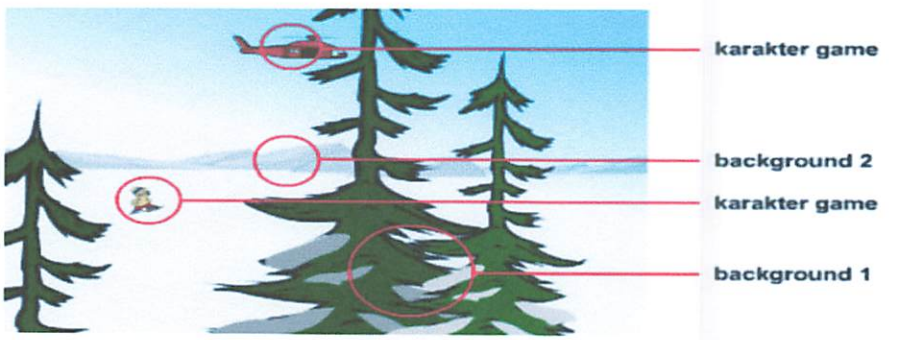
Berbeda dengan *static background* yang diam, *backgrond* bertipe *scrolling background* bergerak dari kanan kekiri, atas kebawah, atau bergerak bebas. Tingkat kesulitan dari pembuatan *scrolling background* adalah pada pengelolahan *script*, karena tanpa *script* yang baik, movie dapat berjalan lambat. Perhatikan contoh dari game yang menggunakan *scrolling background* berikut :



Gambar 2-16 scrolling background

2.4.3. Paralax Background [3]

Yang dimaksud dengan *paralax 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 *paralax background* dapat menambah kesan realis suatu *game*. Untuk membuat paralax background pada *game*, kita harus membuat minimal 2 lapis *background* dengan kecepatan yang berbeda. Perhatikan gambar berikut:



Gambar 2-17 Jenis paralax background

2.5 Program Aplikasi Macromedia Flash 8 [3]

Macromedia Flash 8, merupakan *software* yang dirancang untuk membuat animasi berbasis vektor dengan hasil yang mempunyai ukuran yang kecil. Awalnya *software* ini memang diarahkan untuk membuat animasi atau aplikasi berbasis internet (*online*). Tetapi pada perkembangannya banyak digunakan untuk membuat animasi atau aplikasi yang bukan berbasis internet (*offline*). Dengan *Actionscript 2.0* yang dibawanya, Flash 8 dapat digunakan untuk mengembangkan *game* atau bahan ajar seperti kuis atau simulasi.

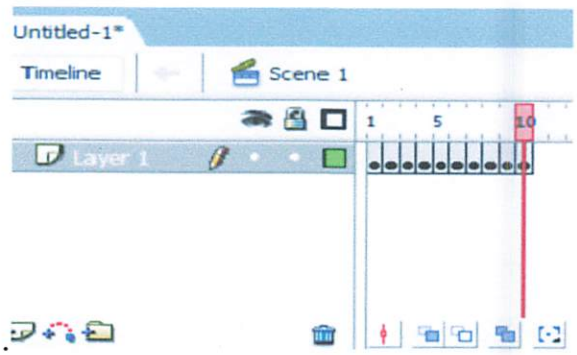
2.6 ANIMASI FLASH [4]

Animasi adalah gambar begerak berbentuk dari sekumpulan objek (gambar) yang disusun secara beraturan mengikuti alur pergerakan yang telah ditentukan pada setiap pertambahan

hitungan waktu yang terjadi. Gambar atau objek yang dimaksud dalam definisi di atas bisa berupa gambar manusia, hewan, maupun tulisan. Pada proses pembuatannyam sang pembuat animasi atau yang lebih dikenal dengan animator harus menggunakan logika berfikir untuk menentukan alur gerak suatu objek dari keadaan awal hingga keadaan akhir objek tersebut. Perencanaan yang matang dalam perumusan alur gerak berdasarkan logika yang tepat akan menghasilkan animasi yang menarik untuk disaksikan.

2.6 1.Frame by Frame Animation [5]

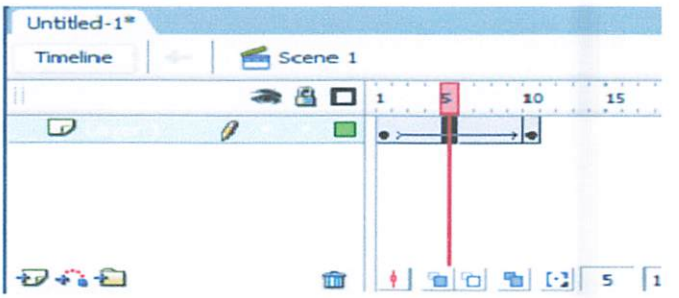
Pembuatan animasi dengan cara melakukan perubahan objek pada setiap *frame* secara manual, sehingga dihasilkan perubahan gambar yang teratur. Metode ini biasanya digunakan pada animasi dengan perubahan bentuk objek secara terus menerus. Misalnya, film kartun



Gambar 2-18 Tampilan layer animasi frame by frame

2.6 2. Motion tween [9]

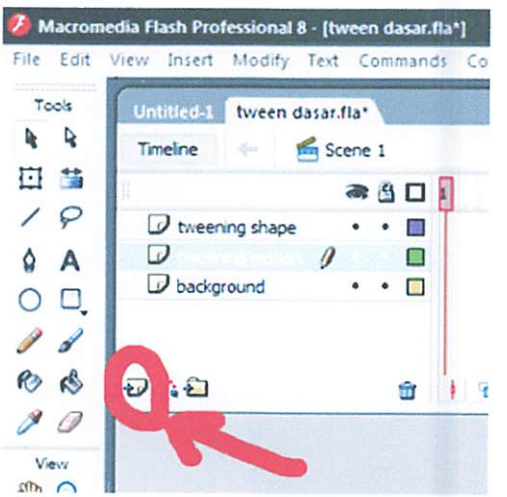
Pembuatan animasi dengan cara menentukan dua poin keadaan pada objek awal dan akhir, sedangkan macromedia flash membuat rangkaian gerakan diantaranya. Animasi yang dihasilkan menggunakan metode ini adalah gerakan yang halus, perubahan letak, ukuran, rotasi, bentuk maupun warna.



Gambar 2-19 Tampilan panel motion tween pada layer

2.6.3. Shape Tween [9]

Animasi *motion shape Tween*: adalah animasi yang digunakan untuk membuat animasi perputaran,baik berputar di tempat maupun berputar sambil berjalan.



Gambar 2-20 Tampilan Shape Tween pada layer

2.6.4 Motion Guide

gerakan yang teratur secara vertikal atau horisontal dan gerakan acak dapat membuat gerakan yang tidak beraturan dalam hal arah, namun teratur dalam pola gerakan

penggunaan **orient to path**

Gambar 2-21 Tampilan Motion Guide pada layer

2.7 Symbol, Instance, dan Library [3]

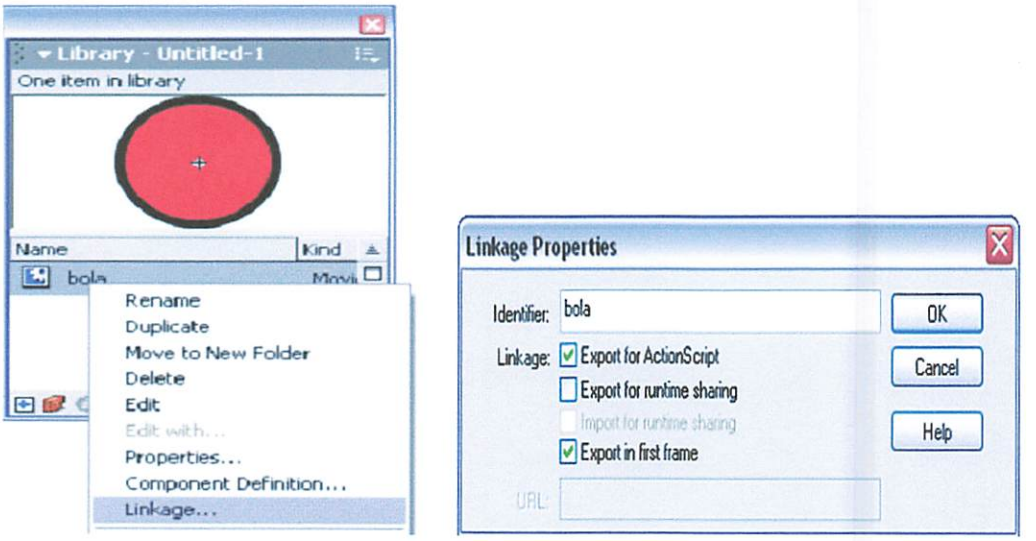
-*Stage Symbol* dalah suatu ruangan khusus yang digunakan untuk mengedit obyek dari *symbol* tersebut. *Stage symbol* ini tidak mempunyai hubungan dengan *stage* utama.

-Instances adalah duplikat dari symbol yang ada dalam *stage.*

-Library didalam flash fungsinya sesuai dengan namanya adalah sebuah tempat peyimpan symbol yang sudah kita buat.

2.8 Linkage [3]

Selain *instance* name dalam flash juga terdapat istilah *linkage* sebagai identitas pengenal suatu *symbol* (*movieclip, button dan sound*). Dengan menggunakan linkage pada *movie clip* maupun sound maka tidak perlu meletakkan *movie clip* ataupun sound yang telah memiliki *lingkage* kedalam area stage karena cukup dengan memanggil sesuai dengan nama l*ingkage*-nya,dengan *lingkage* akan membuat stage tidak terlalu penuh berisi movie clip dan frame tidak akan penuh dengan sound.



Gambar 2-22 Tampilan membuat linkage

2.9 Metode Penulisan Action Script [2]

Untuk membuat animasi yang dapat berintraksi dengan pengguna diperlukan sejumlah perintah, baik itu untuk mengontrol objek, membuat navigasi maupun interaksi lainnya, metode penulisan action Script dibagi dua cara menurut peletakannya, yaitu.:

- *Action Frame*, adalah Action Script yang dituliskan pada freme yang telah diberi action akan diberi tanda a.
- *Action Object*, adalah Action Script yang dituliskan pada objek baik berupa tombol maupun *movie clip*

2.10 Sound [5]

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 jens sound tersebut dapat dibuat dengan 3 cara, yaitu : dengan cara manual, dengan action script attachSound, dan dengan mengimpor Sound.

2.10.1 Mengatur Sound dengan Cara Manual [5]

cara *manual* 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.

2.10.2 Menambah Sound dengan attachSound() [5]

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.*

```
Variabel = new Sound();
Variabel.attachSound("linkage _ Soundname");
```

2.10.3 Mengimpor Sound dengan loadSound() [5]

penggunaan *soun*d 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 menyuport *file* bertipe MP3 dan harus berada pada satu tempat atau *folder*, Pengunaan loadSound() dalam action script adalah

```
Variabel = new Sound();
Variabel.loadSound("Soundname.mp3",0);
```

2.11 Programming [5]

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 :

2.11.1 Mengeset Variabel [5]

Bahasa pemrograman umumnya memiliki variable, dalam *game* variable tersebut misalnya *score*, energy, senjata, dan sebagainya. Untuk mengeset variable dalam AS menggunakan perintah berikut :

```
onClipEvent(load) {
        variable = nilai;
}
```

2.11.2 onClipEvent [3]

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

2.11.3 setProperty [3]

Action setproperty digunakan untuk mengubah property *movie clip*,seperti ukuran tranparansi,posisi maupun rotasi, bentuk umum dari setproperty dari action script adalah

Setproperty ("movie_clip",properties,parameter)

Properties berisi.:

- _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 merupakan presentase dari skala pembesaran vertikal suatu symbol terhadap titik regristasi.
- _visible adalah property pada symbol yang menentukan apakah obyek tersebut ditampakkan atau tidak pada stage. _visible bernilai 1 agar obyek tampak dan bernilai 0 agar obyek tidak tampak.
- _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).
- _rotation merupakan property yang menunjukan posisi perputaran obyek tersebut searah jarum pada titik registrationnya.
- _currentframe menunjukan frame yang aktif pada sebuah *movie* atau sebuah *movie klip*.

2.11.4 hitTest [3]

action hitTest adalah perhitungan area tumbukan yang dimulai dari titik-titik terluar, sehingga ketika obyek belum bertumbukan, ada peluang kondisi hitTest terpenuhi.

2.11.5 duplicateMovieClip dan removeMovieClip [6]

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);

action removeMovieclip juga memiliki 2 bentuk yaitu :

removeMovieClip(target); dan

nama instansi.removeMovieClip();

2.11.6 fscommand [6]

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 *movie* secara 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.

action fscommand juaga dapat dimanfaatkan sebagai jembatan penghubung antara program flash dengan program lain seperti java, mdm zinc dan sebagainya.

2.11.7 _ root.[6]

Perintah _root sering digunakan dalam pembuatan game, Perintah ini memiliki 3 bentuk dasar _root yaitu :

1. _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.
2. _root.variable / property,Pada bentuk _root yang kedua, dibelakang _root. diisi dengan variable, seperti variable *score*, waktu dan sebagainya.
3. _root.movieclip Bentuk _root yang ketiga tersebut bedakan lagi menjadi 2 yaitu:

   1_root.instance movieclip.variable/property

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

   2 _root. instance *movie clip*.action Bentuk ini digunakan untuk mengakses suatu *movie clip* dari *movie clip* lain.

2.11.8 Komentar pada Action Script [2]

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

pembuatan aplikasi besar yang terdiri dari banyak baris kode. yaitu dengan memberikan tanda (//) dan tanda (/**/) sebelum menuliskan pesan.

//komenta

2.11.9 Algoritma Game Helishotter

Algoritma yang diterapkan di dalam proses "perang" pada setiap level sedikit berbeda. Hal yang membedakannya adalah dari jumlah musuh dan lamanya level tersebut dimainkan sampai bertemu *Boss*. Namun secara garis besar algoritma inti permainan tetap sama.

1. Inisialisasi variable – variable;
2. Looping pada event onEnterFrame()
   a. Jika pemain belum menang (win = false atau !win), maka
   i. Cek apakah pemain menekan tombol pause (Escape)
   ii. Jika tombol pause tidak ditekan (gamePaused = false), maka
      1. Panggil fungsi scroll_background() untuk menggerakan background permainan secara terus menerus.
      2.Cek apakah boss sudah muncul atau belum?
         a.Jika tidak, maka baca input keyboard untuk menggerakan pemain, musuh, bonus, dan peluru. Kembali ke looping utama (langkah no. 2)
         b. Jika ya, maka cek apakah boss telah siap untuk menembak dan ditembak ? (bossReady = true)
         i. Jika boss belum siap, maka panggil fungsi untuk mempersiapkan boss (createTheBoss) serta menggerakkan objek – objek musuh, pemain, peluru, dan bonus.
         ii. Jika boss telah siap (bossCreated)

1. Baca input keyboard pemain dan pelurunya.
2. Cek apakah sudah menang atau belum? (win = true)
   a. Jika menang, maka hapus seluruh objek yang tidak diperlukan seperti musuh, peluru (clears), serta mainkan efek boss hancur.
   Setelah boss hancur, maka akan muncul tombol untuk melanjutkan ke tahap berikutnya. Kembali ke algoritma awal (langkah 1) untuk level berikutnya.
   b. Jika belum menang, maka gerakkan boss, musuh. Kembali ke looping utama (langkah no. 2)

iii. Jika ditekan, maka gamePaused = true

1. Jika gamePaused = true, maka :
   a. Tampilkan background pause Game (berisi menu lanjut, keluar, dan simpan permainan)
   i. jika pemain menekan menu lanjut, maka permainan dilanjutkan. gamePaused = false;
   ii. Jika pemain menekan tombol keluar, maka muncul pertanyaan apakah yakin ingin keluar? Jika ya, maka akan keluar dari game, jika tidak, akan kembali ke kondisi gamePaused
   iii. Jika memilih simpan permainan, maka user diminta untuk meninput nama pemain, lalu bisa keluar atau melanjutkan permainan
2. Jika gamePaused = false, maka lanjutkan permainan. Kembali ke looping utama (langkah no.2)
3.

2.11.10 Menggerakan MOVIE CLIP dengan Keyboard

Buat sebuah movie clip sembarang. Klik pada movie clip lalu tekan F9. Tambahkan perintah berikut ini :

```
if (Key.isDown(Key.LEFT)) {
        this._x -= 5; // geser posisi movie clip sebanyak 5 piksel (ke kiri)
}
if (Key.isDown(Key.RIGHT)) {
        this._x += 5; // geser posisi movie clip sebanyak 5 piksel (ke kanan)
}
if (Key.isDown(Key.DOWN)) {
        this._y += 5; // geser posisi movie clip sebanyak 5 piksel (ke bawah)
}
if (Key.isDown(Key.UP)) {
        this._y -= 5; // geser posisi movie clip sebanyak 5 piksel (ke atas)
}
```

BAB III

PEMBAHASAN DAN PERANCANGAN

3.1. Pembahasan

Perancangan perangkat *game* helishooter melalui beberapa tahapan, yaitu

3.1.1 Sistem Permainan

Sistem permainan *game* Helishotter ini pemain mengendalikan sebuah helikopter tempur, dimana pemain akan berhadapan dengan helikopter dan pesawat musuh, ada beberapa variasi pesawat musuh yang masing-masing memiliki warna yang berbeda, pada waktu-waktu tertentu pemain dapat memperoleh bonus seperti *gun* yang akan menambah kecepatan senjata tempur pesawat pemain, speed bonus yang akan membuat pesawat pemain akan bergerak lebih cepat, dan health bonus yang akan menambah kesehatan pesawat pemain, permainan berakhir jika pesawat pemain habis atau score tidak mencukupi untuk naik level berikutnya saat waktu sudah habis.

3.1.2 Perancangan *MovieClip*, Animasi, dan *Sound Effect*

MovieClip, Animasi didapat dari berbagai sumber di *internet*. Ada beberapa bagian seperti *background* yang dibuat menggunakan perangkat lunak grafis Photoshop. *Sound Effect* didapat dari CD yang banyak dijual di toko musik dan internet.

1. Menentukan Dimensi Layar

Dimensi layar yg digunakan adalah ukuran default dari Macromedia Flash 8, yakni 550 x 400 piksel.

2. Menambahkan Instance Name

Menambahkan *Instance name* pada *movie clip* diperlukan agar *movie* lain bisa mengakses nilai variable yang terdapat pada *movie clip* tersebut, nama *movie clip* dan nama *instance* yang ditambahkan berikut ini:

Tabel 3-1 Instance Name Game Heli Shotter

No	Movie Clip	Instance Name
1	Helikopter pemain	heliPemain
2	Helikopter Musuh #1	heliMusuh1
3	Helikopter Musuh #2	heliMusuh2
4	Helikopter Musuh #3	heliMusuh3
5	Background Level #1	bgLevel1
6	Background Level #2	bgLevel2
7	Background Level #3	bgLevel3
8	Jenis tembakan pemain #1 (bullet)	tembakan1
9	Jenis tembakan pemain #2 (laser)	tembakan2
10	Jenis rudal #1	rudal1
11	Jenis rudal #2	rudal2
12	Ledakan Kecil	ledakanKecil
13	Ledakan Besar	ledakanBesar

3.1.3 Menambahkan Variable Name

Menambahkan variable *name* pada text bertipe *dinamic text* diperlukan agar *movie clip* bisa menentukan nilai yang akan ditampilkan pada *dinamic tekt* tersebut, *dinamic tekt* yang memiliki variable *name* sebagai berikut:

Tabel 3-2 Variable Name Game Heli Shotter

No	Variabel	Variabel Name
1	Skor	skor
2	Level	level
3	Jenis Bonus Rudal	jenisRudal
4	Jenis tembakan	jenisTembakan

3.1.4 Menambahkan Background Music (BGM)

Karena durasi *backsound* yang panjang akan menyebabkan ukuran *file* membengkak, maka suara *backsound* dalam *game* mengunakan metode loadsound, dimana *file* suara tidak berada dalam area kerja flash, akan tetapi berada diluar *file*, dengan demikian ukuran *file* menjadi tidak membengkak.

Tabel 3-3 Background Music (BGM)Game Heli Shotter

No	BGM	BGM Name
1	Halaman Utama (Menu)	soundUtama
2	Halaman saat perang (permainan)	soundPerang
3	Halaman Petunjuk	soundPetunjuk

4	Halaman Cerita	soundCerita
5	Helicopter Pemain	soundHeliPemain
6	Helikopter Musuh	soundHeliMusuh
7	Pesawat Meledak	soundMeledak
8	Tembakan pesawat	soundTembakan

3.1.5 Metode Penulisan Action script [5]

Di Flash actionscript ditulis pada panel actions. Penulisan actionscript di panel actions dapat dilakukan pada 3 tempat yaitu pada movie clip, button, dan frame. Ketiga buah tempat tadi memiliki aturan penulisan yang berbeda.

1. Untuk menjalankan event pada button, script pada button penulisannya diawali dengan on(){}

2. Untuk menjalankan event pada movie clip, script pada movie clip penulisannya diawali dengan on(){} atau onClipEvent(){}

3. Untuk menjalankan event pada frame, script pada frame penulisannya diawali dengan onLoad=function(){} atau onEnterFrame=function(){} atau onPress= function() {} atau onRelease=function(){} dsb.

3.1.6 Metode Penyimpanan[6]

Untuk menyimpan nilai *score,time,lavel*, dan *healt* serta menyimpan nilai yang diperlukan untuk *hight score* menggunakan *shared objek*, dimana *shared* akan digunakan untuk menyimpan *file cookies* kedalam *storage*, sehingga *shared objekct* dapat digukan untuk menyimpan nilai suatu variable, *file* hasil *shared object* akan berekstensi sol, dalam satu falsh *movie*.

Penyimpanan data menggunakan SharedObject, fitur yg dimiliki oleh flash. SharedObject disimpan ke dalam sebuah file berekstensi **.sol**, yg berada di direktori C:\Documents and Settings\[Username]\Application Data\Macromedia\Flash Player\#SharedObjects\[Random Folder]\localhost\[random folder tergantung lokasi file flash]

Buat sebuah file baru, pada frame pertama, ketik :

```
Stop();
Var so:SharedObject = Sharedbject.getLocal("dataUser");
if (so.data.nama == null) {
        trace ("tidak ada data!");
        // simpan data baru
    So.data.nama = "Tarmin";
}
```

3.1.7 Penentuan *Finish Game*

JIka nilai variable level telah dilewati, maka permaianan dinyatakan selesai dan akan menuju pada halaman selebrasi

```
// FINISH !

// set teks skor sesuai skor yang didapat
_root.txtFinalSkor.text = "Skor kamu : "+_root.skor;
stop();
this.onEnterFrame = function() {}
```

3.1.8 Perhitungan *Score*

Sebuah *game* 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. Dan jika sebuah pesawat telah diledakkan maka *score* permainan akan ditambahkan.

Fungsi randomBonus parameter musuh dengan tipe MovieClip melakukan random bonus HP dan kecepatan peluru.jika HP pemain < 50, maka fungsi akan mencoba untuk memunculkan bonus HP dengan syarat bonusHP masih tersisa.begitu pula dengan bonus kecepatan peluru, akan dimunculkan sesuai random frame yang sebelumnya telah dilakukan pada perulangan di atas.

```
*/
function randomBonus(musuh:MovieClip) {
        //trace (bonusHP);
        // kasi bonus HP jika HP <= 50
        if (playerHP <= 50) {
                if (bonusHP > 0) {
                        if (musuh._y < 300) {
                                //trace (" bonus HP");
                                var hp = this.attachMovie("bonusHP", "bonusHP_"+18000,
18000);
                                hp._x = musuh._x;
                                hp._y = musuh._y;
                                bonusHP--;
                                bonus.push(hp);
```

```
                    bonusType.push("HP");
                    bonusHPFrame.splice(i, 1);
                }
            }
        } else {
            if (bonusHP > 0) {
                // trace ("kasi bonus hp secara random");
                // kasi bonus HP secara random
                for (i  = 0; i < bonusHPFrame.length; i++) {

                    if (currentRepeat == bonusHPFrame[i]) {

                        if (musuh._y <  300) {
                            //trace (" bonus HP");
                            var hp = this.attachMovie("bonusHP",
"bonusHP_"+random(1000), 18000);
                            hp._x = musuh._x;
                            hp._y = musuh._y;
                            bonusHP--;
                            bonus.push(hp);
                            bonusType.push("HP");
                            bonusHPFrame.splice(i, 1);
                            break;
```

```
                    }
                }
            }
        }
        if (bonusBulletSpeed > 0) {
            // trace ("kasi bonus speed secara random");
            // kasi bonus HP secara random
            for (i  = 0; i < bonusBulletInFrame.length; i++) {
                if (currentRepeat == bonusBulletInFrame[i]) {

                    if (musuh._y <  300) {
                        //trace ("bonus Speed");
                        var speed = this.attachMovie("bonusBulletSpeed",
"bonusBulletSpeed_"+random(1000), 18000);
                        speed._x = musuh._x;
                        speed._y = musuh._y;
                        bonusBulletSpeed--;
                        bonus.push(speed);
                        bonusType.push("bulletSpeed");
                        bonusBulletInFrame.splice(i, 1);
                        break;
                    }
                }
```

```
                }
            }
        }
        //trace (bonus);
}

/*
        fungsi untuk menggerakkan bonus (jika ada)
*/
function movesBonus() {

        if (bonus.length > 0) {
                //trace ("bonus is more than 0");
                for (i = 0; i < bonus.length; i++) {

                        bonus[i]._y += 2;

                        if (bonus[i]._y > 600) {
                                removeMovieClip(bonus[i]);
                                bonus.splice(i, 1);
                                bonusType.splice(i, 1);
                        }
                        //trace ("test");
```

```
if (bonus[i].hitTestShape( _root.player ) ) {
        //trace ("hitting player");
        //trace (bonusType[i]);
        if (bonusType[i] == "HP") {
                //trace ("get bonus HP");
                //trace (playerHP);
                removeMovieClip(bonus[i]);
                bonus.splice(i, 1);
                bonusType.splice(i, 1);
                playerHP += bonusHPAmount;
                if (playerHP + bonusHPAmount >= maxHP)
                        playerHP = maxHP;

                //trace (playerHP);
        } else if (bonusType[i] == "bulletSpeed") {
                //trace ("get bonus Speed");
                removeMovieClip(bonus[i]);
                bonus.splice(i, 1);
                bonusType.splice(i, 1);
                bulletSpeed = bonusBulletSpeedAmount;
                bonusBulletSpeedTimer =
setInterval(stopBonusBulletSpeed, bonusBulletSpeedTime);
                //trace ("got bonus bullet");
```

```
                    }
                    gotBonus.start(0, 0);
                }
            }
        } else {
            //trace ( "bonus is 0 " );
        }

        //trace ( "test" );
}
```

3.2 Perancangan

Game Helishooter ini dirancang menggunakan perangkat lunak *Macromedia Flash 8* dengan bahasa pemrograman Action Script 2.0. Sebelum memulai pengerjaan, terlebih dahulu dibuat sebuah *story board game* agar lebih jelas dan terarah selama proses pembuatan. Berikut ini:

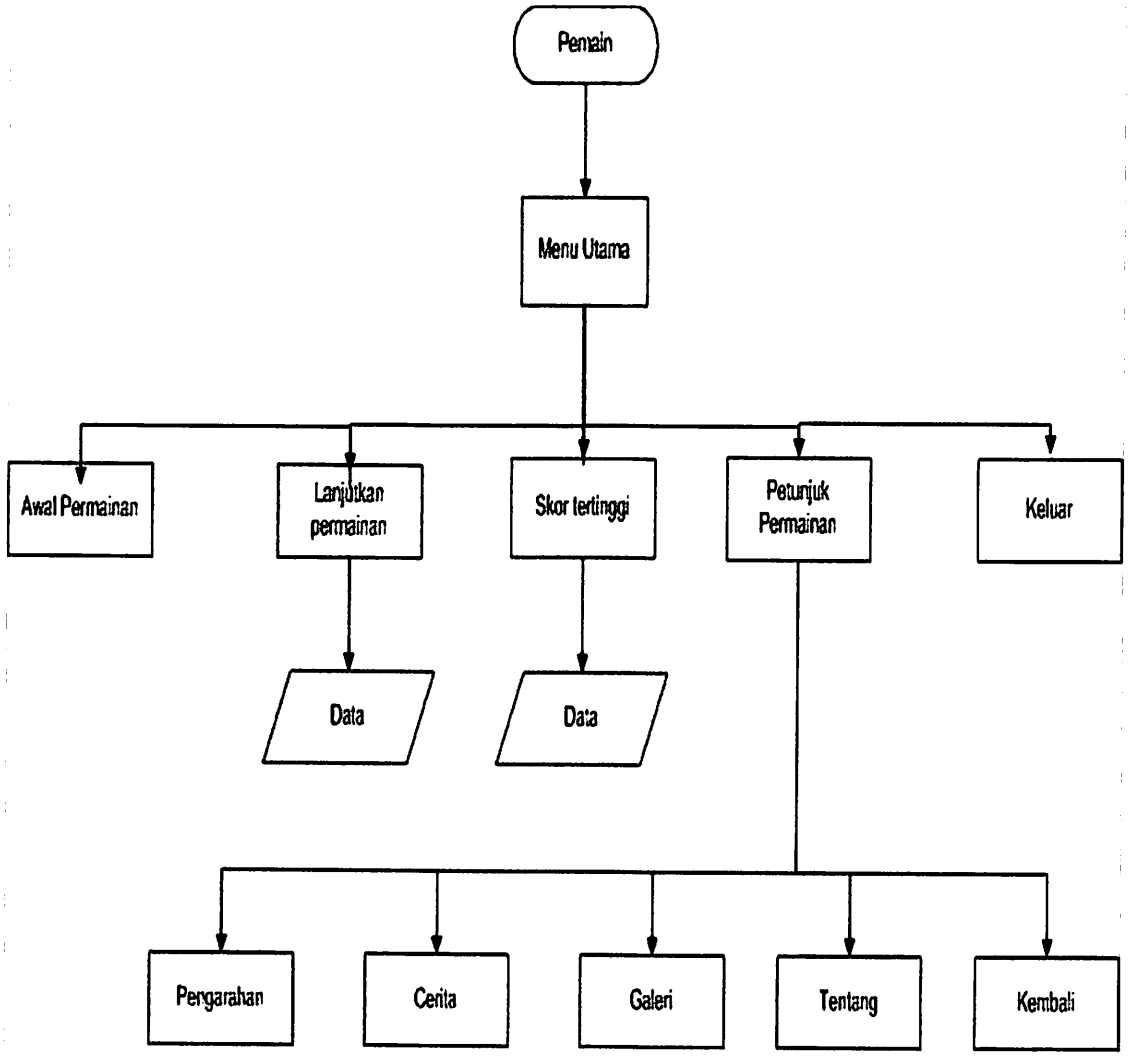
Tabel 3-4 Story Board Game Heli Shotter

Judul Game	Heli Shooter
Jenis Game	Arcade
Sistem Kendali	Keyboart untuk arah panah untuk menggerakkan karakter dan spasi untuk menembak, Shift untuk menyimpan permainan.
Karakter Utama	Helicopter coklat
Musuh	Helicopter biru, Helikopter hitam, dan pesawat jet coklat
Background	Level 1 padang pasir, Level 2 Hutan Vietnam, Level 3 gunung

Setting	bebatuan
Sistem Permainan	Pemain harus menembakan pesawat musuh, agar dapat menuju level berikutnya, pada level dan waktu tertentu pemaian dapat memperoleh gun untuk mempercepat laju rudal saat menembak, speed untuk menambah kecepatan gerak pesawat dan healt yang akan menambah kesehatan pesawat, permainan berakhir jika kesehatan pesawat habis atau skor tidak mencukupi untuk naik level selanjutnya saat waktu habis.

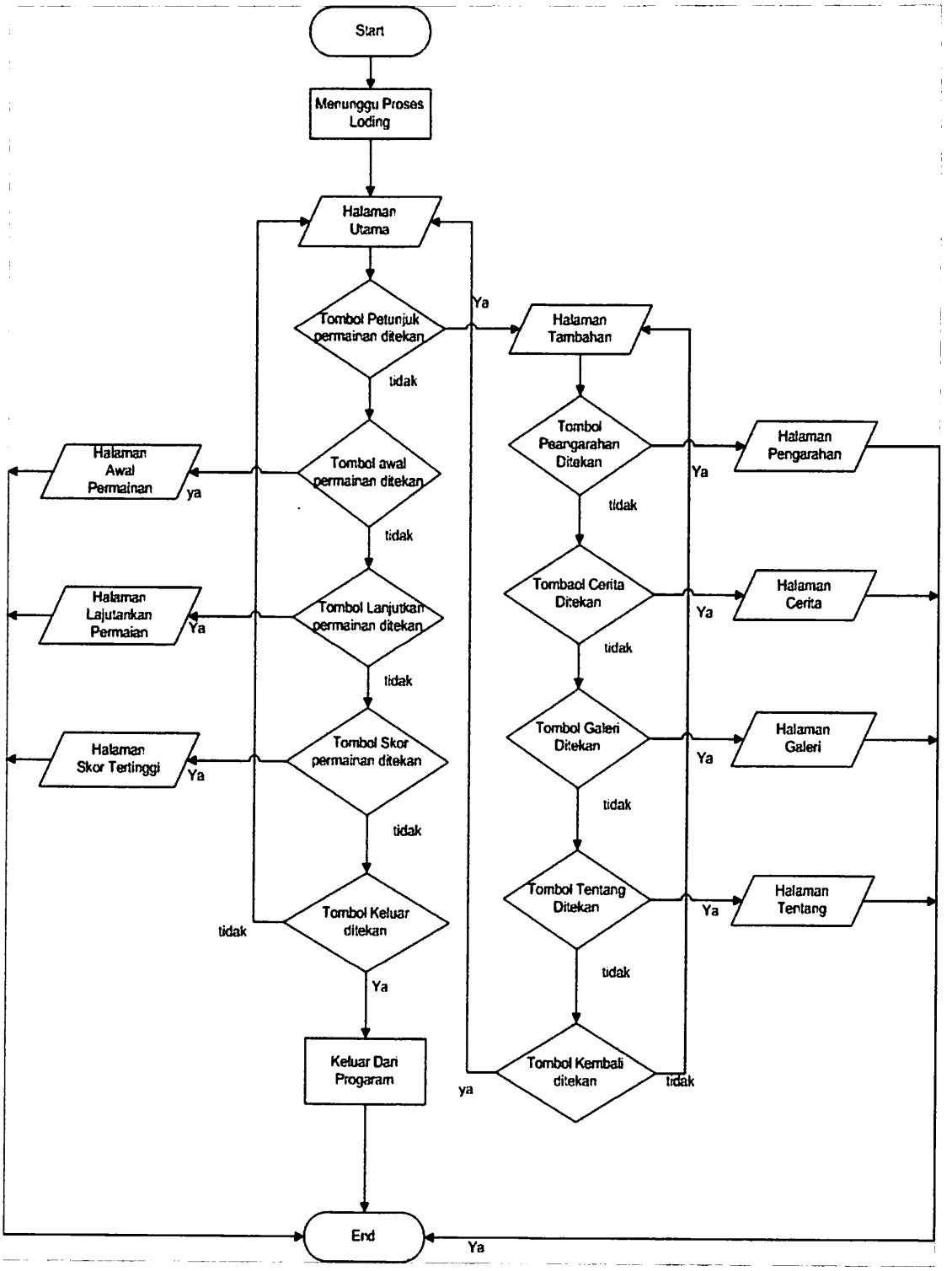
3.2.1 Desain Sistem Game Helishotter

Secara garis besar desain sistem *game* Helishooter memiliki gambaran desain seperti dibawah ini:



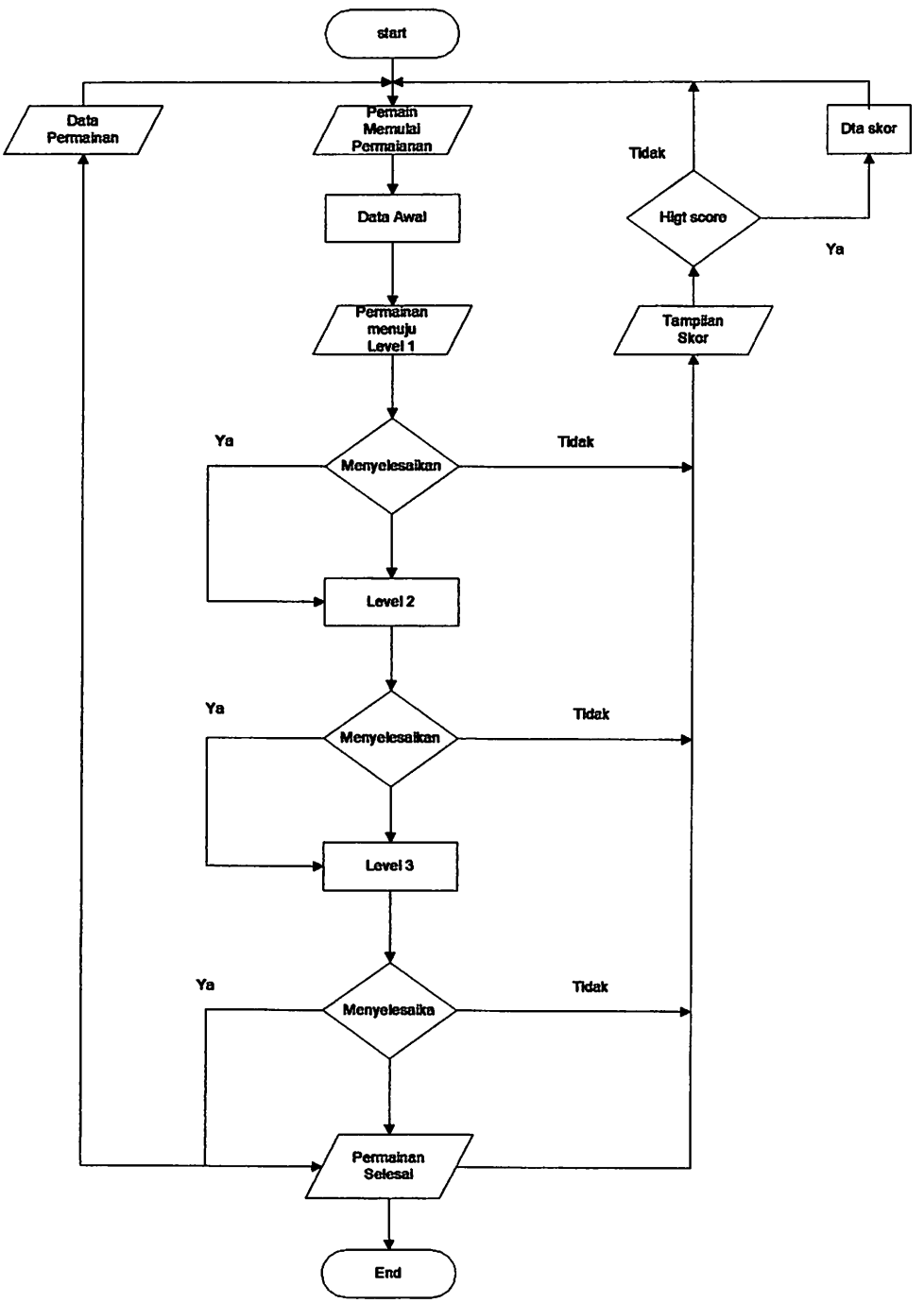
Gambar 3.1 Desain Sistem game Heli Shotter.

3.2.2 Flowchart game Helishotter



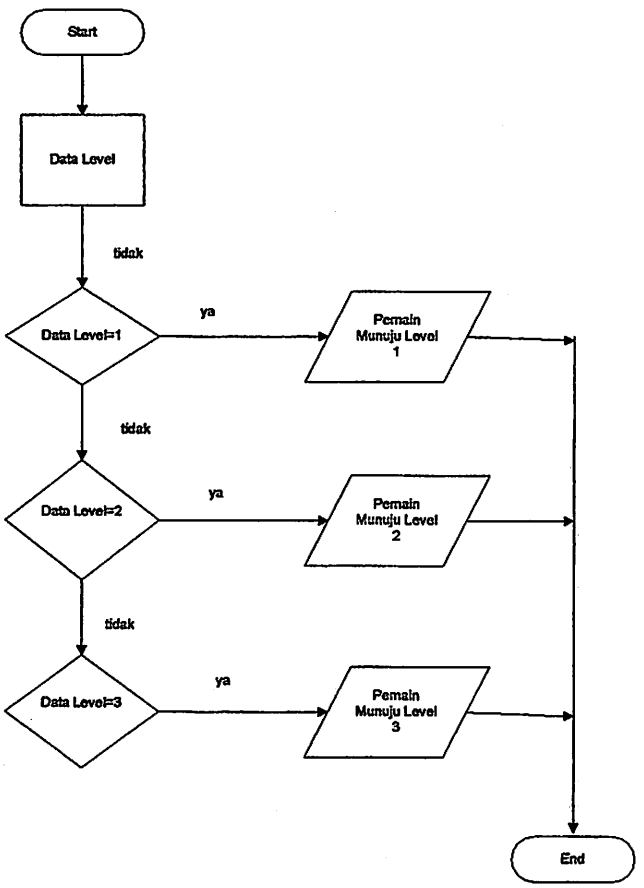
Gambar 3.2 Flowchart Game Heli shotter.

3.2.3 Flowchart alur permainan game Helishooter



Gambar 3.3 Flowchart Alur permaian Game Heli shotter.

3.2.4 Flowchart Alur melanjutkan game Helishooter :

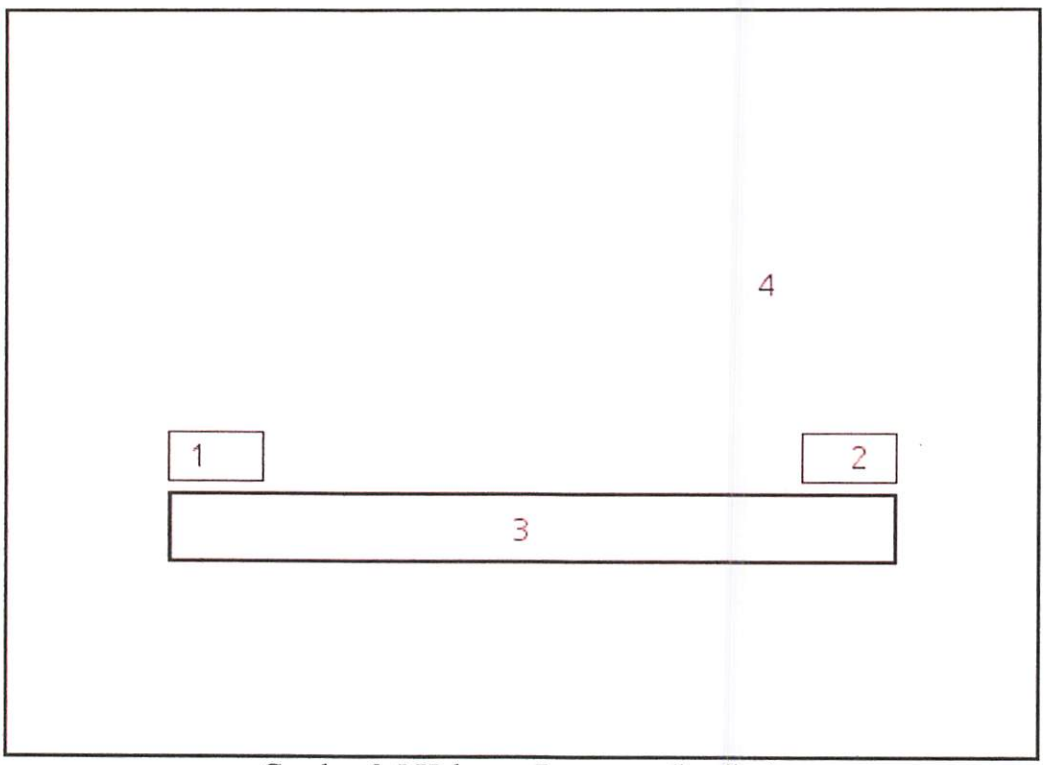


Gambar 3.4 Flowchart Melanjutkan Permainan

3.2.5 Desain Tampilan

Berikut ini tampilan game helishooter :

1. Tampilan Halaman Pemuatan (loading)

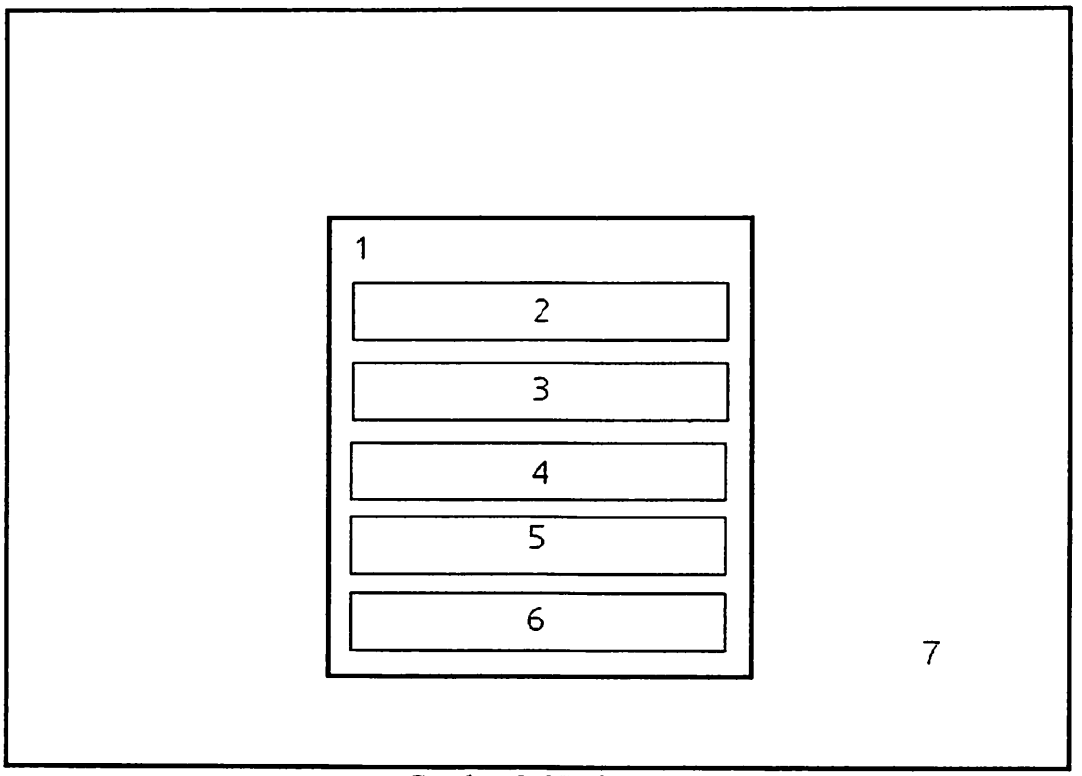


Gambar 3.5 Halaman Pemuatan (loading)

Keterangan:

1. Indek Loading
2. Indek Loading angka persen
3. Indek warna

3.2.6 Tampilan Halaman Utama



Gambar 3.6 Halaman Utama

Keterangan:

1 .Garis luar tombol

2..Tombol Permaian baru, untuk memulai permainan

3.Tombol lanjutkan permaian, untuk menuju halaman lanjutkan permaianan

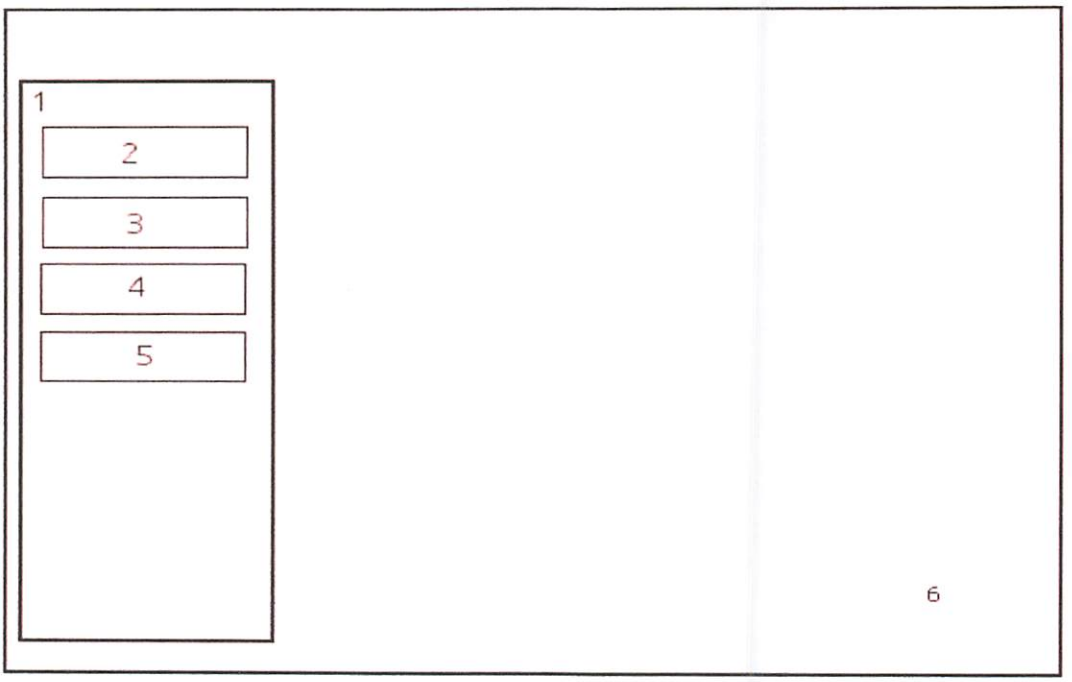
4.Tombol skor Tertinggi, untuk menuju halaman skor tertinggi

5.Petunjuk Permaian, untuk menuju halaman petunjuk permaian

6.Tombol keluar untuk keluar dari permainan game

7.Gambar Beckground halaman utama

3.2.7 Tampilan Halaman Petunjuk Permainan

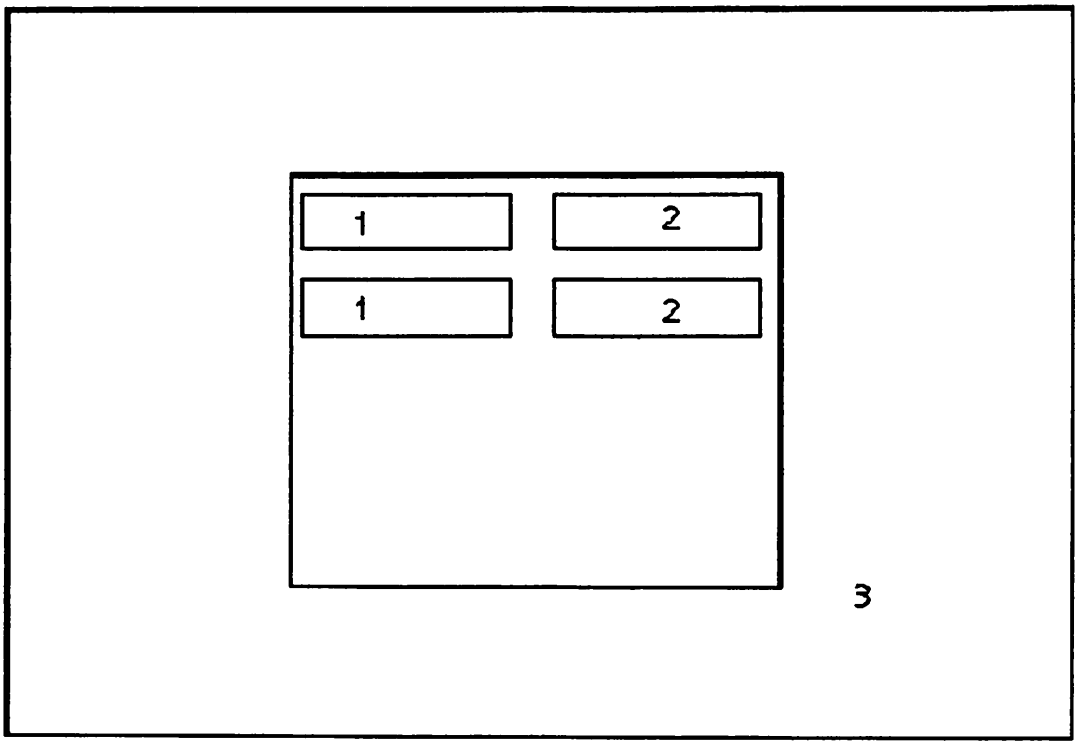


Gambar 3.7 Halaman Petunjuk Permainan

Keterangan:

1. Garis luar tombol
2. Tombol pengarahan untuk menuju halaman pengarahan
3. Tombol cerita untuk menuju halaman pengarahan
4. Tombol galeri untuk menuju halaman galeri
5. Tombol tentang untuk menuju halaman tentang
6. Gambar Beckground Petunjuk Permainan

3.2.8 Tampilan Halaman Skor Tertinggi

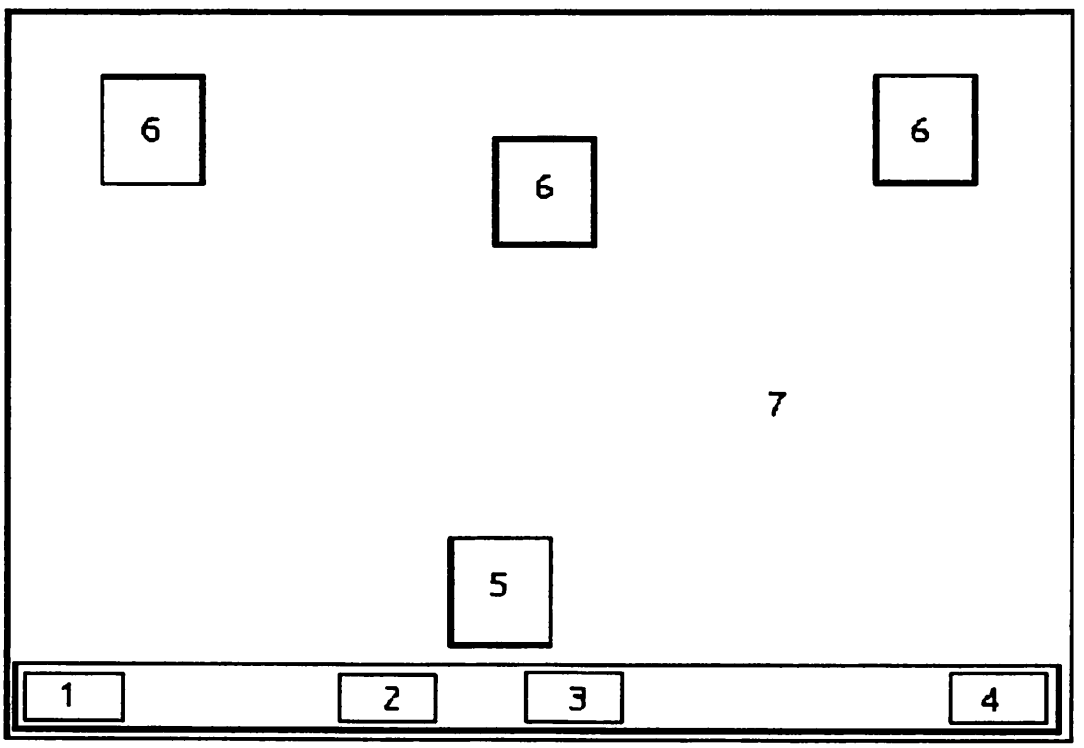


Gambar 3.8 Halaman Skor Tertinggi

Keterangan:

1. Menampilkan nama nilai skor
2. Menampilkan nilai.
3. Gambar Beckground Skor Tertinggi

3.2.9 Tampilan Halaman Perang

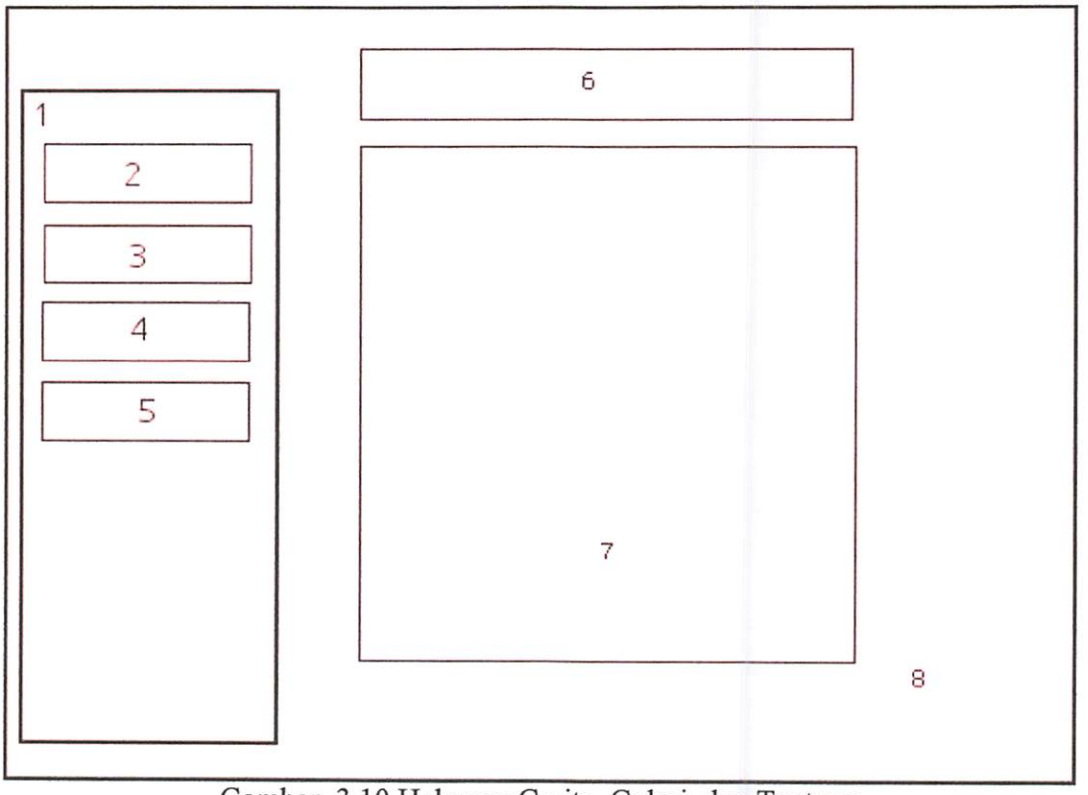


Gambar 3.9 Halaman Perang

Keterangan:

1. Helth power helicopter
2. Level.
3. Skor
4. Gun
5. Helicopter pemain
6. Pesawat musuh
7. Gambar Beckground Halaman perang

3.2.10 Tampilan Halaman Pengarahan, Cerita, Galeri, dan Tentang



Gambar 3.10 Halaman Cerita, Galeri, dan Tentang

Keterangan:

1. Garis luar tombol
2. Tombol pengarahan untuk menuju halaman pengarahan
3. Tombol cerita untuk menuju halaman pengarahan
4. Tombol galeri untuk menuju halaman galeri
5. Tombol tentang untuk menuju halaman tentang
6. Judul teks
7. Halaman teks
8. Gambar Beckground Petunjuk Pengarahan

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 IMPLEMENTASI

Implementasi dan system dalam perangkat lunak game Heli shooter ini mencakup spesifikasi kebutuhan perangkat keras (hardware) dan spesifikasi perangkat lunak (software)

4.2 Spesifikasi Prangkat keras dan Perangkat Lunak

Program ini dibuat dengan menggunakan perangkat keras (hardware) sebagai berikut:

1. Prosesor Intel Pentium IV 2.66 GHz
2. Memory 512 MB
3. Hardisddisk 80 GB, dengan Freespase 10,5
4. VGA card 128 MB
5. Monitor dengan resolusi 1024 x 768 pixel
6. Keyboard dan mouse

Adapun perangkat lunak (software) yang direkomendasikan untuk menjalankan aplikasi ini adalah lingkungan system operasi MS-Windows NT/2000/XP/WIN7.

4.3 Pengujian Hasil

Pengujian ini bertujuan untuk mengetahui sejauh mana keberhasilan aplikasi ini diterapkan dalam system. Pengujian ini juga untuk mencari letak error atau kesalahan program dalam aplikasi.

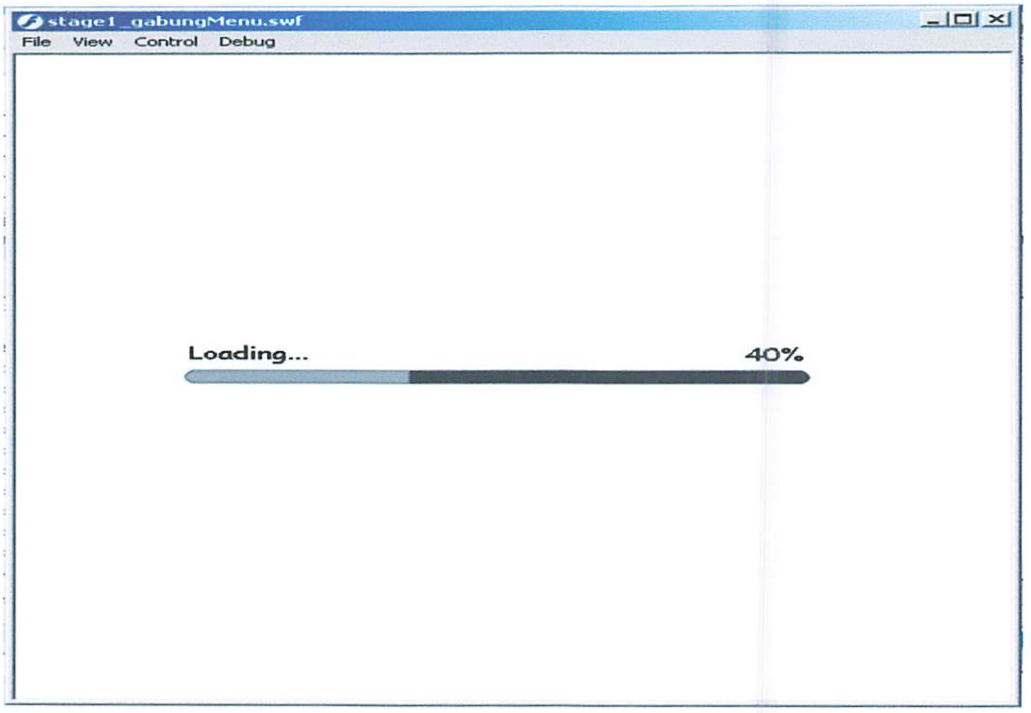
4.3.1 Perbandingan Game Sebelum Dikembangkan

Gambar 4.1 Halaman game sebelum dikembangkan

1. Game sebelum dikembangkan masih berbentuk 2 dimensi
2. Game belum ada tingkatan level.
3. Tidak ada variasi tembakan dari pemain
4. Game sebelumnya pembuatannya menggunakan software Gamemaker,

4.3.2 Pengujian Halaman Pemuatan

Preloader digunakan untuk mengetahui berapa lama waktu yang diperlukan untuk menampilkan flash movie, Preloader dapat disisipkan animasi agar user sabar menunggu flash movie selesai diload.



Gambar 4.2 Halaman proleader

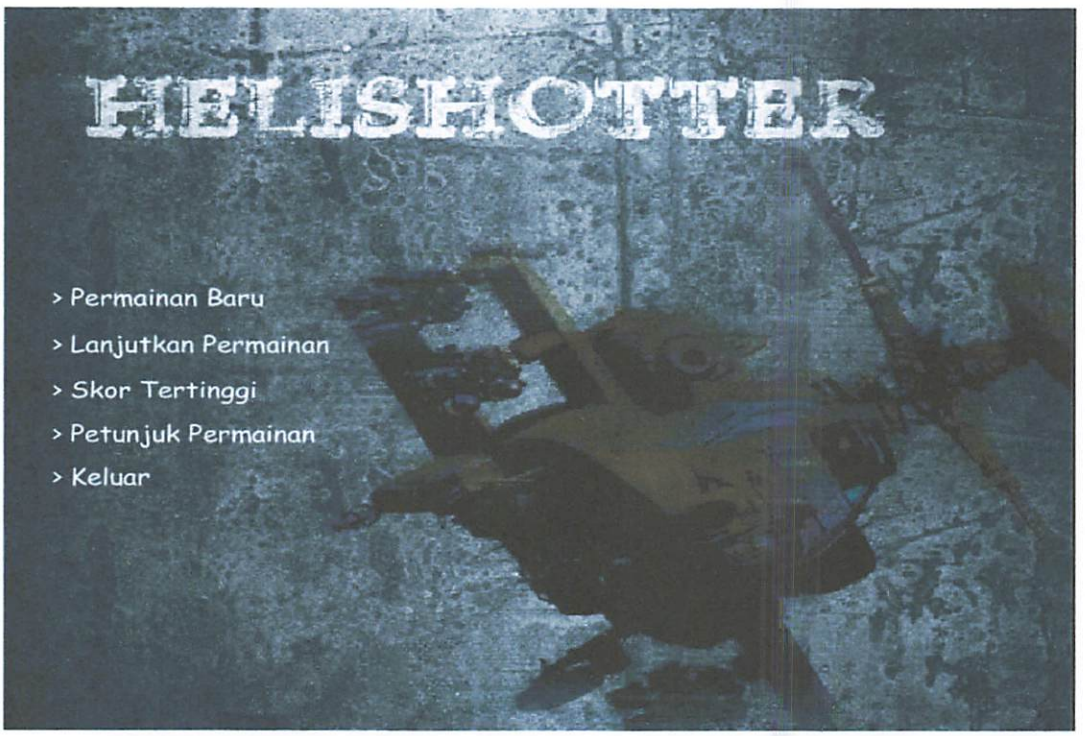
```
// stop() digunakan untuk menghentikan animasi frame
stop();

/*      skrip di bawah ini untuk melakukan loading
        loading adalah suatu proses untuk memuat/ me-load komponen-komponen
        berupa gambar, suara, movieClip, dsb yang digunakan oleh game
```

```
        variabel angka untuk menyimpan nomor frame pada movieClip loading_mc
        variabel persen_tampil adalah dynamic_text yang berisi keterangan
        sudah berapa persen proses loading berjalan

        variabel persen untuk menyimpan nilai persen yang sudah di-load oleh flash
        perhitungannya dengan pembulatan (round) jumlah byte yang telah didownload
        dibagi dengan total yang harus didownload dikali 100

        jika persen dimoduluskan oleh 10 hasilnya 0, maka persen adalah kelipatan 10,
        maka tambahkan variabel angka menjadi nomor frame berikutnya. lalu load animasi
        pada frame ke-angka pada movieClip loading_mc.
        update teks persen_tampil menjadi persen yang sudah di-load
*/
angka = 0;
persen_tampil = 0+"%";
_root.onEnterFrame = function() {
        if (Math.round(_root.getBytesTotal()) <= Math.round(_root.getBytesLoaded())) {
                gotoAndStop(2);
        } else {

                persen                                                          =
Math.round(Math.round(_root.getBytesLoaded())/Math.round(_root.getBytesTotal())*100);
                if (persen%10 == 0) {
```

```
                angka = (persen/10)+1;
                loading_mc.gotoAndStop(angka);
                persen_tampil = persen+"%";
            }
        }
};
```

4.3.3 Pengujian Halaman Utama

Setelah Preloader selesai program akan menuju halaman utama, pada halaman ini dari lima tombol yaitu awal permainan untuk memulai permainan awal, tombol lanjutkan permaian untuk melanjutkan permainan yang telah tersimpan, tombol skor tertinggi untuk melihat skor tertinggi yang diperoleh pemain, tombol petunjuk permainan untuk menuju halaman dan tombol keluar dari program



Gambar 4.3 Halaman Halaman Utama

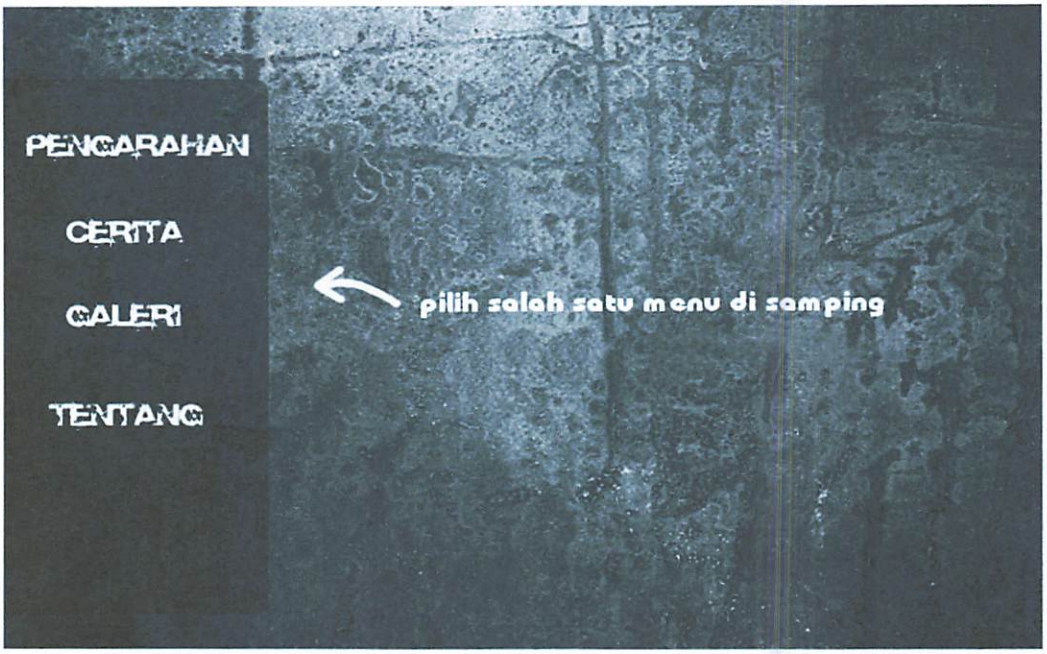
```
// untuk me-load BGM (Background Music)
var bgMenu:Sound = new Sound();
bgMenu.attachSound("BGMStage"); // untuk load music dengan nama BGMStage dari library
bgMenu.start(0, 0); // untuk memainkan music

// setting variabel awal
var stage = 1; // stage
var playerLives = 3; // nyawa
var maxHP = 200; // maksimum HP pemain
var playerHP = maxHP; // HP pemain
var skor = 0; // skor pemain
var prevSkor = skor;
var prevHP = playerHP;
var prevLives = playerLives;

// load SharedObject (local) dengan nama highScore
// jika belum ada, buat 1 terlebih dahulu
var hS:SharedObject = SharedObject.getLocal("highScore");

if (hS.data.player1Name == null) {
        trace ("object doesn't exit, create a new one");
        hS.data.player1Name = "player1";
        hS.data.player1Skor = 0;
```

```
        hS.data.player2Name = "player2";
        hS.data.player2Skor = 0;

        hS.data.player3Name = "player3";
        hS.data.player3Skor = 0;

        hS.data.player4Name = "player4";
        hS.data.player4Skor = 0;

        hS.data.player5Name = "player5";
        hS.data.player5Skor = 0;

}
        namaPlayer1 = hS.data.player1Name;
        skorPlayer1 = hS.data.player1Skor;

        namaPlayer2 = hS.data.player2Name;
        skorPlayer2 = hS.data.player2Skor;

        namaPlayer3 = hS.data.player3Name;
        skorPlayer3 = hS.data.player3Skor;
```

```
        namaPlayer4 = hS.data.player4Name;
        skorPlayer4 = hS.data.player4Skor;

        namaPlayer5 = hS.data.player5Name;
        skorPlayer5 = hS.data.player5Skor;

stop();

// event onEnterFrame adalah event yang akan dijalankan setiap saat
// selama berada di dalam frame ini
this.onEnterFrame = function () {
        // jika menekan tombol ESCAPE, maka tampilkan menu untuk keluar dari permainan
        if (Key.isDown(Key.ESCAPE)) {
                // frame ke-5 adalah frame berisi menu keluar dari permainan
                _root.gotoAndPlay(5);
        }
}
```

4.3.4 Pengujian Halaman Petunjuk dan pengarahan Permainan

Halaman petunjuk digunakan sebagai pedoman tombol mana saja yang perlu ditekan saat permainan game Helishoter

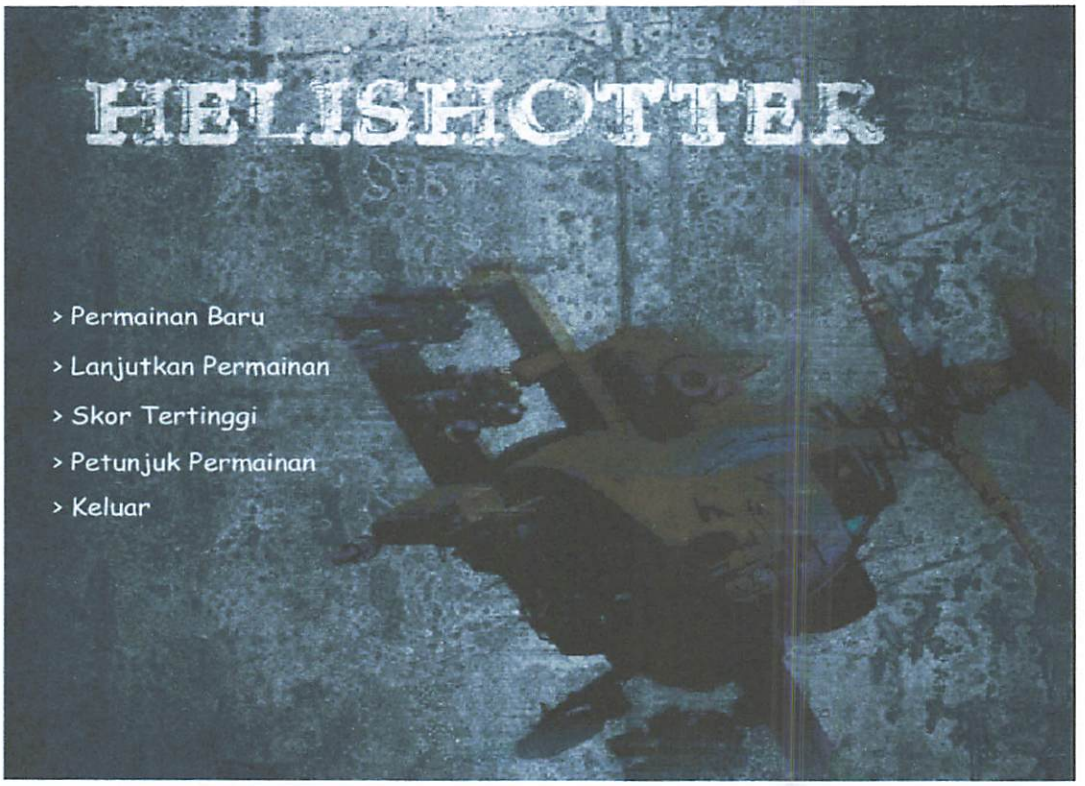


Gambar 4.4 Halaman Petunjuk Permainan dan pengarahan

```
stop();
this.onEnterFrame = function() {}
```

4.3.5 Pengujian Halaman Perang, Level,simpan,Game Over dan usai permainan

Klik tombol awal permainan untuk menuju halaman perang dan memulai permainan dari awal, pemain tidak bisa menuju halaman lanjutkan permainan bila belum melakukan penyimpanan permainan. Dan disetiap level permainan tampilan background akan selalu berbeda.



Gambar 4.5 Halaman Utama game heli shotter

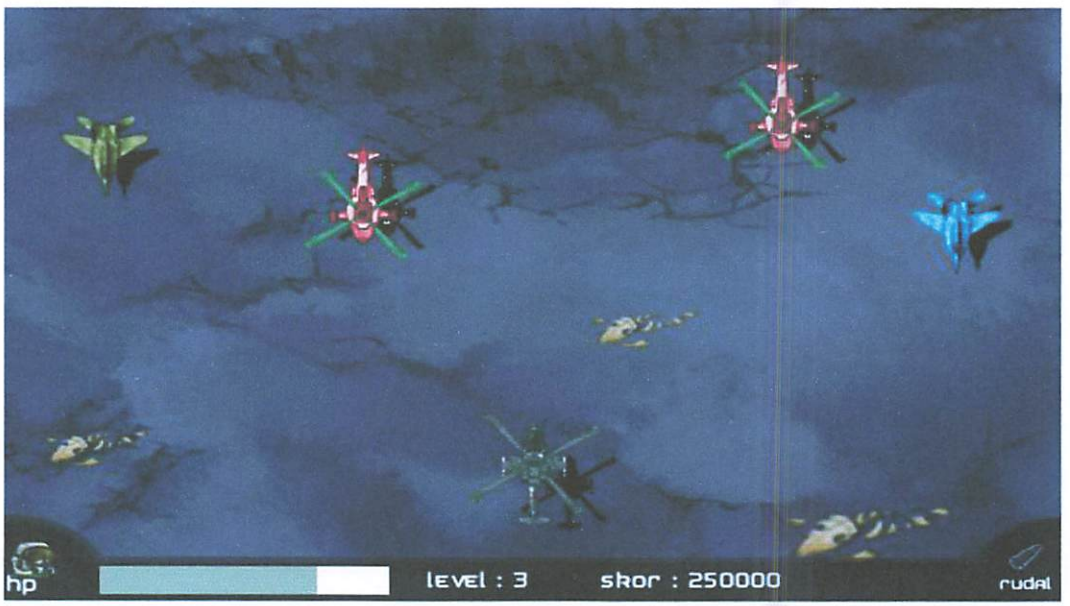
Game Helishotter ketika mamesuki level 1 (satu) dengan Background padang pasir

Gambar 4.6 Halaman game Heli shotter level 1

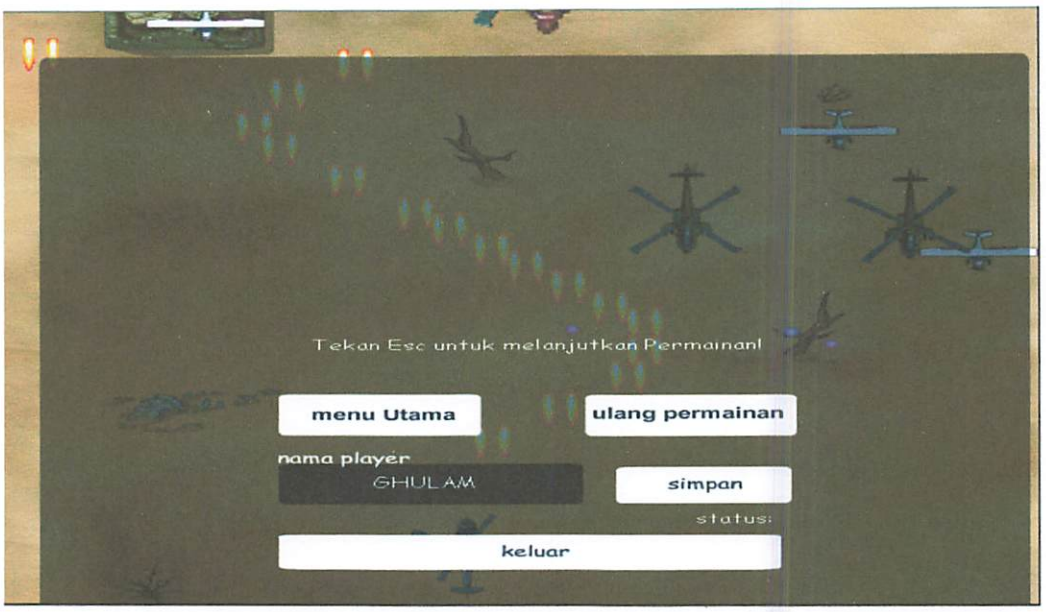
Game Helishotter ketika mamesuki level 2 (satu) dengan Background Hutan Vietnam

Gambar 4.7 Halaman game Heli shotter level 2

Game Helishotter ketika mamesuki level 3 (tiga) dengan Background Level Gunung bebatuan

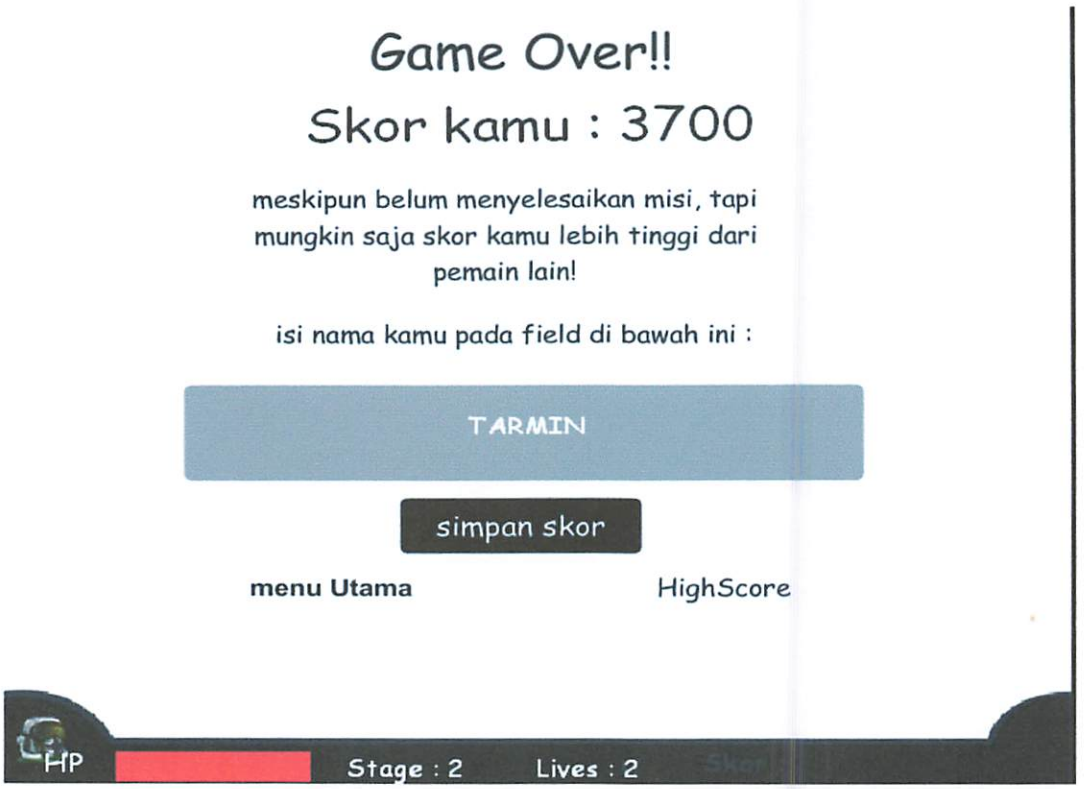


Gambar 4.8 Halaman game Heli shotter level 3



Gambar 4.9 Halaman melanjutkan permainan

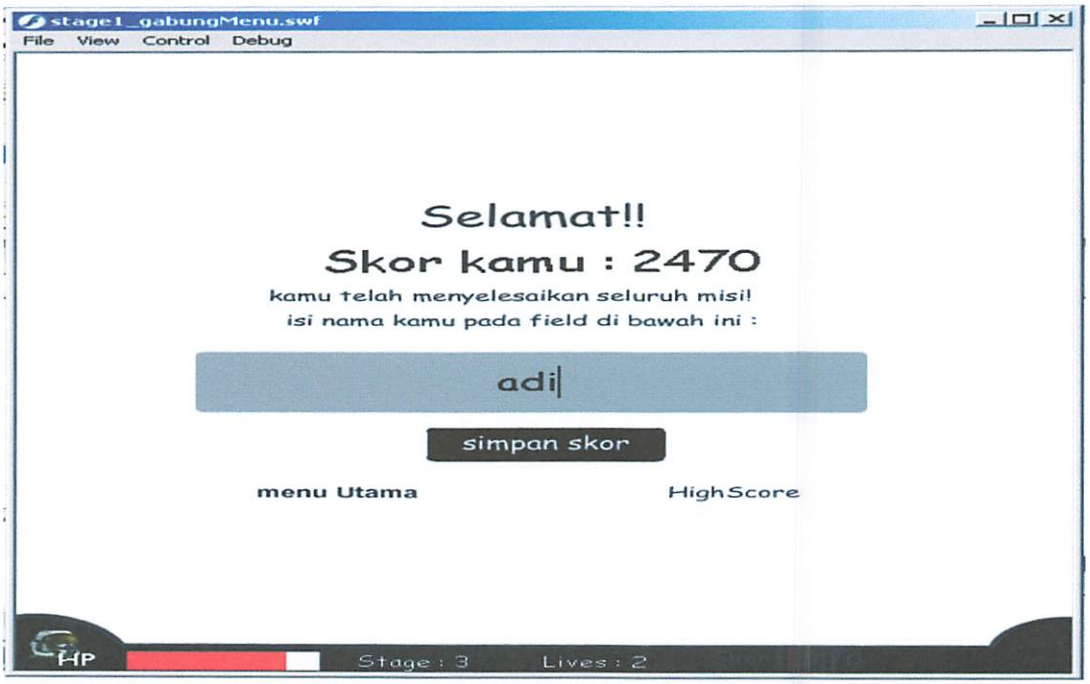
Selama permainan berlangsung pemain dapat menyimpan permaianan dan melanjutkan permaianan yang telah tersimpan, Untuk melakukan peyimpanan tekan tombol simpan, maka secara otomatis pemain akan menuju halaman simpan permainan, pada halaman ini terdapat tombol menu utama, untuk menuju halaman utama, ulang permainan untuk mengulang permainan dan keluar untuk mengakhiri permainan



Gambar 4.10 Halaman Game Over

Bila pemaian gagal menyelesaikan permainan maka akan tampil lembar halaman movie clip permainan Game Over dan menampilkan skor yang telah didapat selama permainan serta input untuk diisi oleh pemain

```
// GAME OVER !
// set teks skor sesuai skor yang didapat
this.txtFinalSkor.text = "Skor kamu : "+_root.skor;
stop();
this.onEnterFrame = function() {}
```

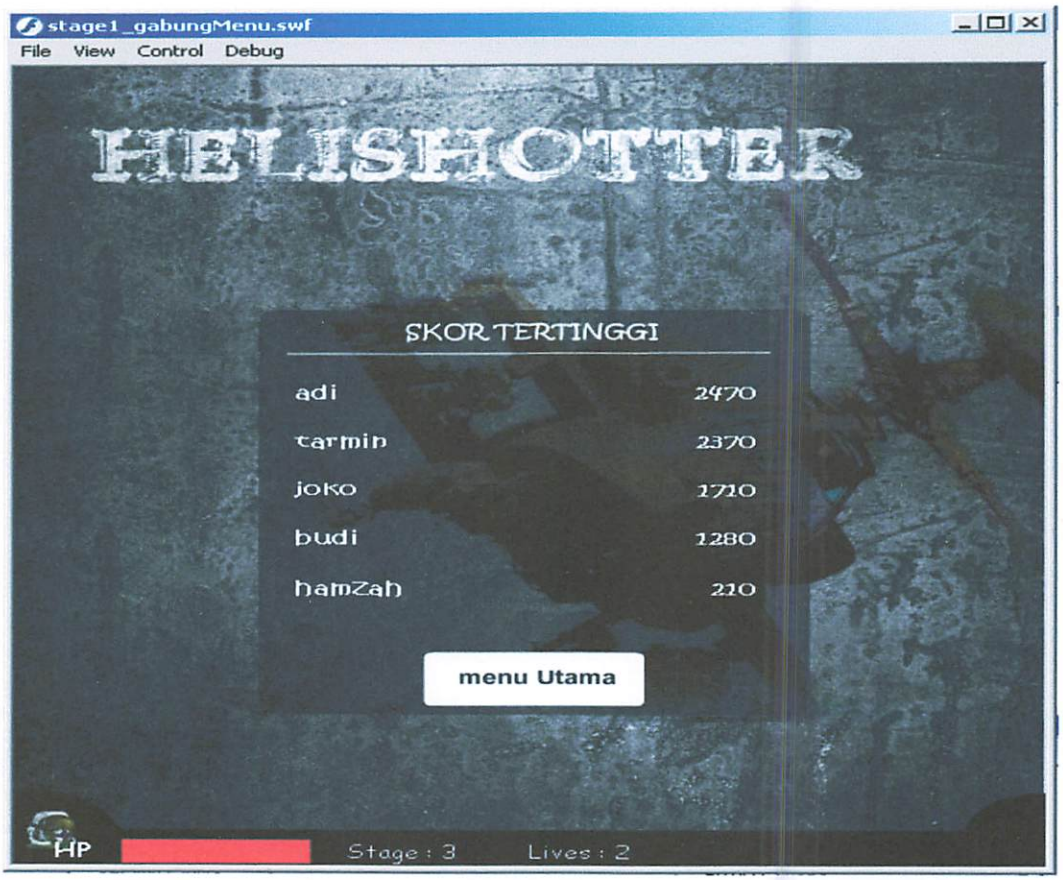


Gambar 4.11 Halaman Permaian Usai

Bila pemain dapat menyelesaikan permainan, maka halaman tersebut akan memunculkan skor tertinggi. Untuk mengetahui apakah skor didapat termasuk dalam skor tertinggi atau tidak, yang telah didapat selama permainan serta input untuk diisi oleh pemain.

4.3.6 Pengujian Halaman Skor Tertinggi

Halaman skor tertinggi, halaman menampilkan sebuah skor dalam game Helishooter



Gambar 4.12 Halaman skor tertinggi

```
// SKOR //
// load SharedObject (local) dengan nama highScore
var hS:SharedObject = SharedObject.getLocal("highScore");
/*
if (hS.data.player1Name == null) {
        trace ("object doesn't exit, create a new one");
        hS.data.player1Name = "player1";
        hS.data.player1Skor = 0;
        hS.data.player2Name = "player2";
        hS.data.player2Skor = 0;
```

```
        hS.data.player3Name = "player3";
        hS.data.player3Skor = 0;
        hS.data.player4Name = "player4";
        hS.data.player4Skor = 0;
        hS.data.player5Name = "player5";
        hS.data.player5Skor = 0;
} else {


}
*/
        namaPlayer1 = hS.data.player1Name;
        skorPlayer1 = hS.data.player1Skor;
        namaPlayer2 = hS.data.player2Name;
        skorPlayer2 = hS.data.player2Skor;
        namaPlayer3 = hS.data.player3Name;
        skorPlayer3 = hS.data.player3Skor;

        namaPlayer4 = hS.data.player4Name;
        skorPlayer4 = hS.data.player4Skor;
        namaPlayer5 = hS.data.player5Name;
        skorPlayer5 = hS.data.player5Skor;
stop();
this.onEnterFrame = function() {}
```

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah menyelesaikan pengembangan game Heli shooter menggunakan action script berbasis macromedia flash, penulis menarik kesimpulan sebagai berikut:

1. Perangkat lunak Macromedia Flash memungkinkan pemakai (*user*) komputer untuk memainkan permainan *Heli Shotter*
2. Perangkat lunak Macromedia Flash mempunyai bahasa skrip yang diberi nama Action script. Action script adalah menunjukkan kolektif set dari fanction, event dan event handler yang memungkinkan dikembangkan oleh developer untuk menbuat flash movie atau Game yang lebih komplek dan interaktif.

5.2 SARAN

Penulis memberikan beberapa saran sebagai berikut

1. Game dapat dikembangkan pada sisi pengendali menggunakan stick.
2. Game dapat dikembangkan untuk dua pemain atau yang lebih banyak.

DAFTAR PUSTAKA

1. PC MILD Tabloid sumber edisa 07/05 tanggal 14 maret2005
2. Sunyoto Andi, Adode Flash + XML= Rich Multimedia Application
3. W wandah, Dasar Pemrogramam Flash Game
4. Pranita Oktri yanis, Flash MX untuk Pemula
5. Prasetio Arno Dimas, Kenalan dengan action script 02.
6. Drajat. 2009. *Panduan Belajar Flash untuk Pemula.* Yogyakarta: Penerbit MediaKom.
7. pengertian-action-script http://edo-brenk.blogspot.com
8. sejarah-game http://inron01.blogspot.com
9. macam-macam-game komputer http://tutorialkuliah.blogspot.

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAM TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA	:	TARMIN
NIM	:	06.12.516
JURUSAN	:	TEKNIK ELEKTRO S-1
KONSENTRASI	:	Teknik Komputer dan Informatika
JUDUL SKRIPSI	:	**PENGEMBANGAN GAME HELI SHOTTER MENGGUNAKAN ACTION SKRIPT ACTION BERBASIS MACROMEDIA FLASH**

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari/Tanggal : selasa / 23 Agustus 2011

Dengan nilai : 76,3 (B+)

Panitia Ujian Skripsi

Ketua Majelis Penguji

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y. 1018800189

Sekretaris Majelis Penguji

Dr.Eng. Aryuanto Soetedjo, ST, MT
NIP.Y. 1030800417

Anggota Penguji

Dosen Penguji I

Sandy Nataly Mantja,S.kom
NIP.P. 1030800418

Dosen Penguji II

Irmalia Suryani Faradisa. ST, MT
NIP.P. 1030000365

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Karanglo KM.2 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi Teknik Komputer dan Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

Nama : TARMIN
Nim : 06.12.516
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Komputer dan Informatika S-1
Judul : PENGEMBANGAN GEME HELI SHOTTER MENGGUNAKAN ACTION SCRIPT BERBASIS MACROMEDIA FLASH

Tanggal	Uraian	Paraf
Penguji I 23 Agustus 2011	1. Tidak perlu dijelaskan game dari berbagai generasi pada bab II 2. Kesimpulan salah, kesimpulan hasil pengujian bab IV 3. Bab II yang tidak menunjang program tidak perlu ditulis. Contoh Racing, sesuaikan dengan program saja 4. Perbaikan Flowchard	
Penguji II 23Agustus 2011	1. Kesimpulan tidak sesuai dengan tujuan Bab IV	

Disetujui :

Dosen Penguji I

Sandy Nataly Mantja S.kom
NIP.P. 1030800418

Dosen Penguji II

Irmalia S. Faradisa, ST,MT.
NIP.P.1030000365

Mengetahui :

Dosen Pembimbing I

Ibrahim Ashari,ST, MT.
NIP.P. 1030100358

Dosen Pembimbing II

Ahmad Faisol, ST.
NIP.P.1030000431

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : TARMIN
Nim : 06.12.516
Masa Bimbingan : 07 Juni 2011 s/d 16 Juli 2011
Judul Skripsi : Pengembangan Game Heli Shotter Menggunakan Action Script Berbasis Mcromedia Flash

No.	Tanggal	Uraian	Parap Pembimbing
1.	04 juli 2011	Acc Bab I	
2.	07 Juli 2011	Acc Bab II	
3.	08 Juli 2011	Acc Bab III	
4.	10 Juli 2011	Revisi Bab IV	
5.	13 Juli 2011	Acc Bab V	
6.	16 Juli 2011	Acc Bab IV	
7.	16 Juli 2011	Acc Makalah seminar	
8.			
9.			
10.			

Malang, 10 Oktober
Dosen Pembimbing,

Ibrahim Ashari,ST,MT
NIP.P.1030100358

Form.S-4b

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : TARMIN
Nim : 06.12.516
Masa Bimbingan : 07 Juni 2011 s/d 16 Juli 2011
Judul Skripsi : Pengembangan Game Heli Shotter Menggunakan Action Script Berbasis Mcromedia Flash

No.	Tanggal	Uraian	Parap Pembimbing
1.	05 juli 2011	Acc bab I	
2.	08 Juli 2011	Acc Bab II	
3.	09 Juli 2011	Revisi Bab III	
4.	11 Juli 2011	Acc Bab IV dan Bab III	
5.	14 Juli 2011	Acc Bab V	
6.	16 Juli 2011	Acc Makalah seminar	
7.			
8.			
9.			
10.			

Malang,
Dosen Pembimbing,

Ahmad Faisol,ST,
NIP.P.1031000431

Form.S-4b

SCRIPT GAME

1.Script Loading

```
// stop() digunakan untuk menghentikan animasi
frame
stop();
/*        skrip di bawah ini untuk melakukan loading
loading adalah suatu proses untuk memuat/ me-load
komponen-komponen
berupa gambar, suara, movieClip, dsb yang
digunakan oleh game
variabel angka untuk menyimpan nomor frame pada
movieClip loading_mc
variabel persen_tampil adalah dynamic_text yang
berisi keterangan
sudah berapa persen proses loading berjalan
variabel persen untuk menyimpan nilai persen yang
sudah di-load oleh flash
perhitungannya dengan pembulatan (round) jumlah
byte yang telah didownload
dibagi dengan total yang harus didownload dikali 100
jika persen dimoduluskan oleh 10 hasilnya 0, maka
persen adalah kelipatan 10,
maka tambahkan variabel angka menjadi nomor
frame berikutnya. lalu load animasi
pada frame ke-angka pada movieClip loading_mc.
update teks persen_tampil menjadi persen yang sudah
di-load
*/
angka = 0;
persen_tampil = 0+"%";
_root.onEnterFrame = function() {
if (Math.round(_root.getBytesTotal()) <=
Math.round(_root.getBytesLoaded())) {
gotoAndStop(2);
} else {
persen =
Math.round(Math.round(_root.getBytesLoaded())/Ma
th.round(_root.getBytesTotal())*100);
if (persen%10 == 0) {
angka = (persen/10)+1;
loading_mc.gotoAndStop(angka);
persen_tampil = persen+"%";
}
}
};
```

2. Skrip BGM music dan Menu

```
// untuk me-load BGM (Background Music)
var bgMenu:Sound = new Sound();
bgMenu.attachSound("BGMStage"); // untuk load
music dengan nama BGMStage dari library
bgMenu.start(0, 0); // untuk memainkan music
// setting variabel awal
var stage = 1; // stage
```

```
hS.data.player1Skor = 0;
hS.data.player2Name = "player2";
hS.data.player2Skor = 0;
hS.data.player3Name = "player3";
hS.data.player3Skor = 0;
hS.data.player4Name = "player4";
hS.data.player4Skor = 0;
hS.data.player5Name = "player5";
hS.data.player5Skor = 0
} else {
}
*/
namaPlayer1 = hS.data.player1Name;
skorPlayer1 = hS.data.player1Skor;
namaPlayer2 = hS.data.player2Name;
skorPlayer2 = hS.data.player2Skor;
namaPlayer3 = hS.data.player3Name;
skorPlayer3 = hS.data.player3Skor;
namaPlayer4 = hS.data.player4Name;
skorPlayer4 = hS.data.player4Skor;
namaPlayer5 = hS.data.player5Name;
skorPlayer5 = hS.data.player5Skor;
stop();
this.onEnterFrame = function() {}
```

4.Skrip Level

```
// stop frame
stop();
// stop BGM menu
_root.bgMenu.stop();
stage = 1;
var radians:Number = Math.PI/180;
var win = false;
var finished = false;
var finishStage1 = 8;
var currentRepeat = 0;
var bgStage1_speed = 3;
var started = false;
var movedRight = false;
var moveLeft;
var init;
var gamePaused = false;
// Boss
var seeTheBoss = false;
var bossCreated = false;
var bossHP = 500;
var bossMovedRight = false;
var bossMovedUp = false;
var bossReadyToFire = false;
var bossTimer = 0;
var bossFireDelay = 5000;
var bossBulletNumber = 8;
var bossBulletOffset = 20;
var bossBulletSpeed = 5;
var bossBullets = [];
var bossBulletSpeedX = [];
var bossBulletSpeedY = [];
var bossBulletCnt = 0;
var bossBulletDamage = 10;
// end boss
// Sounds
var playerFiring:Sound = new Sound();
playerFiring.attachSound("playerFiring");
var stageBGM:Sound = new Sound();
stageBGM.attachSound("BGMStage");
var musuhExplode:Sound = new Sound();
musuhExplode.attachSound("mediumExplosion");
var gotBonus:Sound = new Sound();
gotBonus.attachSound("gotBonus");
var bossExplode:Sound = new Sound();
bossExplode.attachSound("bigExplode");
// end Sounds
// Player's Data
var playerSpeed = 5;
var playerReady = false;
skor = prevSkor;
playerLives = prevLives;
playerHP = prevHP;
var bulletDamage = 20;
var bulletSpeed = 4;
var bulletReady = true;
var bulletDelay = 150;
var bulletArray = [];
var bulletCount = 0;
// end Player's Data
// Bonuses
var bonusHP = 3;
var bonusHPAmount = 50;
var bonusBulletSpeed = 4;
var bonusBulletSpeedAmount = 10;
var bonusBulletSpeedTime = 15000; // 15 detik
var bonusBulletSpeedTimer;
var bonus = [];
var bonusHPFrame = [];
var bonusBulletInFrame = [];
var bonusType = [];
var bonusRudal = 3;
var rudalDamage = 10;
var rudalAmount = 10;
// end Bonuses
// Musuh #1
var maxMusuh1 = 4;
var musuh1Damage = 10;
var musuh1Poin = 10;
var musuh1Speed = 4;
var musuh1Count = 0;
var musuh1Array = [];
var musuh1BulletSpeed = 350;
var musuh1BulletCount = 0;
var musuh1BulletArray = [];
var musuh1BulletSpeedX= [];
var musuh1BulletSpeedY = [];
var musuh1BulletReady = [];
```

```
var musuh1BulletDamage = 5;
// end Musuh #1
// Musuh #2
var maxMusuh2 = 3;
var musuh2Damage = 15;
var musuh2Poin = 20;
var musuh2Speed = 4;
var musuh2Count = 0;
var musuh2Array = [];
var musuh2HP = []; // array
var musuh2hp = 50; // konstanta HP
var musuh2BulletSpeed = 380;
var musuh2BulletCount = 0;
var musuh2BulletArray = [];
var musuh2BulletSpeedX= [];
var musuh2BulletSpeedY = [];
var musuh2BulletReady = [];
var musuh2BulletDamage = 10;
// Musuh #2
//Mouse.hide();
_root.bgStatus.txtLives.text = "Lives : "+playerLives;
_root.bgStatus.txtStage.text = "Level : 1";
_root.bgStatus.swapDepths(50000);
_root.hitEffectColor._visible = false;
_root.pauseBG._visible = false;
_root.pauseBG.swapDepths(100000);
stageBGM.start(0,0);
/*
fungsi iniStage1()
digunakan untuk melakukan inisialisasi permainan
pada stage 1
didalamnya memanggil fungsi clears dengan
parameter "all"
*/
function initStage1() {
clears("all");
}
// panggil fungsi initStage1()
initStage1();
/*
perulangan untuk melakukan random bonus.
logika yang digunakan adalah merandom jumlah
maksimum perulangan frame
pada stage 1, dan menyimpannya ke dalam variabel
bonusHPFrame
*/
while (bonusHPFrame.length < bonusHP) {
same = false;
var i = random(finishStage1);
for (j = 0; j < bonusHPFrame.length; j++) {
if (bonusHPFrame[j] == i) {
same = true;
break;
}
}
if (same == false)
bonusHPFrame.push(i);
}
/*
perulangan untuk melakukan random kecepatan
peluru.
logika yang digunakan adalah merandom jumlah
maksimum perulangan frame
pada stage 1, dan menyimpannya ke dalam variabel
bonusBulletInFrame
*/
while (bonusBulletInFrame.length <
bonusBulletSpeed) {
same = false;
var i = random(finishStage1);
for (j = 0; j < bonusBulletInFrame.length; j++) {
if (bonusBulletInFrame[j] == i) {
//trace (bonusBulletInFrame[j]+","+i);
same = true;
break;
}
}
if (same == false)
bonusBulletInFrame.push(i);
}
/*
fungsi randomBonus
parameter musuh dengan tipe MovieClip
melakukan random bonus HP dan kecepatan peluru.
jika HP pemain < 50, maka fungsi akan mencoba
untuk memunculkan
bonus HP dengan syarat bonusHP masih tersisa.
begitu pula dengan bonus kecepatan peluru, akan
dimunculkan sesuai random frame
yang sebelumnya telah dilakukan pada perulangan di
atas.
*/
function randomBonus(musuh:MovieClip) {
//trace (bonusHP);
// kasi bonus HP jika HP <= 50
if (playerHP <= 50) {
if (bonusHP > 0) {
if (musuh._y < 300) {
//trace (" bonus HP");
var hp = this.attachMovie("bonusHP",
"bonusHP_"+18000, 18000);
hp._x = musuh._x;
hp._y = musuh._y;
bonusHP--;
bonus.push(hp);
bonusType.push("HP");
bonusHPFrame.splice(i, 1);
}
}
} else {
if (bonusHP > 0) {
// trace ("kasi bonus hp secara random");
```

```
// kasi bonus HP secara random
for (i  = 0; i < bonusHPFrame.length; i++) {
if (currentRepeat == bonusHPFrame[i]) {
if (musuh._y < 300) {
//trace (" bonus HP");
var hp = this.attachMovie("bonusHP",
"bonusHP_"+random(1000), 18000);
hp._x = musuh._x;
hp._y = musuh._y;
bonusHP--;
bonus.push(hp);
bonusType.push("HP");
bonusHPFrame.splice(i, 1);
break;
}
}
}
}
if (bonusBulletSpeed > 0) {
// trace ("kasi bonus speed secara random");
// kasi bonus HP secara random
for (i  = 0; i < bonusBulletInFrame.length; i++) {
if (currentRepeat == bonusBulletInFrame[i]) {

if (musuh._y < 300) {
//trace ("bonus Speed");
var speed = this.attachMovie("bonusBulletSpeed",
"bonusBulletSpeed_"+random(1000), 18000);
speed._x = musuh._x;
speed._y = musuh._y;
bonusBulletSpeed--;
bonus.push(speed);
bonusType.push("bulletSpeed");
bonusBulletInFrame.splice(i, 1);
break;
}
}
}
}
}
//trace (bonus);
}
/*
fungsi untuk menggerakkan bonus (jika ada)
*/
function movesBonus() {
if (bonus.length > 0) {
//trace ("bonus is more than 0");
for (i = 0; i < bonus.length; i++) {

bonus[i]._y += 2;
if (bonus[i]._y > 600) {
removeMovieClip(bonus[i]);
bonus.splice(i, 1);
bonusType.splice(i, 1);
}
//trace ("test");
if (bonus[i].hitTestShape( _root.player ) ) {
//trace ("hitting player");
//trace (bonusType[i]);
if (bonusType[i] == "HP") {
//trace ("get bonus HP");
//trace (playerHP);
removeMovieClip(bonus[i]);
bonus.splice(i, 1);
bonusType.splice(i, 1);
playerHP += bonusHPAmount;
if (playerHP + bonusHPAmount >= maxHP)
playerHP = maxHP;
//trace (playerHP);
} else if (bonusType[i] == "bulletSpeed") {
//trace ("get bonus Speed");
removeMovieClip(bonus[i]);
bonus.splice(i, 1);
bonusType.splice(i, 1);
bulletSpeed = bonusBulletSpeedAmount;
bonusBulletSpeedTimer =
setInterval(stopBonusBulletSpeed,
bonusBulletSpeedTime);
//trace ("got bonus bullet");
}
gotBonus.start(0, 0);
}
}
} else {
//trace ( "bonus is 0 " );
}

//trace ( "test" );
}
/*
fungsi initPlayer
digunakan untuk inisialisasi posisi dan HP pemain
*/
function initPlayer() {
playerHP = 200;
_root.player._x = 242;
_root.player._y = 476;
_root.player.gotoAndPlay(1);
//trace( _root.player._currentframe );
}
/*
fungsi createBossBullet
digunakan untuk membuat peluru Boss
tetap menggunakan fungsi attachMovie
arah gerakan peluru boss ditentukan oleh perhitungan
tertentu menggunakan sudut
sudut ditentukan oleh offset tertentu, dan akan
menghasilkan nilai
perpindahan x dan y (bossBulletSpeedX dan
bossBulletSpeedY)
*/
```

```
function createBossBullet() {
for (i = 0; i < bossBulletNumber; i++) {

var musuhBullet = this.attachMovie("musuh2Bullet",
"bossBullets_"+bossBulletCnt,
20000+bossBulletCnt);
musuhBullet._x = _root.stage1Boss._x + 5 *
Math.cos(90 * radians);
musuhBullet._y = _root.stage1Boss._y + 5 *
Math.sin(90 * radians) + _root.stage1Boss._height/2;
//trace (musuhBullet._x);
//calculate random bullet offset.
//var randomNum:Number =
random(bossBulletOffset)-(bossBulletOffset/2);
//set bullet firing angle
var bulletAngle:Number = (170 - (i *
bossBulletOffset)) * radians;
//set bullet speed based on angle
var bossBulletX = Math.cos(bulletAngle) *
bossBulletSpeed;
var bossBulletY = Math.sin(bulletAngle) *
bossBulletSpeed;
//add bullet to bulletArray
bossBulletSpeedX.push(bossBulletX);
bossBulletSpeedY.push(bossBulletY);
bossBullets.push(musuhBullet);
bossBulletCnt++;
//trace(musuhBullet);
}
}
/*
fungsi untuk stop bonus kecepatan peluru pemain
clearInterval digunakan untuk menghapus timer yang
telah dibuat
*/
function stopBonusBulletSpeed() {
bulletSpeed = 4;
clearInterval(bonusBulletSpeedTimer);
//trace ("bonus Speed ended");
}
/*
fungsi moveBossBullet()
digunakan untuk menggerakkan peluru boss,
berdasarkan nilai bossBulletSpeedX
dan bossBulletSpeedY
*/
function moveBossBullet() {
if (bossBullets.length > 0) {
for (i = 0; i < bossBullets.length; i++) {
//trace ( bossBullets[i] );
bossBullets[i]._x += bossBulletSpeedX[i];
bossBullets[i]._y += bossBulletSpeedY[i];
if ( bossBullets[i].hitTestShape( _root.player ) ) {
playerHP -= bossBulletDamage;
hitted = true;
hitEffect();
updateLives();
removeMovieClip(bossBullets[i]);
bossBullets.splice (i, 1);
bossBulletSpeedX.splice(i, 1);
bossBulletSpeedY.splice(i, 1);
}
if (bossBullets[i]._x > 550 ||
bossBullets[i]._y > 600) {

removeMovieClip(bossBullets[i]);
bossBullets.splice (i, 1);
bossBulletSpeedX.splice(i, 1);
bossBulletSpeedY.splice(i, 1);
}
}
}
}
/*
fungsi createBullet
digunakan untuk membuat peluru pemain dengan
memanggil fungsi attachMovie
peluru2 yang dibuat dimasukkan ke dalam array
dengan perintah push()
setiap menembak, mainkan suara tertentu
*/
function createBullet() {
var playerBullet1 =
_root.playerBullet1.attachMovie("playerBullet1",
"playerBullet1_"+bulletCount, 1000+bulletCount);
bulletCount++;
playerBullet1._x = _root.player._x + 16;
playerBullet1._y = _root.player._y + 10;
bulletArray.push(playerBullet1);
// this.attachSound("playerFiring");
playerFiring.start(0, 0);
}
/*
fungsi movePlayerBullet
digunakan untuk menggerakkan peluru pemain
didalamnya akan memanggil fungsi createBullet,
yang digunakan untuk
membuat peluru pemain
peluru hanya dibuat jika pemain menekan spasi dan
peluru dalam kondisi ready
artinya peluru siap untuk dibuat lagi (untuk
menghindari jarak antar
peluru yang terlalu dekat)
peluru pemain akan hilang jika :
1. melewati batas jendela permainan (_y < (0 - tinggi
peluru)
2. jika mengenai musuh/ boss (hitTestShape(musuh))
*/
function movePlayerBullet() {
if ( bulletReady && Key.isDown(Key.SPACE) ) {
bulletReady = false;
currentTime = getTimer();
```

```
createBullet();
} else {
if ( currentTime+bulletDelay <= getTimer() ) {
bulletReady = true;
}
}
for (var i = 0; i < bulletArray.length; i++) {
bulletArray[i]._y -= bulletSpeed;

if (bulletArray[i]._y < (0-bulletArray[i]._height)) {
removeMovieClip(bulletArray[i]);
bulletArray.splice (i, 1);
}
for (j = 0; j < musuh1Array.length; j++) {
if ( bulletArray[i].hitTestShape(musuh1Array[j]) ) {
// is it worth to give a bonus?
randomBonus(musuh1Array[j]);
skor += musuh1Poin;
updateSkor();
musuh1Array[j].gotoAndPlay(2);
removeMovieClip(bulletArray[i]);
musuh1Array.splice(j, 1);
bulletArray.splice(i, 1);
musuhExplode.start(0, 0);
break;
}
}
//continue;
for (j = 0; j < musuh2Array.length; j++) {
if ( bulletArray[i].hitTestShape(musuh2Array[j]) ) {
musuh2HP[j] -= bulletDamage;
//                                    trace
(musuh2Array[j].HP);
if (musuh2HP[j] <= 0) {
// is it worth to give a bonus?
randomBonus(musuh2Array[j]);
skor += musuh2Poin;
updateSkor();
musuh2Array[j].gotoAndPlay(10);
musuh2HP.splice(j, 1);
musuh2Array.splice(j, 1);
musuhExplode.start(0, 0);
} else {
musuh2Array[j].gotoAndPlay(8);
}
removeMovieClip(bulletArray[i]);
bulletArray.splice(i, 1);
break;
}
}
if ( seeTheBoss ) {
if (bulletArray[i].hitTestShape(_root.stage1Boss) ) {
bossHP -= bulletDamage;

removeMovieClip(bulletArray[i]);
bulletArray.splice(i, 1);
_root.stage1Boss.gotoAndPlay(6);
if ( bossHP <= 0 ) {
win = true;
}
}
}
}
}
/*
function movePlayer
digunakan untuk menggerakkan pesawat pemain
menerima input keyboard berupa panah KIRI,
KANAN, ATAS dan BAWAH
fungsi ini juga membatasi area permainan (agar
pemain tidak keluar jendela)
*/
function movePlayer() {
if (playerReady == true) {
if (Key.isDown(Key.LEFT)) {
_root.player._x -= _root.playerSpeed;
}
if (Key.isDown(Key.RIGHT)) {
_root.player._x += _root.playerSpeed;
}
if (Key.isDown(Key.DOWN)) {
_root.player._y += _root.playerSpeed;
}
if (Key.isDown(Key.UP)) {
_root.player._y -= _root.playerSpeed;
}
}
//trace (player._x);
//last_x = _root.player._x;
if (_root.player._x < (_root.player._width/2) ) {
_root.player._x += playerSpeed;
}
if (_root.player._x > 550 - (_root.player._width/2) ) {
_root.player._x -= playerSpeed;
}
if (_root.player._y < 20 ) {
_root.player._y += playerSpeed;
}
if (_root.player._y > 570 ) {
_root.player._y -= playerSpeed;
}

}
/*
fungsi enemyFire
parameter enemy bertipe MovieClip,
digunakan untuk mengetahui musuh mana yg sedang
menembak
parameter musuh bertipe String
digunakan untuk mengetahui musuh mana yg sedang
menembak
```

```
digunakan untuk membuat peluru dari musuh,
menggunakan attachMovie
peluru yang ditembakkan mengarah ke pesawat
pemain. Hal ini dilakukan dengan
menggunakan teori pythagoras dan trigonometri
lainnya.

fungsi ini menghasilkan bulletSpeedX dan
bulletSpeedY untuk masing2 peluru
yang ditembakkan.
*/
function enemyFire(enemy:MovieClip,
musuh:String) {
if ( musuh == "musuh1")
var musuhBullet = this.attachMovie("musuh1Bullet",
"musuh1Bullet_"+musuh1BulletCount,
3000+musuh1BulletCount);
else if ( musuh == "musuh2")
var musuhBullet = this.attachMovie("musuh2Bullet",
"musuh2Bullet_"+musuh2BulletCount,
6000+musuh2BulletCount);
musuhBullet._x = enemy._x;
musuhBullet._y = enemy._y + 10;
//trace (musuh1Bullet);
var dx = (_root.player._x + 34) - musuhBullet._x;
// distance to other object on x-axis
var dy = (_root.player._y+30) - musuhBullet._y;
// distance on y-axis
var dist = Math.sqrt(dx*dx + dy*dy);

var angle;
// figure out angle in radians (0- 2*PI)
if (dy <0)
angle = Math.PI*2-Math.acos(dx/dist);
else
angle = Math.acos(dx/dist);
//trace (angle);
if ( musuh == "musuh1") {
musuh1BulletSpeedX.push( Math.cos( angle ) *
radians * musuh1BulletSpeed );
musuh1BulletSpeedY.push( Math.sin( angle ) *
radians * musuh1BulletSpeed );

musuh1BulletCount++;
musuh1BulletArray.push(musuhBullet);
} else if ( musuh == "musuh2") {
musuh2BulletSpeedX.push( Math.cos( angle ) *
radians * musuh2BulletSpeed );
musuh2BulletSpeedY.push( Math.sin( angle ) *
radians * musuh2BulletSpeed );
musuh2BulletCount++;
musuh2BulletArray.push(musuhBullet);
}
//moveEnemyBullet(musuh1Bullet, destX, destY);
}
/*
fungsi createEnemy2()
digunakan untuk membuat jenis musuh nomor 2
*/
function createEnemy2() {
var musuh2 = this.attachMovie("musuh2",
"musuh2_"+musuh2Count, 4000+musuh2Count);
musuh2._x = musuh2._width/2 + random(550 -
musuh2._width/2);
musuh2._y = 0-random(500);
for (var z:Number = 0; z < musuh2Array.length; z++)
{
if ( musuh2.hitTestShape(musuh2Array[z]) ) {
removeMovieClip(musuh2);
createEnemy2();
return;
}
}
for (var z:Number = 0; z < musuh1Array.length; z++)
{
if ( musuh2.hitTestShape(musuh1Array[z]) ) {
removeMovieClip(musuh2);
createEnemy2();
return;
}
}
musuh2Count++;
musuh2Array.push(musuh2);
musuh2HP.push(musuh2hp);
}
/*
fungsi createEnemy1()
digunakan untuk membuat jenis musuh nomor 1
*/
function createEnemy1() {
var musuh1 = this.attachMovie("musuh1",
"musuh1_"+musuh1Count, 2000+musuh1Count);

musuh1._x = musuh1._width/2 + random(550 -
(musuh1._width/2));
musuh1._y = 0-random(500);
for (var z:Number = 0; z < musuh1Array.length; z++)
{
if ( musuh1.hitTestShape(musuh1Array[z]) ) {
removeMovieClip(musuh1);
createEnemy1();
return;
}
}
//enemyFire(musuh1);
//          musuh1BulletReady.push(false);

musuh1Count++;
musuh1Array.push(musuh1);
}

/*
```

```
fungsi moveEnemy1Bullet()
digunakan untuk menggerakan peluru musuh-1
di dalam fungsi ini, dilakukan pemeriksaan apakah
peluru melewati jendela permainan atau peluru
mengenai
sasaran (pesawat pemain). jika hal tersebut terjadi,
maka
hapus peluru musuh-1
*/
function moveEnemy1Bullet() {
if ( musuh1BulletArray.length > 0) {
for (i = 0; i < musuh1BulletArray.length; i++) {
musuh1BulletArray[i]._x +=
musuh1BulletSpeedX[i];
musuh1BulletArray[i]._y +=
musuh1BulletSpeedY[i];
if (musuh1BulletArray[i].hitTestShape(_root.player)
&&
playerReady == true) {
hitted = true;
hitEffect();
playerHP -= musuh1BulletDamage;
updateLives();
removeMovieClip(musuh1BulletArray[i]);
musuh1BulletArray.splice(i, 1);
musuh1BulletSpeedX.splice(i, 1);
musuh1BulletSpeedY.splice(i, 1);
//_root.playerHPBar._xscale = playerHP;
}
if (musuh1BulletArray[i]._y > 600 ||
musuh1BulletArray[i]._x > 550) {
removeMovieClip(musuh1BulletArray[i]);
musuh1BulletArray.splice(i, 1);
musuh1BulletSpeedX.splice(i, 1);
musuh1BulletSpeedY.splice(i, 1);
}
}
}
}
/*
fungsi moveEnemy2Bullet()
digunakan untuk menggerakan peluru musuh-2
di dalam fungsi ini, dilakukan pemeriksaan apakah
peluru melewati jendela permainan atau peluru
mengenai
sasaran (pesawat pemain). jika hal tersebut terjadi,
maka
hapus peluru musuh-2
*/
function moveEnemy2Bullet() {
if ( musuh2BulletArray.length > 0) {
for (i = 0; i < musuh2BulletArray.length; i++) {
musuh2BulletArray[i]._x +=
musuh2BulletSpeedX[i];
musuh2BulletArray[i]._y +=
musuh2BulletSpeedY[i];
if (musuh2BulletArray[i].hitTestShape(_root.player)
&&
playerReady == true) {
hitted = true;
hitEffect();
playerHP -= musuh2BulletDamage;
updateLives();
removeMovieClip(musuh2BulletArray[i]);
musuh2BulletArray.splice(i, 1);
musuh2BulletSpeedX.splice(i, 1);
musuh2BulletSpeedY.splice(i, 1);
//_root.playerHPBar._xscale = playerHP;
}
if (musuh2BulletArray[i]._y > 600 ||
musuh2BulletArray[i]._x > 550) {
removeMovieClip(musuh2BulletArray[i]);
musuh2BulletArray.splice(i, 1);
musuh2BulletSpeedX.splice(i, 1);
musuh2BulletSpeedY.splice(i, 1);
}
}
}
}
/*
fungsi moveEnemy1()
parameter action bertipe String
digunakan untuk menentukan bagaimana musuh
bergerak
digunakan untuk menggerakkan musuh-1
musuh akan memanggil fungsi enemyFire
(menembak) jika
musuh baru muncul di jendela, dan jika posisi y
berada diantara 200 - 204
musuh akan hilang jika melewati jendela permainan
atau
ditabrak oleh pesawat pemain
*/
function moveEnemy1(action:String) {
if (musuh1Array.length > 0) {
for (i = 0; i < musuh1Array.length; i++) {
musuh1Array[i]._y += musuh1Speed;
if ( musuh1Array[i]._y > 0 && musuh1Array[i]._y <
4 ) {
enemyFire(musuh1Array[i], "musuh1");
}

if (musuh1Array[i]._y > 200 &&  musuh1Array[i]._y
< 204 &&
(musuh1Array[i]._y+musuh1Array[i]._height) <
(_root.player._y - 50))
enemyFire(musuh1Array[i], "musuh1");

if (musuh1Array[i]._y > 600) {
removeMovieClip(musuh1Array[i]);
musuh1Array.splice(i, 1);
createEnemy1();
```

```
}

if ( musuh1Array[i].hitTestShape(_root.player) &&
playerReady == true ) {
musuh1Array[i].gotoAndPlay(2);
hitted = true;
hitEffect();
playerHP -= musuh1Damage;
updateLives();
//removeMovieClip(musuh1Array[i]);
musuh1Array.splice(i, 1);
createEnemy1();
}
}
if (action == "new") {
if (musuh1Array.length < maxMusuh1) {
createEnemy1();
}
}
} else {
if (action == "new")
createEnemy1();
}
}

/*
fungsi moveEnemy2()
parameter action bertipe String
digunakan untuk menentukan bagaimana musuh
bergerak
digunakan untuk menggerakkan musuh-2
musuh akan memanggil fungsi enemyFire
(menembak) jika
musuh baru muncul di jendela, dan jika posisi y
berada diantara 200 - 204

musuh akan hilang jika melewati jendela permainan
atau
ditabrak oleh pesawat pemain
*/
function moveEnemy2(action:String) {
if (musuh2Array.length > 0) {
for (i = 0; i < musuh2Array.length; i++) {
musuh2Array[i]._y += musuh2Speed;
if ( musuh2Array[i]._y > 0 && musuh2Array[i]._y <
4 ) {
enemyFire(musuh2Array[i], "musuh2");
}
if (musuh2Array[i]._y > 200 &&  musuh2Array[i]._y
< 204 &&
(musuh2Array[i]._y+musuh2Array[i]._height) <
(_root.player._y - 50))
enemyFire(musuh2Array[i], "musuh2");
if (musuh2Array[i]._y > 600) {
removeMovieClip(musuh2Array[i]);
musuh2Array.splice(i, 1);
createEnemy2();
}

if ( musuh2Array[i].hitTestShape(_root.player) &&
playerReady == true ) {
musuh2Array[i].gotoAndPlay(10);
hitted = true;
hitEffect();
playerHP -= musuh2Damage;
updateLives();
//removeMovieClip(musuh1Array[i]);
musuh2Array.splice(i, 1);
createEnemy2();
}
}
if (action == "new") {
if (musuh2Array.length < maxMusuh2) {
createEnemy2();
}
}
} else {
if (action == "new")
createEnemy2();
}
}
/*
fungsi createTheBoss()
digunakan untuk mempersiapkan kemunculan boss
*/
function createTheBoss() {
_root.stage1Boss._y += 5;
if (_root.stage1Boss._y > 20) //{
bossCreated = true;
//}
}

/*
fungsi moveTheBoss()
digunakan untuk menggerakan boss secara diagonal
*/
function moveTheBoss() {
if (bossMovedRight) {
_root.stage1Boss._x += 2;
if (_root.stage1Boss._x > (550 -
_root.stage1Boss._width/2) )
bossMovedRight = false;
if (bossMovedUp) {
_root.stage1Boss._y += random(3);
if (_root.stage1Boss._y > 50)
bossMovedUp = false;
} else {
_root.stage1Boss._y -= random(3);
if (_root.stage1Boss._y <= 0)
bossMovedUp = true;
}
} else {
```

```
_root.stage1Boss._x -= 2;
if ( _root.stage1Boss._x < (0 +
_root.stage1Boss._width/2) )
bossMovedRight = true;
if (bossMovedUp) {
_root.stage1Boss._y += random(3);
if (_root.stage1Boss._y > 50)
bossMovedUp = false;
} else {
_root.stage1Boss._y -= random(3);
if (_root.stage1Boss._y <= 0)
bossMovedUp = true;
}
}
bossFire();
}
/*
fungsi bossFire
digunakan untuk menentukkan apakah boss siap
untuk menembak atau tidak
siap atau tidaknya ditentukan dari delay yang telah
diatur sebelumnya.
jika delay telah komplit, maka bossReadyToFire akan
diset menjadi true,
lalu boss akan memanggil fungsi createBossBullet()
*/
function bossFire() {
if ( !win ) {
if (bossReadyToFire) {
// fire
trace ( "boss is firing" );
createBossBullet();
moveEnemy2("new");
moveEnemy2("new");
moveEnemy1("new");
bossTimer = getTimer();
bossReadyToFire = false;
} else {
if ( bossTimer + bossFireDelay <= getTimer() ) {
bossReadyToFire = true;
}
}
}
}
/*
function createBonusHP(musuh:MovieClip) {
if (bonusHPAmount == 0) {return;}

var bonusHP = this.attachMovie("bonusHP",
"bonusHP_"+1, 5000);
bonusHP._x = musuh._x;
bonusHP._y = musuh._y;
}*/
/*
fungsi updateSkor()
digunakan untuk melakukan update skor
*/
function updateSkor() {
_root.bgStatus.txtSkor.text = "Skor : "+skor;
}
/*
fungsi updateLives()
digunakan untuk mengupdate nyawa pemain
jika nyawa == 0, maka gameOver dan frame akan
diarahkan ke frame
game Over (frame ke-17) dan seluruh object (musuh,
pemain, peluru) akan
dihapus
*/
function updateLives() {
if (playerHP <= 0) {
_root.player.gotoAndPlay(71);
initPlayer();
//init = setInterval(initPlayer, 1000);
playerLives--;
_root.bgStatus.txtLives.text = "Lives : "+playerLives;
}
if (playerLives == 0) {
clears("all");
_root.gotoAndStop(17);
}
}
/*
fungsi hitEffect()
digunakan untuk memberi effect bergetar pada
jendela permainan
hal ini dilakukan dengan menggerakkan background
ke kanan dan ke kiri
serta menampilkan overlay layer berwarna kemerah-
merahan.
*/
function hitEffect() {
_root.player.gotoAndPlay(66);
if (movedRight) {
_root.bgStage1_coba._x -= 6;
movedRight = false;
clearInterval(moveLeft);
_root.hitEffectColor._visible = false;
} else if (movedRight == false && hitted) {
_root.bgStage1_coba._x += 6;
hitted = false;
movedRight = true;
_root.hitEffectColor._visible = true;
moveLeft = setInterval(hitEffect, 200);
}
}
/*
fungsi clears()
parameter objek bertipe string
digunakan untuk mengidentifikasi jenis objek yg
akan dihapus
(musuh-1, musuh-2, pemain, dsb)
```

```
proses hapus dilakukan dengan removeMovieClip
dan men-set array menjadi kosong
array = []
*/
function clears(objek:String) {
if (objek == "musuh1" ||
objek == "all") {
// clear enemy
//trace (musuh1Array.length);
for (i = 0; i < musuh1Array.length; i++) {
removeMovieClip(musuh1Array[i]);
}
musuh1Array = [];
}
if (objek == "enemyBullet" ||
objek == "all") {
// clear enemuBullet
for (i = 0; i < musuh1BulletArray.length; i++) {
removeMovieClip(musuh1BulletArray[i]);
}
musuh1BulletArray = [];
musuh1BulletSpeedX = [];
musuh1BulletSpeedY = [];
for (i = 0; i < musuh2BulletArray.length; i++) {
removeMovieClip(musuh2BulletArray[i]);
}
musuh2BulletArray = [];
musuh2BulletSpeedX = [];
musuh2BulletSpeedY = [];
for (i = 0; i < bossBullets.length; i++) {
removeMovieClip(bossBullets[i]);
}
bossBullet = [];
bossBulletSpeedX = [];
bossBulletSpeedY = [];
}
if (objek == "playerBullet" ||
objek == "all") {
// clear playerBullet
for (i = 0; i < bulletArray.length; i++) {
removeMovieClip(bulletArray[i]);
}
bulletArray = [];
}
if (objek == "musuh2" ||
objek == "all") {
for (i = 0; i < musuh2Array.length; i++) {
removeMovieClip(musuh2Array[i]);
}
musuh2Array = [];
}
if (objek == "all") {
for (i = 0; i < bonus.length; i++) {
removeMovieClip(bonus[i]);
}
}
bonus = [];
bonusHPFrame = [];
bonusBulletInFrame = [];
bonusType = [];
}

/*
fungsi checkPaused()
digunakan untuk memeriksa apakah pemain menekan
tombol pause (escape)
selama permainan berlangsung.
*/
var pauseTimer = 0;
function checkPaused() {
if (Key.isDown(Key.ESCAPE)) {
if (pauseTimer > 4) {
if (gamePaused) {
gamePaused = false;
stageBGM.start(stageBGM.position/ 1000);
} else {
gamePaused = true;
}
pauseTimer = 0;
}
} else {
pauseTimer++;
}
}
/*fungsi scrollBackground()
digunakan untuk melakukan perulangan scrolling
background permainan
jika perulangan telah mencapai maksimum, maka
boss akan bersiap untuk muncul
dengan men-set variabel seeTheBoss menjadi true
*/
function scrollBackground() {
bgStage1_coba._y += _root.bgStage1_speed;
if (bgStage1_coba._y > 3280) {
bgStage1_coba._y = 1575;
_root.currentRepeat++;
//trace ( _root.currentRepeat );
if (_root.currentRepeat == _root.finishStage1) {
_root.seeTheBoss = true;
trace ("boss is coming !");
}
}
if (_root.gamePaused) {
bgStage1_coba.stop();
} else {
bgStage1_coba.play();
}
}
/*
event onEnterFrame
dijalankan setiap saat ketika berada di frame ini
```

```
di dalamnya memanggil fungsi untuk logika utama
permainan,
misalnya menggerakkan pemain, musuh, peluru, dsb
*/this.onEnterFrame = function () {
// selama belum menang di stage ini, maka
if ( !win ) {
// cek apakah pemain menekan tombol pause
checkPaused();
// jika status game TIDAK sedang di-pause, maka
lanjutkan permainan
if (!gamePaused) {
// sembunyikan background layar pause
_root.pauseBG._visible = false;
// jika BGM stage 1 selesai dimainkan, maka ulangi
lagi
// dari awal (endless looping)
stageBGM.onSoundComplete = function() {
stageBGM.start(0, 0);
}
// panggil fungsi scrollBackground
scrollBackground();
// jika boss belum muncul, maka
if ( !seeTheBoss ) {
// gerakkan pemain dan peluru pemain
movePlayer();
movePlayerBullet();
// gerakkan musuh-1 dan pelurunya
moveEnemy1("new");
moveEnemy1Bullet();
// gerakkan musuh-2 dan pelurunya
moveEnemy2("new");
moveEnemy2Bullet();
// jika ada bonus, gerakkan bonus2 tersebut
movesBonus();
// jika boss siap untuk muncul
} else if ( seeTheBoss ) {
// selama boss belum ready, maka
if ( !bossCreated ) {
// panggil fungsi createTheBoss
createTheBoss();
// gerakkan musuh-1 dan 2 serta pelurunya
moveEnemy1("seeTheBoss");
moveEnemy2("seeTheBoss");
moveEnemy1Bullet();
moveEnemy2Bullet();
// gerakkan pemain dan peluru pemain
movePlayer();
movePlayerBullet();
// gerakkan bonus jika ada
movesBonus();
// jika boss sudah ready, maka
} else {
// gerakkan pemain dan peluru pemain
movePlayer();
// gerakkan bonus jika ada
movesBonus()
movePlayerBullet();
// jika menang, maka
if ( win ) {
// bersihkan semua object (pemain, musuh, peluru,
dsb)
clears("all");
// tampilkan finish stage 1
_root.stage1Boss.gotoAndPlay(8);
// jika belum menang, maka
} else {
// gerakkan musuh-1 dan 2 beserta pelurunya
moveEnemy1("seeTheBoss");
moveEnemy2("seeTheBoss");
moveEnemy1Bullet();
moveEnemy2Bullet();
// gerakkan boss beserta pelurunya
moveTheBoss();
moveBossBullet();
// tampilkan background pause
_root.pauseBG._visible = true;
// stop BGM stage 1
stageBGM.stop();
}
}
}
```

5.Skrip Game over

```
// GAME OVER !
// set teks skor sesuai skor yang didapat
this.txtFinalSkor.text = "Skor kamu : "+_root.skor;
stop();
this.onEnterFrame = function() {}
```

6.Skrip Finish

```
// FINISH !
// set teks skor sesuai skor yang didapat
_root.txtFinalSkor.text = "Skor kamu : "+_root.skor;
stop();
this.onEnterFrame = function() {}
```