RANCANG BANGUN APLIKASI *GAME* SERANGAN ALIEN PADA TELEPON SELULAR DENGAN MENGGUNAKAN J2ME

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

Oleh :

Nama : **ENDRA PERWITA SURYA**
No. Mahasiswa : **02523143**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

Saya yang bertanda tangan di bawah ini,

Nama : **ENDRA PERWITA SURYA**

No. Mahasiswa : **02523143**

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, Oktober 2007

(ENDRA PERWITA SURYA)

LEMBAR PENGESAHAN PEMBIMBING

Rancang Bangun Aplikasi *Game* Serangan Alien pada Telepon Selular dengan Menggunakan J2ME

TUGAS AKHIR

Disusun Oleh:

Nama : **ENDRA PERWITA SURYA**

No Mahasiswa : **02523143**

Yogyakarta, 24 Oktober 2007

Telah Diterima dan Disetujui dengan baik oleh:

Dosen Pembimbing

(Taufiq Hidayat, ST, MCS)

LEMBAR PENGESAHAN PENGUJI

Rancang Bangun Aplikasi *Game* Serangan Alien pada Telepon Selular dengan Menggunakan J2ME

TUGAS AKHIR

Disusun Oleh:

Nama : **ENDRA PERWITA SURYA**

No Mahasiswa : **02523143**

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 28 November 2007

Tim penguji

Taufiq Hidayat, ST, MCS.
Ketua

Syarif Hidayat, S.Kom
Anggota I

Lizda Iswari, ST
Anggota II

Mengetahui
Ketua Program Studi Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si., M.Kom

Persembahan

Puji syukur Alhamdulillah kehadirat Allah SWT atas segala nikmat dan karunia-Nya

Ibu Any Risprapti dan Bapak Soejoto (alm.) yang sangat saya cintai
terima kasih atas segalanya, do'a, didikan serta kasih sayang yang tak terhingga.
semoga pahala yang berlipat yang mereka dapatkan, karena ketidaksanggupan saya untuk membalasnya....

Mas Andi Surya Nugraha yang kusayang terima kasih atas do'a dan dorongannya,

Seseorang nun jauh di sana yang selalu saya sayangi
terima kasih atas segala do'a dan dorongannya serta semoga apa yang kamu harapkan dapat terwujud....

Sohibku Rony, Jatmiko, Drajat, Agus, Nurs, Aan, Wahid, Hru,
teman-teman Gali Kentungan 05B, anak-anak Boloterz, VoIP'02, xP6 Smunsa Sragentina,
Pasoepati, Slemania, Milanisti, Liverpudlian, dan teman-teman yang tidak bisa
saya sebutkan satu persatu terima kasih atas segala bantuan, do'a, dan dorongannya.

Motto :

" Sesungguhnya sesudah kesulitan itu ada kemudahan; Maka apabila kamu telah selesai (dari suatu urusan), kerjakanlah dengan sungguh-sungguh (urusan) yang lain ".

(Q.S. Alam Nasyrah ayat 6 dan 7)

" Jadilah sabar dan sholat sebagai penolongmu, sesungguhnya Allah beserta orang-orang yang sabar ".

(Q.S. Al Baqarah ayat 153)

" Dunia hanya berjalan tiga hari, yaitu : Kemarin, yang kita tidak berpengharapan apa-apa lagi darinya. Hari ini, yang harus kita peroleh kebaikan dan kesuksesannya. Dan esok hari, yang tidak kita ketahui apakah kita termasuk yang masih hidup atau yang tergolong sudah meninggal ".

(Al Hasan Al Bashri)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Dengan mengucapkan Alhamdulillah, puji dan syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga penyusun dapat menyelesaikan tugas akhir dengan judul ***"Rancang Bangun Aplikasi Game Serangan Alien pada Telepon Selular dengan Menggunakan J2ME"***.

Laporan Tugas Akhir ini saya buat sebagai syarat untuk mendapatkan gelar sarjana di jurusan Teknik Informatika Universitas Islam Indonesia.

Untuk itu saya ingin mohon maaf atas segala kekurangan yang terdapat dalam Program dan Laporan Tugas Akhir ini, baik yang disengaja ataupun yang tidak disengaja.

Ucapan terima kasih yang sebesar-besarnya kepada sumber-sumber dan referensi yang menjadi inspirasi. Dan khususnya kepada pihak-pihak yang telah membantu dalam proses penyelesaian Tugas Akhir ini, diantaranya :

1. Allah SWT yang memberikan hidayah-Nya serta semua karunia hingga saya dapat menyelesaikan Tugas Akhir ini dengan sebaik-baiknya.
2. Ibu Any Risprapti dan Mas Andi Surya Nugraha yang tak henti-hentinya memberikan doa dan dukungan.
3. Yang terhormat Bapak Drs. Imam Djati Widodo M., Eng Sc selaku Pembantu Dekan I Fakultas Teknologi Industri Universitas Islam Indonesia, yang telah membantu dalam permohonan penyelenggaraan Tugas Akhir ini.
4. Ketua jurusan Teknik Informatika FTI UII Bapak Yudi Prayudi S.Si., M.Kom. yang membantu sebagai pihak yang juga megesahkan Laporan Tugas Akhir.
5. Bapak Taufiq Hidayat ST, MCS, sebagai dosen pembimbing untuk meyelesaikan Tugas Akhir ini hingga dapat selesai dengan sebaik-baiknya dan sesuai dengan waktu yang telah ditentukan.

6. Seluruh karyawan dan staf bagian pengajaran khususnya bagian KP/TA yang telah membantu kami dalam perijinan dan surat menyurat.
7. Terima kasih juga kepada Nurs, teman-teman G-Six, Gali Kentungan 05B, Boloterz, VoIP'02, xP6 Smunsa Sragentina, dan teman-teman yang tidak dapat saya sebutkan satu persatu, yang telah membantu memberikan dukungan dan ilmu sehingga terselesaikannya Tugas Akhir ini.

Wassalamu'alaikum Wr. Wb

Yogyakarta , 5 Oktober 2007

Hormat Saya

(Penulis)

SARI

Makin banyaknya telepon selular yang mendukung Java di pasar membuka peluang bisnis pembuatan *game* dan aplikasi berbasis Java. Tren menunjukkan telepon selular sekarang dipakai pula untuk sarana hiburan. Bahasa pemrograman Java telah dikembangkan untuk perangkat telepon selular dengan memperkenalkan *Java 2 Micro Edition* (J2ME) yang menyediakan lingkungan yang lengkap untuk membuat suatu aplikasi Java.

Aplikasi *game* Serangan Alien merupakan salah satu aplikasi yang digunakan sebagai sarana hiburan yang dapat di akses melalui ponsel. Aplikasi *game* Serangan Alien dibangun dengan menggunakan bahasa pemrograman J2ME. Aplikasi *game* Serangan Alien terdiri dari beberapa menu yaitu: mulai, bantuan, misi, option, dan keluar dari aplikasi. Aktor *game* terdiri dari pemain (mobil) dan musuh (alien, monster, UFO). Aplikasi *game* ini mempunyai kemampuan untuk menyimpan nilai pemain dan menghentikan aplikasi sementara. Aplikasi *game* Serangan Alien yang dibuat dapat berjalan dengan baik pada perangkat *emulator* maupun perangkat telepon selular Sony Ericsson W810i. Dengan aplikasi *game* Serangan Alien, orang bisa mendapatkan hiburan kapanpun dan dimanapun dengan menggunakan ponsel yang mendukung Java.

Kata kunci: ponsel, *game*, J2ME

TAKARIR

class	kelas
configurations	konfigurasi
connected	terhubung
database	basis data
download	transfer file
embedded	ditanam
emulator	benda pengganti yang sebenarnya
free disk space	ruang kosong pada disk
game	permainan
game over	permainan berakhir
graphical user interface	antarmuka grafis untuk pengguna
high-level view	pandangan level tinggi
information hiding	penyembunyian informasi
intangiable	tidak dapat diraba
items	barang
java	bahasa pemrograman Java
java virtual machine	mesin maya java
limited	terbatas
mobile	bergerak
multiplatform	banyak platform
multiplayer	banyak pemain
path	jejak
paused	dihentikan sementara
play	bermain
private	pribadi
profile	profil
record management system	sistem manajemen penyimpanan
record store	tempat penyimpanan

runtime environment	lingkungan waktu jalan
screen	layar
strategy	strategi
tangiable	dapat diraba
three-dimensional	tiga dimensi
two-dimensional	dua dimensi
user-friendly	mudah
virtual machine	mesin virtual
with rules	dengan aturan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

teknol
denga
1.2 R

ini m
Alien
telah
1.3 B

1.4 T

meml
meng
1.5 M

manf
wakt
1.6 M
1.6.1

BAB I
PENDAHULUAN

1.1 Latar Belakang

Makin banyaknya telepon selular yang mendukung Java di pasar membuka peluang bisnis pembuatan *game* dan aplikasi berbasis Java. Apalagi, ternyata respon dan minat para pengguna telepon selular terhadap *game* dan aplikasi tersebut cukup tinggi dan cenderung meningkat dari waktu ke waktu. Tren menunjukkan telepon selular sekarang dipakai pula untuk sarana hiburan. Karenanya sebagai langkah awal perusahaan telepon selular berkonsentrasi membangun *game* telepon selular berbasis Java yang tidak kalah hebat dengan *game* komputer. Salah satu kelebihan utama aplikasi Java adalah sifatnya yang *multiplatform*. Dampaknya semua program yang dibuat dengan basis Java dapat digunakan di berbagai merk telepon selular. Pemakai juga dapat menambahkan *game* sendiri di telepon selular.

Pada beberapa negara di Asia seperti Jepang dan Korea Selatan, industri pembuatan aplikasi *game* ini telah berkembang dengan pesat. Sejumlah teknologi yang berbeda telah digunakan untuk pengembangan aplikasi *game* pada telepon selular, antara lain: *embedded games*, *sms games* dan *browsing games* (http://www.forum.nokia.com). Perkembangan ini juga diikuti oleh perkembangan bahasa pemrograman untuk pengembangan aplikasi *game* pada perangkat telepon selular. Bahasa pemrograman Java telah dikembangkan untuk perangkat telepon selular dengan memperkenalkan *Java 2 Micro Edition* (J2ME) yang menyediakan lingkungan yang lengkap untuk membuat suatu aplikasi Java.

Saat ini telepon selular generasi terbaru memiliki kemampuan mengaplikasikan program J2ME, yaitu program Java yang khusus dirancang untuk peranti bergerak. Aplikasi Java memungkinkan telepon selular memenuhi kebutuhan penggunanya melalui berbagai program menarik. *J2ME* merupakan bagian dari bahasa pemrograman Java yang ditujukan untuk perangkat kecil, seperti telepon selular dan *Personal Digital Assistants* (http://www.forum.nokia.com). Perangkat telepon selular yang telah mendukung

dengan bidang permasalahan yang dihadapi.

1.6.2 Metode Pengembangan Sistem

Pada penelitian ini, untuk mencapai hasil yang baik dalam merancang program, maka metodologi yang digunakan adalah:

a. Analisis Kebutuhan Perangkat Lunak

Melakukan analisis terhadap permasalahan yang ada untuk lebih mendapat gambaran yang luas mengenai sistem yang dibutuhkan dalam mengatasi permasalahan tersebut.

b. Perancangan Sistem

Tahap ini digunakan untuk merancang sistem yang akan dibangun..

c. Implementasi Sistem

Implementasi/realisasi rancangan dalam bentuk program

d. Pengujian Perangkat Lunak

Pada tahap ini digunakan untuk pengujian program tersebut.

1.7 Sistematika Penulisan

Penulisan tugas akhir ini disusun sedemikian rupa sehingga memberikan penjelasan yang sistematis dengan susunan sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi mengenai latar belakang masalah, rumusan masalah, batasan masalah, manfaat dan tujuan, metodologi penelitian, serta sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi landasan teori yang meliputi gambaran umum tentang teori yang diterapkan dalam pembuatan aplikasi ini. Dalam bab ini dijelaskan tentang konsep dasar *game*, sekilas pemrograman berorientasi objek, J2ME, pemrograman aplikasi *game* dengan J2ME.

BAB III METODOLOGI

Bab ini berisi mengenai langkah-langkah penyelesaian masalah yang meliputi:

Analisis Kebutuhan Sistem

Berisi mengenai analisis kebutuhan perangkat lunak yang meliputi spesifikasi sistem, diagram aktivitas, dan diagram kelas.

Perancangan Sistem

Berisi mengenai metode perancangan dan hasil perancangan sistem yang meliputi rancangan data, rancangan menu, rancangan antarmuka, dan rancangan gambar.

Implementasi Sistem

Berisi mengenai batasan implementasi, tahapan pembuatan program, dan implementasi sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi analisis kinerja perangkat lunak yang meliputi pengujian terhadap perangkat lunak dengan menggunakan perangkat simulasi.

BAB V SIMPULAN DAN SARAN

Berisi kesimpulan dari bab-bab sebelumnya, dan saran yang perlu diperhatikan berdasar keterbatasan yang ditemukan dan asumsi yang dibuat selama tugas akhir.

BAB II
LANDASAN TEORI

2.1 *Game*

Game adalah kegiatan yang dilakukan manusia untuk mendapatkan hiburan. *Game* sering kali terwujud atas dasar realita. Dari segi teknik, *game* adalah sistem tertutup yang menggambarkan kehidupan nyata. Disebut sistem karena *game* itu sendiri secara langsung adalah replikasi dari kehidupan nyata dan disebut tertutup karena peraturan-peraturan dan batasan-batasan yang terdapat di dalamnya. Bisa dibilang bermain *game* adalah cara yang aman untuk mencoba menjalani jenis kehidupan nyata yang lain. Dari menembaki monster-monster kejam, memimpin seribu pasukan militer, menjadi pembalap, menjadi walikota, sampai menjadi pilot pesawat. Pemain melihat suatu *game* sebagai perwujudan alam fantasi pribadinya. *Game* menggambarkan kenyataan secara subyektif, bukan obyektif. *Game* adalah tidak nyata secara obyektif, namun mereka nyata bagi pemain secara subyektif. Sesuatu yang nyata tersebut adalah fantasi si pemain, yang akan membentuk eksistensi dari pemain. Jadi, fantasi dan eksistensi pemain memiliki peran yang sangat penting dalam *game* apapun untuk membuat *game* tersebut menjadi nyata secara psikologi (Crawford, 1982).

2.1.1 Motivasi bermain *game*

Motivasi merupakan pendorong seseorang untuk bermain *game.* Ada dua komponen yang dibutuhkan untuk membuat *game* itu berjalan, permainan dan pemain (manusia). Pembuat *game* menemui keasyikan dalam pembuatan permainan itu sendiri, namun tujuan mereka pada akhirnya adalah mendidik, menghibur dan membimbing pemain.

Motivasi-motivasi tersebut adalah fantasi, *nose-thumbing*, pembuktian diri, pembauran sosial, latihan dan kebutuhan pengakuan (Crawford, 1982). Berikut ini penjelasan dari motivasi-motivasi tersebut :

1. Fantasi

   Motivasi yang sangat penting dalam bermain *game* adalah pemenuhan fantasi. *Game* bisa menciptakan sebuah fantasi sehingga pemain dapat melupakan masalahnya.

2. *Nose-thumbing*

   Sebuah fungsi yang umum bagi *game* adalah untuk menyediakan sarana untuk menanggulangi pembatasan sosial, setidaknya dalam khayalan. *Game* seringkali menempatkan pemainnya sebagai sesuatu yang tidak bisa diterima dalam kehidupan nyata, misalnya sebagai seorang pencuri.

3. Pembuktian diri

   Game berfungsi sebagai alat untuk mendemonstrasikan keberanian. Semua *game* mendukung motivasi untuk menjadi sesuatu yang lebih hebat dengan cara memberikan nilai dan menampilkan pemain dengan peringkat tertinggi.

4. Pembauran sosial

   Game seringkali dijadikan alat untuk berinteraksi sosial. *Game* itu sendiri merupakan hal yang *minor* bagi pemain karena hal yang lebih penting adalah sosialisasi yang akan dibangun.

5. Latihan

   Latihan dapat berarti secara mental atau fisik atau kombinasi dari keduanya. Dalam suatu kejuaraan, pemain seringkali melatih kemampuannya untuk mencapai suatu tingkatan tertentu sehingga pemain bisa berbuat lebih banyak dalam kejuaraan tersebut.

6. Kebutuhan pengakuan

   Semua orang membutuhkan pengakuan dan dikenal oleh orang lain. *Game* memungkinkan seseorang untuk diakui oleh orang lain.

2.1.2 Elemen-elemen *game*

Di setiap *game* terdapat beberapa elemen yang perlu diperhatikan, yaitu :

1. Interaksi

   Interaksi adalah proses timbal-balik antara lebih dari satu aktor. Aktor umum dalam hal ini adalah manusia dan *game*. Interaksi adalah elemen pertama kali yang akan ditemui pemain saat mulai bermain. Dimulai dari mengklik tombol *play*, menggerak-gerakan karakter dengan joypad/mouse/keyboard, menangkap *items* yang ditemukan, menangkis serangan lawan, sampai ke layar *game over*. Interaksi yang kurang nyaman akan langsung terasa oleh pemain. Grafik dan audio adalah faktor yang dijadikan media interaksinya. Pastikan agar media ini dan interaksinya selalu masuk akal dalam perpaduannya.

2. Cerita

   Cerita merupakan elemen yang sangat penting dalam *game*. Cerita yang sering kali ditemui kemiripannya dengan *game* lainnya akan menimbulkan kebosanan, dan sekali lagi pemain tidak sungkan untuk segera mencoba *game* selanjutnya. Cerita akan lebih menarik bila tetap sepadan dengan tema utama *game* (serius, kartun, komedi, drama, horor, dll). Melengkapi cerita dengan unsur yang lain boleh dilakukan asal masih bisa diterima sehingga tidak menyimpang dari tema utamanya.

3. Konflik

   Konflik berarti pertentangan yang terjadi dalam sebuah *game*. Konflik ini bisa membuat sebuah *game* menjadi lebih menantang dan lebih membuat pemain semakin penasaran. Hampir di setiap *game* yang kita temui sampai sekarang memiliki konflik. Konflik bisa dibuat secara langsung dan tidak langsung, bisa berupa kekerasan maupun bukan. Contoh dari konflik adalah: monster, misteri, teka-teki, peta lokasi, dan lain-lain.

4. Pilihan

   Dengan menawarkan pilihan-pilihan, sebuah *game* memberikan kebebasan kepada pemain untuk mengambil jalan dalam menyelesaikan konflik, dengan cara si pemain. Dalam *game*, pilihan-

pilihan akan membuat cabang-cabang cerita sehingga *game* tersebut bisa dimainkan berulang kali.

5. Bonus

   Bonus akan mengiming-imingi pemain untuk mendapatkanya, kalau bisa sebanyak mungkin. Maka bonus adalah termasuk elemen yang membuat pemain merasa tertantang dan termasuk elemen yang bisa dipakai untuk mengukur kehebatan pemain.

2.1.3 Komponen dan aturan

Komponen-komponen dan aturan-aturan merupakan penyusun sebuah *game* (Kramer, 2000). Komponen adalah perangkat keras dan aturan adalah perangkat lunak. Kedua hal tersebut yang mendefinisikan sebuah *game*.

2.1.4 Kriteria–kriteria *game*

Game berkaitan dengan rasa. Nilai yang dirasakan dari sebuah *game* tergantung dari pilihan-pilihan individu yang memainkannya. Sebuah *game* dapat disebut dengan *game with rules* jika memenuhi kriteria-kriteria sebagai berikut (Kramer, 2000), yaitu :

1. Aturan-aturan permainan

   Aturan-aturan merupakan inti dari sebuah *game*. Aturan-aturan itu hanya ada dalam dunia *game* bukan pada dunia sesungguhnya.

2. Tujuan

   Setiap *game* mempunyai tujuan yang harus dicapai yaitu agar bisa memenangkan *game* itu.

3. Kesempatan

   Kesempatan merupakan peluang untuk memenangkan sebuah *game*. Kesempatan memberikan ketidakpastian dan ketidaktahuan sehingga membuat *game* menjadi menarik.

4. Kompetisi

   Kompetisi bisa terjadi antara pemain dan sistem maupun antara sesama pemain. Kompetisi membutuhkan sebuah sistem yang bisa digunakan sebagai pembanding.

2.1.5 Kategori-kategori *game*

Jenis-jenis *game* pada perangkat kecil seperti perangkat telepon selular adalah sebagai berikut (Neuenhofen, 2003) :

1. *Strategy games* , seperti *Mine Sweeper, Reversi* dan *Bejeweled.*
2. *Card games*, seperti *Solitaire, Black Jack* dan *Poker.*
3. *Two-dimensional (2D) action games*, seperti *Galaxian, PacMan, Defender* dan *Asteroids.*
4. *Three-dimensional (3D) action games*, seperti *Doom, Quake* dan T*ony Hawk Pro Skater.*

2.2 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek telah berkembang luas dan digunakan dalam aplikasi-aplikasi komersial, seperti SmallTalk, C++ dan Java. Pemrograman berorientasi objek merupakan metode yang digunakan untuk mendapat solusi dari suatu masalah melalui perspektif objek. Pemrograman berorientasi objek merupakan inti dari Java dan pada kenyataannya semua program Java adalah berorientasi objek (Schildt, 2002). Tujuan pemrograman berorientasi objek adalah untuk memecah program menjadi beberapa program yang lebih kecil sehingga menjadi lebih mudah dan lebih sederhana untuk ditangani dan dipelihara. Dalam pengertian luas, objek merupakan sesuatu benda, baik yang dapat diraba *(tangiable)* maupun yang tidak *(intangiable)* atau sesuatu yang dapat dibayangkan (Sinaga, 2005). Sebuah objek memiliki keadaan *(state)* dan perilaku *(behavior)* yang melekat pada objek tersebut. Keadaan *(state)* menyatakan kondisi yang ada pada objek tersebut pada suatu saat tertentu, yang dinyatakan dalam atribut. Sedangkan perilaku *(behavior)* merupakan sekumpulan aksi yang dapat dilakukan oleh objek. Dalam pemrograman, *state* direlasikan sebagai *variable* data dan perilaku sebagai fungsi *(methods)*.

Elemen-elemen dari data disebut variabel instans karena nilainya bisa berubah setiap waktu. Objek-objek dialokasikan dengan operator yang baru dengan menggunakan konstruktor objek. Konstruktor merupakan metode khusus yang mempunyai nama sama dengan kelasnya dan tidak ada nilai yang

dikembalikan. Gambar 2.1 berikut ini menunjukkan bagaimana *variable* dan *methods* membentuk sebuah objek.

Gambar 2.1 Sebuah objek

Kelas merupakan sebuah model, pola atau cetak biru yang digunakan untuk menciptakan sebuah objek (Sinaga, 2005). Kelas merupakan blok-blok bangunan dari sebuah aplikasi Java. Sebuah kelas dapat berisi metode (fungsi), variabel, kode inisialisasi ataupun kelas yang lainnya. Kelas merupakan *template* untuk sebuah objek dan objek merupakan instans dari sebuah kelas. Kelas dideklarasikan dengan menggunakan kata kunci *class*. Metode dan variabel diletakkan di dalam percabangan kelas. Berikut ini bentuk umum pendefinisian sebuah kelas.

Sebuah kelas

```
class namaKelas{
    variabel instans 1;
    variabel instans 2;
    //...
    variable instans N;

    namaMetode1(daftar-parameter){
    // badan metode
    }
    //...
    namaMetodeN(daftar-parameter){
    // badan metode
    }
}
```

Gambar 2.2 Pendefinisian kelas

Semua bahasa pemrograman berorientasi objek menyediakan mekanisme untuk mengimplementasikan model berorientasi objek. Mekanisme-mekanisme itu adalah pengkapsulan, pewarisan, dan polimorphisme (Schildt, 2002).

2.2.1 Pengkapsulan

Objek digambarkan sebagai "kotak hitam" yang tidak perlu tahu apa yang ada di dalamnya (Sinaga, 2005). Komunikasi dengan objek dilakukan melalui pengiriman pesan. Pengiriman pesan dilakukan dengan menggunakan antarmuka *(interface)*. Dengan demikian, suatu objek yang berhubungan dengan objek lain tidak perlu mengetahui apa yang ada di dalam objek lain tersebut. Penyediaan akses terhadap objek melalui pengiriman pesan sembari menyembunyikan hal-hal yang *private* disebut dengan penyembunyian informasi *(information hiding)* atau disebut juga dengan enkapsulasi. Pengkapsulan atau enkapsulasi adalah proses pemaketan data bersama dengan fungsi. Keuntungan utama dari pengkapsulan adalah penyembunyian detil implementasi terhadap objek lain. Hal ini memiliki arti bahwa bagian internal dari suatu objek memiliki visibilitas yang terbatas dibandingkan dengan bagian eksternalnya. Bagian eksternal ini sering disebut sebagai antar muka objek. Objek lain berkomunikasi dengan objek ini melalui penggunaan antar muka. Oleh karena itu bagian internal objek harus dilindungi dari akses eksternal objek yang lain.

2.2.2 Pewarisan

Pewarisan merupakan suatu sifat yang memungkinkan dibuat kelas baru dengan menggunakan kembali kelas yang sudah ada (Sinaga, 2005). Kata kunci *extends* dapat digunakan untuk membuat kelas baru yang diturunkan dari kelas yang sudah ada.

2.2.3 Polimorphisme

Polimorphisme memiliki arti "banyak rupa" yang merujuk pada suatu kemungkinan suatu pesan yang sama yang diberikan kepada objek-objek yang berbeda akan mendapatkan tanggapan yang tidak sama (Sinaga, 2005). Objek yang menerima pesan memiliki tanggung jawab untuk memberi tanggapan atas pesan yang diterimanya. Dalam pemrograman berorientasi objek, polimorphisme

memiliki arti, fungsi yang sama memiliki perilaku yang berbeda pada kelas-kelas yang berbeda.

2.3 *Java 2 Micro Edition*

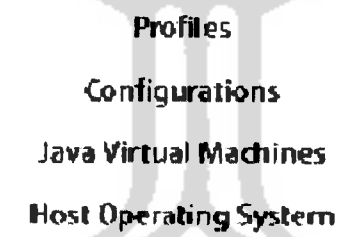
J2ME merupakan bagian dari bahasa Java yang ditujukan untuk perangkat kecil (http://www.forum.nokia.com). Hal ini menunjukkan bahwa peluncuran J2ME ditujukan untuk pengembangan aplikasi-aplikasi pada perangkat kecil.

2.3.1 *High level view*

Komponen-komponen *high-level view* terdiri dari :

1. Sejumlah *Java Virtual Machine* (JVM).
2. *Configurations dan Profiles.*
3. Alat-alat pengembangan dan konfigurasi perangkat.

Dua komponen pertama merupakan penyusun dari J2ME *runtime environment* (Mahmoud, 2002). Gambar 2.3 di bawah ini merupakan arsitektur dari *runtime environment.*



Gambar 2.3 Arsitektur J2ME *runtime environment*

JVM berada di atas sistem operasi yang digunakan oleh perangkat, kemudian terdapat *Configurations* yang terdiri dari sejumlah pustaka pemrograman yang menyediakan fungsi-fungsi dasar untuk mengakses kebutuhan sumber daya pada perangkat. Di tingkat paling atas terdapat *Profiles* yang menyediakan pustaka pemrograman tambahan.

2.3.2 *Profiles*

J2ME *profile* merupakan lapisan yang berada di atas *configurations*. J2ME *profile* membahas sesuatu yang spesifik untuk sebuah perangkat. J2ME *profile* terdiri dari beberapa buah *profile*, yaitu :

1. *Mobile Information Device Profile* (MIDP).
2. *Foundation Profile.*

3. *Personal Digital Assistance Profile (*PDA *Profile).*
4. *Personal Basis and Personal Profile.*
5. *Game Profile.*
6. RMI *Profile.*

2.3.3 *Configurations*

J2ME *configuration* mendefinisikan lingkungan kerja J2ME, *runtime* dan menyediakan *library* standar yang mengimplementasikan fitur standar sebuah perangkat genggam. Lingkungan kerja yang dimaksud meliputi JVM (*Java Virtual Machine*). J2ME *configuration* dikategorikan menjadi dua yaitu :

1. CDC *(Connected Device Configuration).*
2. CLDC *(Connected, Limited Device Configuration).*

2.3.3.1 *Connected Device Configuration*

Connected Device Configuration (CDC) ditujukan untuk perangkat yang terhubung pada sebuah jaringan, seperti tv *via internet* dan sistem navigasi mobil.

2.3.3.2 *Connected Limited Device Configuration*

Connected Limited Device Configuration (CLDC) merupakan konfigurasi yang ditujukan untuk perangkat *mobile*, seperti telepon selular dan PDA.

Berikut adalah perbandingan antara CDC dan CLDC (Shalahuddin dan Rosa, 2006) :

Tabel 2.1 Perbandingan antara CDC dan CLDC

CDC	CLDC
Mengimplemetasikan seluruh fitur J2SE.	Mengimplementasikan sebagian dari J2SE.
Java Virtual Machine yang digunakan adalah CVM *(Compact Virtual Machine).*	*Java Virtual Machine* yang digunakan adalah KVM *(Kilobyte Virtual Machine).*
Digunakan pada perangkat genggam *(internet* TV, *car* TV, *communicator)* dengan memori minimal 2MB.	Digunakan pada perangkat genggam *(handphone,* PDA, *two way pager)* dengan memori terbatas (160-512 KB)
Prosesor: 32 bit	Prosesor: 32 bit

2.3.4 *Kilobyte Virtual Machine*

Kilobyte Virtual Machine (KVM) merupakan implementasi dari *Java Virtual Machine* untuk spesifikasi CLDC. KVM ditujukan untuk perangkat kecil

dengan sumber daya terbatas sehingga bisa berjalan hanya dengan beberapa ratus *kilobyte* dari total memori (Fox, 2002).

2.3.5 MIDP

MIDP *(Mobile Information Device Profile)* merupakan sebuah versi dari platform Java yang didasarkan pada CLDC dan KVM yang ditujukan untuk *Mobile Information Devices (MIDs),* seperti telepon selular, pager dua arah dan PDA (Topley, 2002). Saat ini Sun telah mengeluarkan MIDP 2.0 yang memberikan dukungan multimedia. MIDP 2.0 digunakan dalam pengembangan aplikasi *game* ini.

Sebuah aplikasi yang berjalan dalam lingkungan MIDP disebut *MIDlet.* Satu atau lebih *MIDlet* yang dipaketkan secara bersama-sama disebut *MIDlet Suite.* MIDP akan membuat sebuah *aplication descriptor* atau berkas JAD *(Java Application Descriptor)* yang akan digunakan untuk mengeksekusi sebuah aplikasi. Berkas JAD akan mengenkapsulasi informasi dari isi berkas JAR.

Tabel 2.2 Atribut berkas JAD

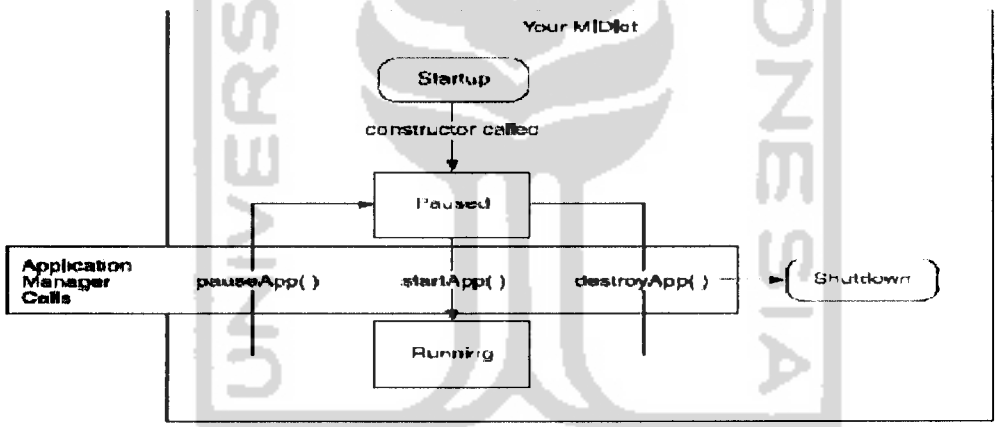
MIDlet- <Number>	<nama>, <ikon>, <kelas>
MIDlet-Jar-Size	Besarnya berkas *jar* dalam ukuran *byte*
MIDlet-Jar-URL	URL dimana lokasi JAR
MIDlet-Name	Nama *MIDlet suite*
MIDlet-Vendor	Pemilik/pengembang aplikasi
MIDlet-Version	Versi dari *MIDlet suite*
MicroEdition-Configuration	Versi *CLDC* yang digunakan
MicroEdition-Profile	Versi *MIDP* yang digunakan

MIDP 2.0 mendefinisikan kebutuhan-kebutuhan perangkat keras minimum yang harus dimiliki perangkat untuk mengimplementasikan *profile* ini, yaitu :

1. Resolusi *screen* minimal sebesar 96 x 54 piksel dengan 1 bit warna .
2. Masukan berupa *one-handed keyboard, two-handed keyboard, touchscreen,* atau *keypad.*
3. Memori non-volatil sebesar 128 *kilobyte* untuk komponen MIDP.
4. Memori non-volatil sebesar 8 *kilobyte* untuk aplikasi.
5. Memori volatil sebesar 32 *kilobyte* untuk *Java runtime.*
6. Jaringan tanpa kabel dua arah dengan *bandwidth* terbatas.
7. Mampu memainkan nada.

2.3.5.1 *MIDlet*

MIDlet adalah aplikasi yang ditulis untuk MIDP (Shalahuddin dan Rosa, 2006). Aplikasi *MIDlet* adalah bagian dari kelas *javax.microedition.midlet.MIDlet* yang didefinisikan pada MIDP. *MIDlet* berupa sebuah kelas abstrak yang merupakan sub kelas dari bentuk dasar aplikasi, sehingga antarmuka antara aplikasi J2ME dan aplikasi manajemen pada perangkat dapat terbentuk. Setiap perangkat J2ME mempunyai *application manager* yang sering disebut *Application Management System* (AMS) atau *MIDlet Management Software*. AMS berfungsi untuk mengontrol *MIDlet*. Metode-metode yang digunakan untuk mengontrol *MIDlet* disebut *Life Cycle Methods*. Gambar 2.4 merupakan gambar siklus hidup sebuah *MIDlet*.



Gambar 2.4 Metode *Life Cycle*

Keterangan-keterangan gambar *Life Cycle Methods* :

1. *public void startApp()*

   Metode ini mengindikasikan bahwa *MIDlet* menjauhi keadaan *paused* menuju keadaan *running*. Pada keadaan ini *MIDlet* menginisialisasi objek-objek yang dibutuhkan ketika dalam *running* dan mengeset tampilan sekarang.

2. *public void pauseApp()*

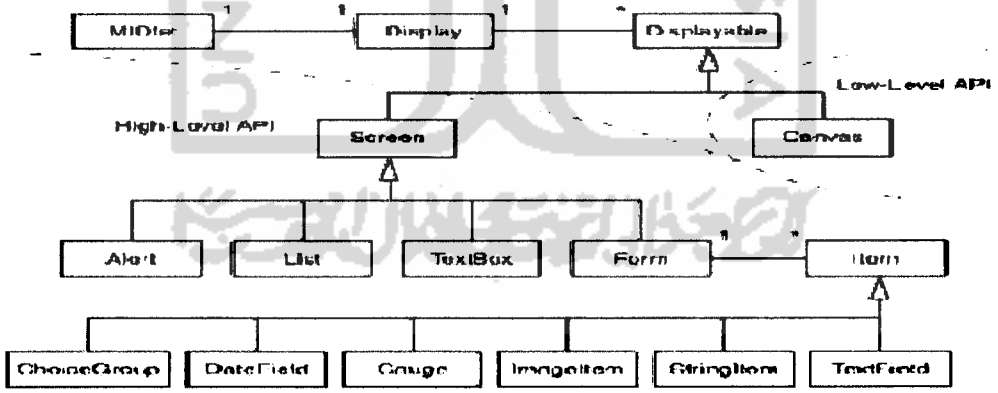
   Metode ini dipanggil ketika *MIDlet* menjauhi keadaan *running* menuju keadaan *paused*. Hal ini berarti *thread-thread* yang sedang berjalan akan menjadi *pause*.

3. *Public void destroyApp(boolean unconditional)*

   Metode ini mengindikasikan bahwa *MIDlet* bergerak menuju keadaan *shutdown*. Metode ini akan membebaskan atau menutup sumber daya yang dibutuhkan selama kehidupan *MIDlet*.

2.3.5.2 Antarmuka MIDP

MIDP menyediakan paket *javax.microedition.lcdui* yang terdiri dari kelas-kelas dan antarmuka untuk pembuatan komponen-komponen *Graphical User Interface* (GUI) pada aplikasi. MIDP mengijinkan penggunaan dua sistem antarmuka yang berbeda ketika mengembangkan sebuah aplikasi *game* yaitu *high level* API dan *low-level* API (Wells, 2004). *High level* API menyediakan unsur-unsur dasar seperti *list* dan *form* sehingga menjadi lebih ringan dan *portable,* tetapi hanya menawarkan kendali yang terbatas pada *screen* sehingga tidak mengijinkan kustomisasi warna, huruf maupun susunan komponen. *Low level* API menyediakan kendali lengkap pada *screen* dan tombol sehingga memungkinkan pembuatan antarmuka yang kompleks. Di bawah ini merupakan gambar MIDP GUI dari hirarki kelas *lcdui*.



Gambar 2.5 MIDP GUI

Ada 3 macam *screen* dalam *lcdui* tetapi hanya satu *screen* yang bisa ditampilkan pada satu dalam sebuah aplikasi. Ketiga macam *screen* tersebut adalah sebagai berikut :

1. *Complex components,* merupakan jenis *screen* yang mengenkapsulasi secara penuh komponen antarmuka yang lengkap.
2. *Form*, merupakan jenis *screen* yang menggunakan komponen *form*.

3. *Low-level* UI, merupakan jenis *screen* yang digunakan dengan konteks *low-level* yang dapat diakses melalui kelas *Canvas* atau kelas *Graphics*.

2.3.5.3 *Timer*

MIDP API mempunyai 2 buah kelas yang berhubungan dengan pengaturan waktu yaitu *java.util.Timer* dan *java.util.TimerTask*. *TimerTask* merupakan sebuah objek yang bisa dijadwalkan dengan *Timer* untuk mengeksekusi interval-interval secara regular (Barbagallo, 2004). *TimerTask* harus dilakukan secara cepat karena jika berjalan lambat maka akan menunda eksekusi layanan-layanan subsekuensial (Fox, 2002). Kelas *Timer* menyediakan kemampuan untuk mengeksekusi secara sekuensial satu atau lebih *TimerTask* dalam sebuah *Thread*. J2ME adalah *multithreaded*. Hal ini memungkinkan beberapa layanan berjalan secara simultan. *Thread* adalah sebuah *path* yang diambil oleh sebuah program pada waktu eksekusi (Suyoto, 2005). Dengan melakukan eksekusi melalui beberapa jejak maka aplikasi bisa berjalan lebih cepat dan lebih fleksibel.

2.3.5.4 *Record Management System*

Penyimpanan data pada MID dilakukan oleh RMS *(Record Management System)*. RMS menyimpan data sebagai rekaman-rekaman yang kemudian direferensikan dengan menggunakan kunci rekaman yang unik (Wells, 2004). Sekumpulan rekaman-rekaman yang disimpan disebut *Record Store*. *Record Store* terdiri berkas-berkas *biner* yang mempunyai sebuah nama unik dan panjangnya tidak boleh lebih dari 32 karakter dalam sebuah *MIDlet suite*. Berikut ini merupakan metode-metode yang terdapat pada kelas *RecordStore*, yaitu :

1. *public static RecordStore openRecordStore(String recordStoreName, boolean createIfNecessary)*

   Metode ini digunakan untuk membuka *Record Store*.
2. *public int addRecord(byte[] data, int offset, int numBytes)*

   Metode ini digunakan untuk menambahkan rekaman-rekaman.
3. *public byte[] getRecord(int recordId)*

Metode ini digunakan untuk mendapatkan kembali rekaman yang pernah disimpan.

4. *public int getNumRecords()*

   Metode ini digunakan untuk mengetahui berapa banyak jumlah rekaman yang terdapat dalam *RecordStore*.

5. *public RecordEnumeration enumerateRecords(RecordFilter filter, RecordComparator comparator, boolean keepUpdated)*

   RecordEnumeration merupakan sebuah antarmuka yang mengijinkan untuk melangkah melalui rekaman-rekaman *Record Store* tanpa harus mempunyai *recordId* tertentu.

6. *public byte[] nextRecord()*

   Metode ini akan mengembalikan rekaman berikutnya dalam urutan *RecordEnumerator.*

7. *public int previousRecordId()*

   Metode ini akan mengembalikan rekaman sebelumnya dalam urutan *RecordEnumerator.*

8. *public void setRecord(int recordId, byte[] newData, int offset, int numBytes)*

   Metode ini digunakan untuk memodifikasi rekaman.

9. *public static RecordStore openRecordStore(String recordStoreName, boolean createIfNecessary, int authmode, boolean writable)*

   Metode untuk saling berbagi *Record Store* di antara *MIDlet suite.*

2.4 Pemrograman Aplikasi *Game* dengan J2ME

Pemrograman aplikasi *game* untuk telepon selular dapat dilakukan dengan J2ME. Pada pengembangan aplikasi *game* dengan menggunakan J2ME terdapat beberapa hal yang harus diperhatikan (Lam, 2004), antara lain:

1. Memori

   Memori merupakan tempat dimana perangkat menyimpan data dan urutan-urutan langkah yang akan dilakukan. Model memori pada J2ME dapat dibagi menjadi tiga bagian :

   a. *Working/Heap Memory*

Mobile game dapat dikembangkan untuk aplikasi *game* pada jaringan dan jumlah pemain lebih dari satu.

5. Standar terbuka

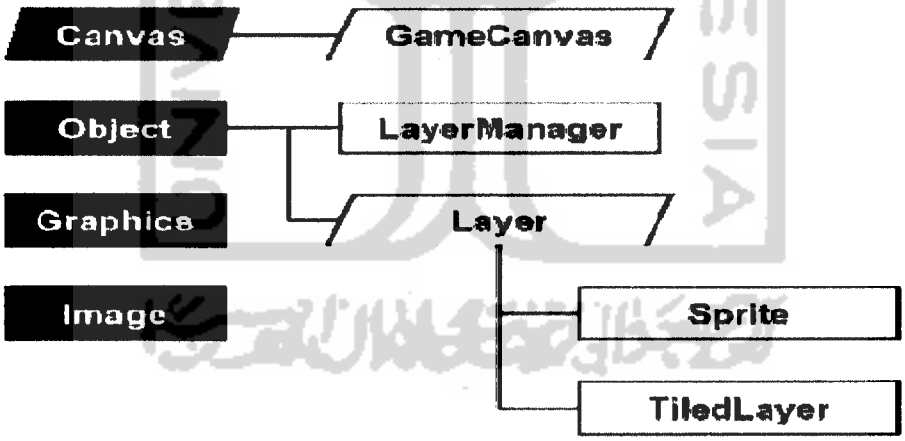
   Pengembangan *mobile game* dapat secara bebas tanpa harus membayar lisensi kepada pabrikan perangkat *mobile.*

6. *Deployment*

   Mobile game kebanyakan didapatkan dengan melakukan *download* dari jaringan kemudian dipasang pada perangkat *mobile.*

2.4.1 MIDP 2.0 *GameAPI*

MIDP 2.0 mempunyai *GameAPI* yang dapat digunakan dalam pengembangan aplikasi *game. GameAPI* tersebut terletak pada paket *javax.microedition.lcdui.game. GameAPI* terdiri dari *GameCanvas, Sprite, Layer, LayerManager dan TiledLaye*r. Hirarki kelas *GameAPI* dapat dilihat pada gambar di bawah ini (Day, 2004):



Gambar 2.6 Hirarki kelas *GameAPI*

2.4.1.1 Kelas *layer*

Layer merupakan kelas abstrak yang merepresentasikan elemen visual dari sebuah *game.* Semua gambar yang ditampilkan didasarkan pada *layer* (Barbagallo, 2004). *Layer* bisa digambar, dihilangkan maupun dipindahkan. Metode-metode yang terdapat pada kelas *Layer* adalah sebagai berikut :

1. *public void setPosition(int x, int y)*

   Metode ini digunakan untuk menempatkan posisi *layer* pada sudut kiri atas dari *screen.*

3. *public void Move(int dx, int dy)*

   Metode ini digunakan untuk memindahkan posisi *layer*.

4. *public void setVisible(Boolean visible)*

   Metode ini digunakan untuk menyembunyikan *layer*.

2.4.1.2 Kelas *sprite*

S*prite* merupakan *layer* animasi yang dapat menampilkan satu dari beberapa *frame* grafis atau dengan kata lain *sprite* adalah objek yang bergerak dalam sebuah *game* (Barbagallo, 2004). *Frame-frame* tersebut mempunyai ukuran yang sama dari sebuah objek gambar tunggal. Berikut ini merupakan metode dan konstruktor yang bisa digunakan pada kelas *Sprite* , antara lain :

1. *public Sprite(Image image)*

   Konstruktor ini digunakan untuk membuat *sprite* sederhana yang terdiri dari sebuah *frame* tunggal.

2. *public void setRefPixelPosition(int x, int y)*

   Metode ini mengijinkan untuk menunjuk titik koordinat pada perangkat.

3. *public void setTransform(int transform)*

   Metode ini digunakan untuk melakukan transformasi pada gambar. Transformasi yang disediakan adalah TRANS_NONE, TRANS_ROT90, TRANS_ROT180, TRANS_ROT270, TRANS_MIRROR, TRANS_MIRR OR_ROT90, TRANS_MIRRPR_180 dan TRANS_MIRROR_270.

4. *public Sprite(Image image, int frameWidth, int frameHeight)*

   Konstruktor ini dapat digunakan untuk membuat *sprite* animasi yang terdiri dari lebih dari satu *frame*.

5. *public void setFrame(int sequenceIndex)*

   Metode ini digunakan untuk menentukan *frame* urutan berapa yang akan ditampilkan.

6. *public void setFrameSequence(int [] sequence)*

   Metode ini digunakan untuk mengambil *array* dan menggunakan sebagai urutan *frame* dari *sprite*.

7. *public void nextFrame()*

   Metode ini digunakan mengeset *frame* sekarang sebagai *frame* berikutnya dalam urutan *frame*.

8. *public void prevFrame()*

   Berfungsi agar *frame* bergerak berbalik arah dari urutan *frame*.

9. *public boolean collidesWith(Image gbr, int x, int y, Boolean pixelLevel)*

   Metode ini digunakan untuk mendeteksi jika terdapat tabrakan antara *sprite* dengan gambar.

10. *public boolean collidesWith(Sprite s, Boolean pixelLevel)*

    Metode ini digunakan untuk mendeteksi jika sebuah *sprite* bertabrakan dengan *sprite* yang lainnya.

11. *public boolean collidesWith(TiledLayer t, Boolean pixelLevel)*

    Metode ini digunakan untuk mendeteksi jika sebuah *sprite* bertabrakan dengan *sprite* lainnya dalam sebuah *TiledLayer*.

12. *public void defineCollisionRectangle(int x, int y, int width, int height)*

    Metode ini digunakan untuk mendeteksi sebuah tabrakan secara manual. Metode ini menghasilkan tabrakan yang lebih akurat.

2.4.1.3 Kelas *tiledLayer*

TiledLayer merupakan sebuah *layer* yang terdiri dari *grid-grid* sel (Day, 2004). Masing-masing sel dapat menampilkan satu dari beberapa *tile* atau kotak dari sebuah objek gambar. Sel dapat juga berisi *tile-tile* animasi dimana data piksel dapat berubah dengan cepat. *Tile* dapat digunakan secara berulang-ulang sehingga menghasilkan gambar yang lengkap. Berikut ini merupakan metode dan konstruktor yang bisa digunakan pada kelas *TiledLayer* antara lain :

1. *public TiledLayer(int columns, int rows, Image image, int tileWidth, int tileHeight)*

   Konstruktor tunggal yang dapat digunakan untuk membuat *Tiledlayer*.

2. *public void setCell(int col, int row, int tileIndex)*

   Metode ini digunakan mengeset sel-sel secara manual.

3. *public int createAnimatedTile(int staticTileIndex)*

Metode ini digunakan untuk membuat *tile* animasi.

4. *public void setAnimatedTile(int animatedTileIndex, int staticTileIndex)*
   Metode ini mengambil nilai indeks tile animasi yang akan dimodifikasi dan indeks tile statis yang akan diasosiasikan dengan *tile* animasi.

2.4.1.4 Kelas *layerManager*

LayerManager merupakan kelas yang mengatur sekumpulan *layer*. Sebuah *screen* tunggal dapat terdiri dari banyak *layer* (Day, 2004). Berikut ini merupakan metode dan konstruktor yang bisa digunakan pada kelas *LayerManager*, antara lain :

1. *public LayerManager()*
   Konstruktor untuk memanggil *LayerManager.*
2. *public void append(Layer l)*
   Merupakan metode yang digunakan untuk menambahkan l*ayer-layer.*
3. *public void paint(Graphics g, int x, int y)*
   Metode ini digunakan untuk menggambar *layer-layer* yang akan ditempelkan pada *LayerManager.*
4. *public void insert(Layer l, int index)*
   Metode yang digunakan untuk menambah *layer* pada indeks tertentu.
5. *public void remove(Layer l)*
   Metode yang digunakan untuk menghapus *layer* dari *TiledLayer.*

2.4.1.5 Kelas *gameCanvas*

GameCanvas merupakan kelas yang menyediakan sebuah versi kelas *Canvas* dengan kemampuan-kemampuan ekstra untuk menangani masukan dan *rendering* (Wells, 2004). Berikut ini merupakan metode dan konstruktor yang bisa digunakan pada kelas *GameCanvas* , antara lain :

1. *protected GameCanvas(boolean suppressKeyEvents)*
   Konstruktor ini harus dimplementasikan pada setiap subkelas *GameCanvas* agar kode program bisa berjalan dengan baik.
2. *int getKeyStates()*

Metode ini digunakan untuk mengecek tombol masukan. Nilai *integer* merupakan pola *bit* yang menyatakan tombol-tombol dari *game*. Tabel 2.3 merupakan pola bit yang ada pada *GameCanvas*.

3. *protected Graphics getGraphics()*

   Metode ini akan mengembalikan kelas *Graphics* yang akan berasosiasi dengan *screen* itu sendiri.

4. *public void flushGraphics()*

   Metode untuk memasukkan dan menampilkan *buffer* kedua pada *screen.*

Tabel 2.3 Pola *bit GameCanvas*

Pola bit	Deskripsi
DOWN_PRESSED	*Bit* untuk tombol aksi *DOWN*
FIRE_PRESSED	*Bit* untuk tombol aksi *FIRE*
GAME_A_PRESSED	*Bit* untuk tombol aksi *GAME A*
GAME_B_PRESSED	*Bit* untuk tombol aksi *GAME B*
GAME_C_PRESSED	*Bit* untuk tombol aksi *GAME C*
GAME_D_PRESSED	*Bit* untuk tombol aksi *GAME D*
LEFT_PRESSED	*Bit* untuk tombol aksi *LEFT*
RIGHT_PRESSED	*Bit* untuk tombol aksi *RIGHT*
UP_PRESSED	*Bit* untuk tombol aksi *UP*

2.4.2 MIDP 2.0 *media* API

Pada MIDP 1.0 fungsi suara yang diberikan hanya suara *beep* sederhana yang dihasilkan dari kelas *alert* (Barbagallo, 2004). MIDP 2.0 *Media* API menawarkan fungsi *sound* yang lebih baik daripada MIDP 1.0. MIDP 2.0 *Media* API merupakan bagian dari *Mobile Media* API (MMA) yang menyediakan kelas-kelas yang dapat digunakan untuk aplikasi audio dan video (Day, 2004). Tabel di bawah ini menerangkan format multimedia yang didukung oleh MMA.

Tabel 2.4 Format multimedia MMA

Media	Format
Audio	PCM dan WAV *audio.*
MIDI	*MIDI* (jenis 0 dan jenis 1), SP-*MIDI*
Video	MPEG-1
Audio *Capture*	*Audio* dari *platform* Solaris, Windows dan Linux.

Media API terdiri dari tiga kelas utama yaitu *Manager, Player dan Control* (Barbagallo, 2004). *Manager* merupakan kelas sederhana yang digunakan untuk meminta objek *Player* untuk memainkan data audio. *Player*

bertanggungjawab untuk mengendalikan permainan audio. *Player* dapat digunakan untuk memainkan media objek seperti berkas *midi* dan *mp3*. *Control* merupakan antarmuka khusus yang digunakan untuk mengubah audio yang sedang dimainkan.

2.4.2.1 Kelas *manager*

Manager merupakan kelas sederhana yang digunakan untuk meminta objek *Player* untuk memainkan data audio (Barbagallo, 2004). Berikut ini merupakan metode-metode yang terdapat pada kelas *Manager*, yaitu :

1. *public static Player createPlayer(String locator)*
   Metode ini digunakan untuk membuat sebuah *Player.*
2. *public static String() getSupportedContentTypes(String protocol)*
   Metode ini dapat digunakan mengecek jenis-jenis media apa saja yang didukung oleh perangkat.
3. *public InputStream getResourceAsStream(String name)*
   Metode ini berada pada kelas abstrak primitif yang disebut *Class.* Dibutuhkan referensi dari objek *Class* dengan memanggil *getClass().*
4. *public static void playTone(int note, int duration, int volume)*
   Metode ini digunakan untuk memainkan sebuah nada.

2.4.2.2 Kelas *player*

Player bertanggungjawab untuk mengendalikan permainan audio (Barbagallo, 2004). Di bawah ini merupakan metode-metode yang dapat digunakan pada kelas *Player*, yaitu :

1. *public void start()*
   Merupakan metode yang dapat digunakan untuk memainkan musik.
2. *public int getState()*
   Metode ini digunakan untuk mengetahui *state* dari *Player.*
3. *public void setLoopCount(int count)*
   Metode yang dapat digunakan untuk mengadakan perulangan.
4. *public void addPlayerListener(PlayerListener playerListener)*
   Metode yang dapat digunakan untuk mengambil objek *PlayerListener.*

5. *public void playerUpdate(Player player, String event, Object eventData)*

   Metode yang digunakan untuk menyediakan *playerUpdate* yang akan digunakan oleh *playerListener*.

2.4.2.3 Kelas *control*

Control merupakan antarmuka khusus yang digunakan untuk mengubah audio yang sedang dimainkan (Barbagallo, 2004). Di bawah ini merupakan metode-metode yang dapat digunakan pada kelas *Control*, yaitu :

1. *public Control getControl(String controlType)*

   Metode ini mengembalikan objek Control.

2. *public int setLevel(int level)*

   Metode ini digunakan untuk mengeset tingkatan *volume* (0-100).

3. *public void setMute(boolean mute)*

   Metode ini digunakan untuk membisukan suara.

BAB III
METODOLOGI

3.1 Analisis

3.1.1 Deskripsi *game*

Aplikasi *game* Serangan Alien merupakan aplikasi *game* yang bercerita tentang sebuah mobil yang harus menghindari serangan alien dari angkasa. Pada tahun 2100, orang-orang dari bumi mulai melakukan penelitian ke planet lain. Seorang ilmuwan dari bumi yang bernama Mr. X melakukan penelitian ke planet antah-berantah dengan mobilnya. Tiba-tiba ia mendapat serangan dari angkasa yang dilakukan oleh alien bersama teman-temannya. Untungnya mobil Mr. X sudah dilengkapi dengan perlengkapan tempur. Dalam proses menyelamatkan diri, Mr. X harus menghindari serangan berupa tembakan dari alien bersama teman-temannya dari angkasa. Jadi misi *game* ini adalah menembaki sebanyak-banyaknya musuh dan menghindari serangan alien bersama teman-temannya agar dapat poin dan menang.

3.1.1.1 Aktor *game*

Aktor-aktor yang terlibat dalam *game* ini adalah mobil dan musuh. Musuh terdiri dari alien, UFO, dan monster. Posisi mobil ditentukan terlebih dahulu. Musuh merupakan sejenis alien bersama teman-temannya, dimana posisinya ditentukan secara acak

3.1.1.2 Aturan-aturan *game*

Sebuah mobil mempunyai misi untuk selamat dari serangan para alien bersama teman-temannya dari angkasa. Serangan dari angkasa tersebut dilakukan oleh alien, UFO, dan monster. Dalam menyelamatkan dirinya, mobil tersebut mempunyai tiga nyawa. Apabila tiga kali kalah maka misinya tidak berhasil dan *game* akan berakhir.

3.1.1.3 Komponen *game*

Komponen-komponen yang digunakan pada aplikasi *game* ini adalah sebuah perangkat *mobile* yang cukup kaya akan warna sehingga bisa menampilkan gambar-gambar *bitmap* dan telah mendukung MIDP 2.0.

3.2 Hasil Analisis

3.2.1 Analisis kebutuhan

Analisis kebutuhan untuk aplikasi *game* Serangan Alien adalah sebagai berikut:

1. Aplikasi harus mampu untuk mendeteksi tabrakan.
2. Aplikasi harus mampu mengurangi nyawa jika mobil berhasil ditembak musuh (alien, UFO, monster).
3. Aplikasi harus mampu menyimpan nilai pemain.
4. Aplikasi harus mampu membuat posisi musuh (alien, UFO, monster) menjadi acak.
5. Aplikasi harus mampu mendeteksi gerakan mobil.
6. Aplikasi harus menyediakan suara musik dan nada.
7. Aplikasi harus mampu membandingkan setiap nilai yang dihasilkan pemain *game.*

Berdasarkan hasil analisis kebutuhan di atas, maka didapatkan pemecahan masalah sebagai berikut :

1. Dibuat metode agar musuh (alien, UFO, monster) bisa mendeteksi keberadaan mobil.
2. Dibuat metode untuk menyimpan nilai.
3. Dibuat metode untuk membandingkan nilai.
4. Dibuat metode untuk menghentikan aplikasi sementara (*pause*).

3.2.2 Spesifikasi *Game*

Di bawah ini merupakan spesifikasi fungsional dari *game* Serangan Alien :

1. Aplikasi *game* bisa mendeteksi tabrakan.
2. Aplikasi *game* bisa menyimpan dan membandingkan nilai.
3. Musuh bisa mendeteksi keberadaan pemain.
4. Antarmuka yang terdapat pada aplikasi ini adalah layar *splash,* menu utama dan sub-sub menu, permainan, dan *game over.*
5. Untuk menyimpan data menggunakan *Record Management System.*
6. Aplikasi *game* ini memerlukan telepon selular yang sudah mendukung MIDP 2.0.

3.2.3 Analisis Kebutuhan *Software*

Software yang digunakan untuk membuat sistem adalah:

1. Sistem Operasi (*Operating System*)
   Windows 98, Windows 2000 atau Windows XP, dan Symbian merupakan sistem operasi yang dapat digunakan untuk mengoperasikan sistem ini.
2. EditPlus 2, UltraEdit-32, dan J2SDK 1.4.2_03 sebagai editor penulisan kode dan pengembangan aplikasi.
3. J2ME Wireless Toolkit sebagai pengujicobaan sistem dan *emulator.*
4. Adobe Photoshop 7.0, Microsoft Visio 2003, dan Enterprise Architect 6.5 untuk merancang desain *game.*

3.2.4 Analisis Kebutuhan *Hardware*

Untuk membuat sistem ini membutuhkan *hardware* dengan spesifikasi sebagai berikut :

a. *Processor* minimal berkecepatan 1,6 GHz
b. RAM minimal 256 MB
c. VGA minimal 64 MB
d. *Free disk space* minimal 150 MB
e. Monitor
f. Mouse
g. Keyboard
h. Ponsel yang memiliki fasilitas Java di dalamnya.

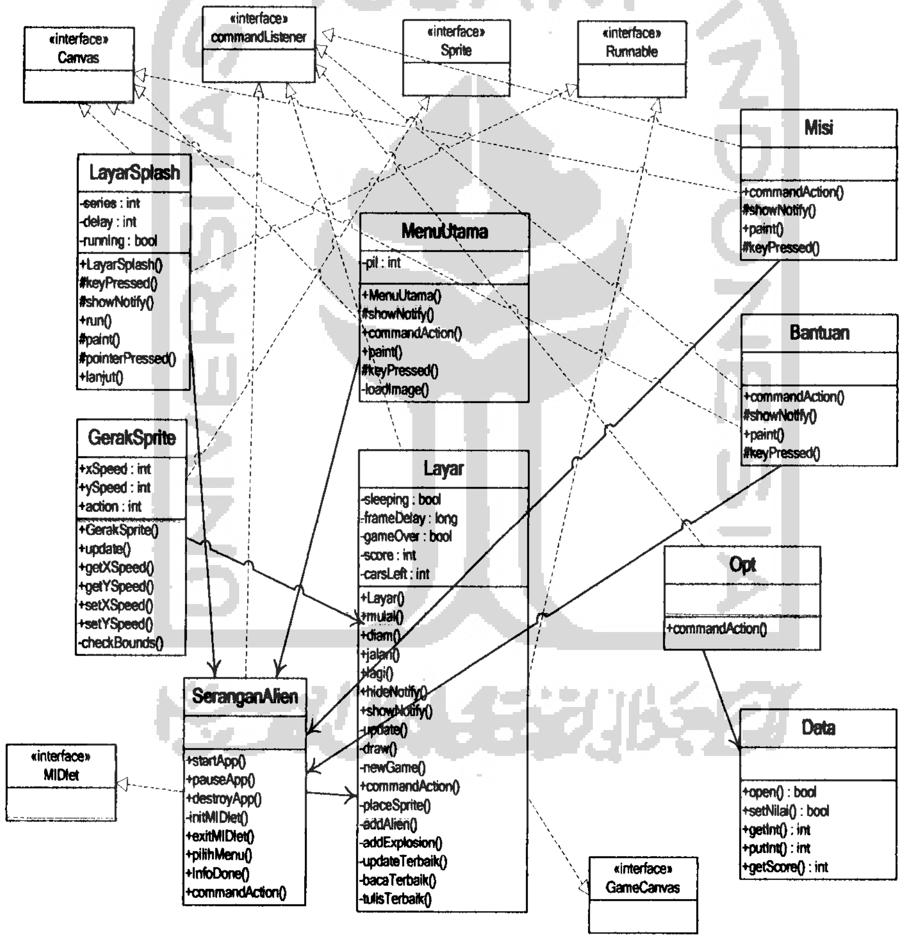
3.3 Perancangan

Metode perancangan perangkat lunak yang digunakan dalam pembuatan implementasi J2ME pada *game* Serangan Alien ini menggunakan *Unified Modelling Language* (UML). Model-model yang digunakan dalam perancangan ini adalah diagram kelas dan diagram aktivitas. Diagram kelas digunakan untuk mendokumentasikan aspek *static* dari sistem. Diagram aktivitas digunakan untuk mendokumentasikan aspek dinamis dari sistem.

3.3.1 *Unified Modelling Language* (UML)

3.3.1.1 Diagram Kelas

Diagram kelas merupakan diagram yang menunjukkan kelas-kelas yang terdapat pada aplikasi *game* Serangan Alien dan hubungan-hubungan yang ada pada kelas-kelas tersebut. Di bawah ini merupakan diagram kelas pada *game* Serangan Alien. Untuk lebih lengkapnya diagram kelas *game* Serangan Alien dapat dilihat dalam gambar 3.1.



Gambar 3.1 Diagram kelas Serangan Alien

Keterangan gambar 3.1 adalah sebagai berikut :

1. ————————→ : Relasi asosiasi
2. - - - - - - - - - - ▷ : Implementasi *interface*

Pada diagram kelas Serangan Alien di atas terdapat dua jenis kelas. Yang pertama yaitu kelas abstrak yang sudah ada dalam Java, yang terdiri dari: Canvas,

commandListener, Sprite, Runnable, MIDlet, dan GameCanvas. Sedangkan yang kedua yaitu kelas yang dibuat dalam *game*, terdiri dari: SeranganAlien, LayarSplash, MenuUtama, Layar, GerakSprite, Data, Opt, Bantuan, dan Misi. Relasi diantara kelas-kelas tersebut adalah sebagai berikut :

1. Kelas SeranganAlien, merupakan kelas yang melakukan *extends* terhadap kelas *MIDlet*. Kelas ini harus ada sebagai syarat aplikasi dapat diciptakan. Pada kelas ini akan ditampilkan menu-menu yang terdapat pada aplikasi dan metode untuk keluar dari aplikasi *MIDlet*. Kelas ini berasosiasi dan mengakses kelas MenuUtama serta membuat objek dari kelas Layar. Kelas ini juga akan mengimplementasikan *CommandListener*. Tabel 3.1 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas SeranganAlien.

Tabel 3.1 Metode/konstruktor Kelas SeranganAlien

Metode/Konstruktor	Keterangan
public SeranganAlien()	Konstruktor kelas SeranganAlien.
public void startApp()	Metode yang menyatakan bahwa *MIDlet* telah dibuat.
public void pauseApp()	Metode yang menyatakan bahwa *MIDlet* dihentikan.
public void destroyApp()	Metode yang menyatakan bahwa *MIDlet* telah dihancurkan.
private void initMIDlet()	Metode untuk menginisialisasi *MIDlet*.
public void exitMIDlet()	Metode untuk keluar dari *MIDlet*.
public void pilihMenu()	Metode untuk memilih menu.
public void infoDone()	Metode untuk kembali ke *MIDlet*.
public void commandAction()	Metode untuk memproses permintaan dari pengguna aplikasi.

2. Kelas LayarSplash, merupakan kelas yang akan digunakan untuk menampilkan gambar *splash*. Kelas ini berasosiasi dan memiliki akses dengan kelas SeranganAlien serta merupakan *extends* dari kelas abstrak *Canvas*. Kelas ini juga akan mengimplementasikan *Runnable*. Tabel 3.2 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas LayarSplash.

Tabel 3.2 Metode/konstruktor kelas LayarSplash

Metode/Konstruktor	Keterangan
public LayarSplash	Konstruktor kelas LayarSplash dan merupakan perluasan dari kelas *Canvas*.
protected void keyPressed()	Metode untuk mendeteksi penekanan tombol.
protected void showNotify()	Metode yang digunakan untuk menunjukkan notifikasi.
public void run()	Metode untuk mengakses event berikutnya dengan menggunakan metode *cekTime()*.
public void cekTime()	Metode untuk berganti ke layar berikutnya.
protected void paint()	Metode yang digunakan untuk menggambar layar *splash*.
protected void pointerPressed()	Metode untuk mendeteksi penekanan pointer.
private void lanjut()	Metode untuk berganti ke layar berikutnya.

3. Kelas MenuUtama, merupakan kelas yang berisi tentang objek untuk memilih menu permainan. Pada kelas ini diperlukan sedikit aksi terhadap tombol panah dan agar dapat memilih menu apa yang akan dijalankan. Efek ini akan menggeser gambar panah penunjuk, sekaligus memilih program. Kelas ini berasosiasi dan memiliki akses dengan kelas SeranganAlien serta merupakan *extends* dari kelas abstrak *Canvas*. Kelas ini juga akan mengimplementasikan *CommandListener*. Tabel 3.3 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas MenuUtama.

Tabel 3.3 Metode/konstruktor kelas MenuUtama

Metode/Konstruktor	Keterangan
public MenuUtama()	Konstruktor kelas MenuUtama.
protected void showNotify()	Metode yang digunakan untuk menunjukkan notifikasi.
public void commandAction()	Metode untuk memproses permintaan dari pengguna aplikasi.
protected void paint()	Metode yang digunakan untuk menampilkan menu utama pada *game*.
protected void keyPressed()	Metode untuk mendeteksi penekanan tombol.

4. Kelas Layar, merupakan kelas yang akan menampilkan mode *game play*. Kelas ini akan menggunakan metode-metode yang terdapat pada *super class*nya. Kelas GerakSprite dan kelas SeranganAlien merupakan *super class* dari kelas Layar. Kelas Layar akan mengimplementasikan *interface Runnable* dan merupakan *extends* kelas abstrak *GameCanvas*. Tabel 3.4 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas Layar.

Tabel 3.4 Metode/konstruktor kelas Layar

Metode/Konstruktor	Keterangan
public Layar()	Konstruktor kelas Layar.
public void mulai()	Metode untuk memulai pertempuran.
public void diam()	Metode untuk menghentikan pertempuran sementara *(pause)*.
public void jalan()	Metode untuk melanjutkan pertempuran.
public void stop()	Metode untuk menghentikan musik *player*.
public void start()	Metode untuk menjalankan musik *player*.
public void henti()	Metode untuk menghentikan *game* sementara.
public void run()	Metode untuk menjalankan *game*.
public void main()	Metode untuk melanjutkan *game* yang di-*pause*.
public void lagi()	Metode untuk main lagi dari awal.
public void hideNotify()	Metode yang digunakan untuk menyembunyikan notifikasi.
public void showNotify()	Metode yang digunakan untuk menunjukkan notifikasi.
public void update()	Metode untuk meng-*update game*.
public void draw()	Metode untuk melakukan *draw* grafis.
public void newGame()	Metode untuk memulai *game* baru.
public void commandAction()	Metode untuk memproses permintaan dari pengguna aplikasi.
private void placeSprite()	Metode untuk menempatkan *sprite*.
private void addAlien()	Metode untuk menampilkan alien.
private void addMissile()	Metode untuk menampilkan peluru.
private void addExplosion()	Metode untuk menampilkan ledakan.
private void updateTerbaik()	Metode untuk meng-*update* nilai terbaik.
private void bacaTerbaik()	Metode untuk membaca nilai terbaik.
private void tulisTerbaik()	Metode untuk menulis nilai terbaik.

5. Kelas GerakSprite, merupakan kelas yang digunakan untuk membuat gerakan *sprite*. Kelas ini merupakan *extends* dari kelas abstrak *Sprite*. Kelas ini berasosiasi dengan kelas Layar dan memiliki metode-metode yang nantinya dapat di-*overloading* oleh kelas Layar dalam menentukan gerakan *sprite*. Tabel 3.5 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas GerakSprite.

Tabel 3.5 Metode/konstruktor kelas GerakSprite

Metode/Konstruktor	Keterangan
public GerakSprite()	Konstruktor kelas GerakSprite.
public void update()	Metode untuk meng-*update* perubahan data.
public int getXSpeed()	Metode untuk mendapatkan kecepatan secara horizontal.
public int getYSpeed()	Metode untuk mendapatkan kecepatan secara vertikal.
public void setXSpeed()	Metode untuk menambah kecepatan secara horizontal.
public void setYSpeed()	Metode untuk menambah kecepatan secara vertikal.
public void checkBounds()	Metode untuk menentukan batas layar.

6. Kelas Data, merupakan kelas yang berisi tentang objek penyimpanan data nilai dengan menggunakan RMS. Kelas ini memiliki fungsi untuk mengganti pilihan *option*. Kelas ini berasosiasi dan memiliki akses ke kelas Opt. Tabel 3.6 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas Data.

Tabel 3.6 Metode/konstruktor kelas Data

Metode/Konstruktor	Keterangan
public Data()	Konstruktor kelas Data.
public void putInt()	Metode untuk mengubah nilai integer menjadi nilai *array byte*.
public void loadSett()	Metode untuk me-*load setting game*.
public void updateSett()	Metode untuk meng-*update setting* musik dan getar.
protected void loadData()	Metode untuk menampilkan data.
protected void dataAwal()	Metode untuk menyimpan data awal.
protected void updateData()	Metode untuk meng-*update* data.

7. Kelas Opt, merupakan kelas yang berisi pilihan pengubahan pada aplikasi *game*. Pilihan pengubahannya yaitu menghilangkan musik getar. Kelas

ini berasosiasi dengan kelas Data. Kelas ini juga akan mengimplementasikan *CommandListener*. Tabel 3.7 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas Opt.

Tabel 3.7 Metode/konstruktor kelas Opt

Metode/Konstruktor	Keterangan
public Opt()	Konstruktor kelas Opt.
public void commandAction()	Metode untuk memproses permintaan dari pengguna aplikasi.

8. Kelas Bantuan, merupakan kelas yang berisi panduan tentang bagaimana cara bermain *game*. Kelas ini hampir sama dengan kelas Misi yang penjelasannya berupa teks. Kelas ini memiliki akses dengan kelas SeranganAlien serta merupakan *extends* dari kelas abstrak *Canvas*. Kelas ini juga akan mengimplementasikan *CommandListener*. Tabel 3.8 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas Bantuan.

Tabel 3.8 Metode/konstruktor kelas Bantuan

Metode/Konstruktor	Keterangan
public Bantuan()	Konstruktor kelas Bantuan.
protected void showNotify()	Metode yang digunakan untuk menunjukkan notifikasi.
public void paint()	Metode untuk menampilkan teks bantuan.
public void commandAction()	Metode untuk memproses permintaan dari pengguna aplikasi.
protected void keyPressed()	Metode untuk mendeteksi penekanan tombol.

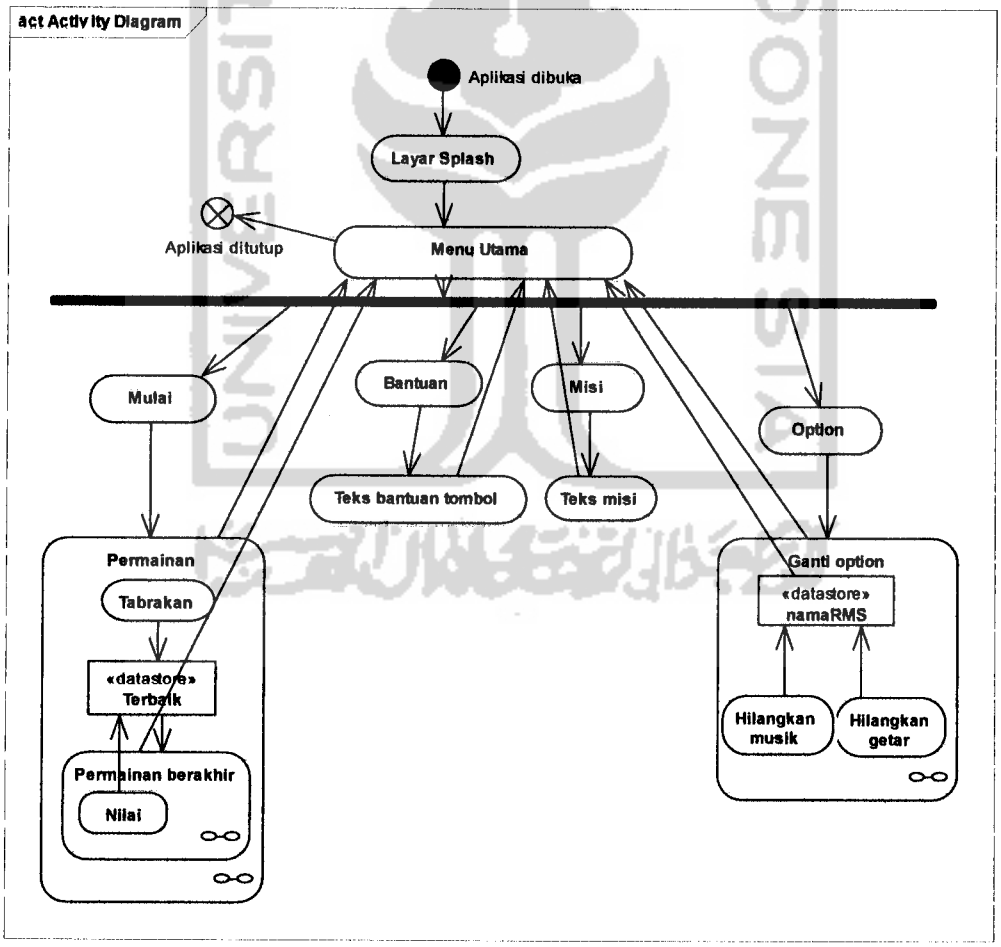
9. Kelas Misi, merupakan kelas yang digunakan untuk menampilkan teks misi *game*. Karena penjelasannya berupa teks, maka diperlukan fungsi yang berkaitan dengan penulisan teks *string*. Sebelumnya teks yang akan ditampilkan telah dideklarasikan sebagai variable bertipe *String array*. Kelas ini memiliki akses dengan kelas SeranganAlien serta merupakan *extends* dari kelas abstrak *Canvas*. Kelas ini juga akan mengimplementasikan *CommandListener*. Tabel 3.9 di bawah ini merupakan konstruktor dan metode yang terdapat pada kelas Misi.

Tabel 3.9 Metode/konstruktor kelas Misi

Metode/Konstruktor	Keterangan
public Misi()	Konstruktor kelas Misi.
public void commandAction()	Metode untuk memproses permintaan dari pengguna aplikasi.
protected void showNotify()	Metode yang digunakan untuk menunjukkan notifikasi.
public void paint()	Metode untuk menampilkan teks misi.
protected void keyPressed()	Metode untuk mendeteksi penekanan tombol.

3.3.1.2 Diagram Aktivitas

Diagram aktivitas pada gambar 3.2 di bawah ini merupakan diagram yang menunjukan proses-proses yang terdapat pada aplikasi *game* Serangan Alien.



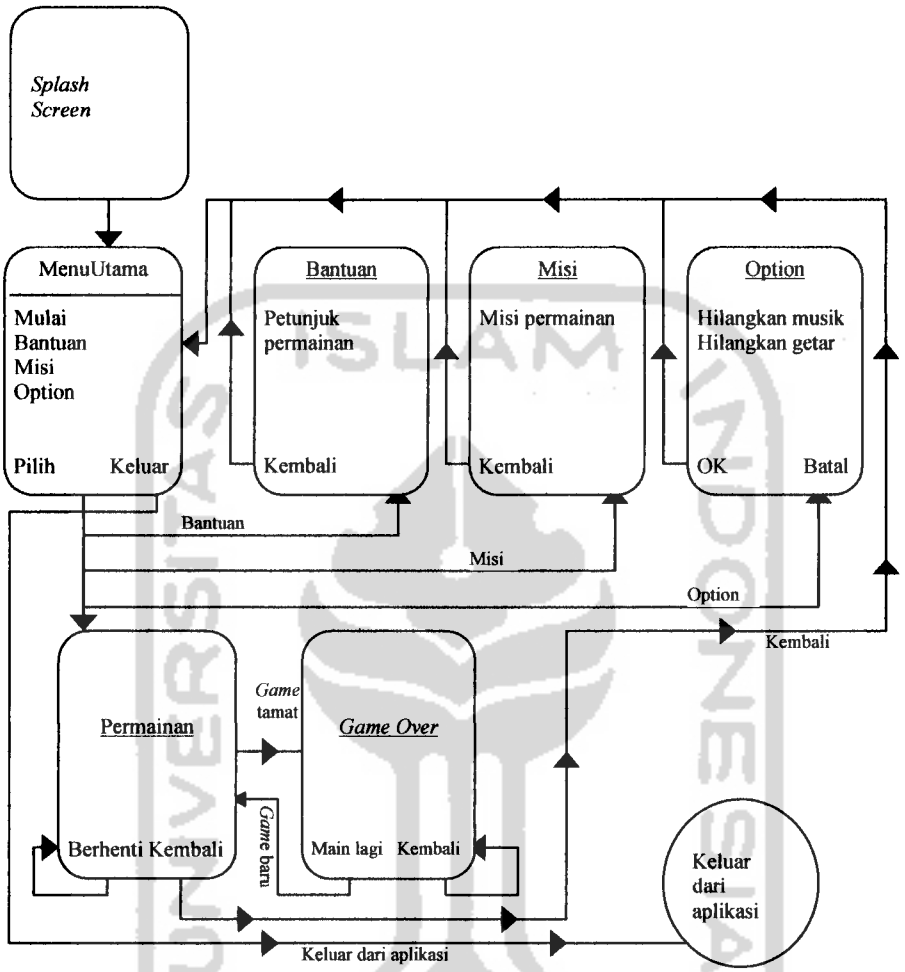
Gambar 3.2 Diagram aktivitas Serangan Alien

Diagram di atas menggambarkan tentang aliran proses-proses pada *game* Serangan Alien. Pada awalnya *MIDlet* akan diciptakan yang berarti aplikasi

Serangan Alien dibuka. *MIDlet* akan menampilkan terlebih dahulu layar *splash* sebelum masuk ke menu utama. Pada menu utama terdapat beberapa menu dan perintah. Menu-menu yang terdapat pada menu utama adalah Mulai, Bantuan, Misi, dan Option. Pada menu utama juga terdapat perintah untuk menutup aplikasi atau keluar dari aplikasi *MIDlet*. Jika pengguna aplikasi memilih menu "Bantuan" dan "Misi" maka akan ditampilkan sekumpulan teks tentang bantuan tombol dan misi *game* Serangan Alien. Pada menu "Bantuan" dan "Misi" hanya terdapat sebuah perintah yang akan membawa pengguna aplikasi ke menu utama. Menu "Option" merupakan menu yang digunakan untuk mengubah *option* yang digunakan dalam aplikasi, yaitu adanya pilihan menghilangkan musik dan getar. Pada menu ini terdapat sebuah subkegiatan yaitu penggantian *option* yang akan disimpan dalam sebuah RMS *(Record Management System)*. Apabila pengguna aplikasi telah merubah pilihan *option* yang digunakan, maka perubahan *option* dapat dirasakan ketika aplikasi *MIDlet* diulang kembali dari awal. Menu "Mulai" akan membuat sebuah mode permainan yang baru. Pada mode permainan terdapat beberapa subkegiatan. Tabrakan merupakan kegiatan yang akan mendeteksi jika terjadi tabrakan antara mobil dan tembakan dari musuh (alien, UFO, dan monster). Tabrakan-tabrakan ini akan membawa pengguna aplikasi ke dalam mode *game over*. Nilai yang dikumpulkan oleh pengguna aplikasi dari hasil tabrakan akan dicek di dalam RMS. Jika nilai yang dihasilkan memenuhi syarat maka akan disimpan di dalam RMS dan akan ditampilkan pada mode *game over* sebagai salah satu nilai terbaik. Pada mode *game over* terdapat subkegiatan yaitu Main lagi dan kembali ke menu utama.

3.3.2 Rancangan Data

Pada pengembangan aplikasi *game* Serangan Alien akan memerlukan perubahan-perubahan pada sistem permainan atau menampilkan pemain *game* yang menghasilkan nilai tertinggi. Semua hal itu membutuhkan *Record Management System* (RMS) untuk melakukan penyimpanan data persisten. Metode-metode yang terdapat pada kelas RMS akan digunakan oleh kelas Layar dan kelas Data. Pada kelas Layar terdapat 1 *field* yaitu Terbaik. Terbaik mempunyai tipe data *integer* dan akan menyatakan nilai-nilai terbaik yang



Gambar 3.3 Rancangan menu

Gambar 3.3 di atas merupakan diagram rancangan menu dan submenu yang terdapat pada aplikasi *game* Serangan Alien. Rancangan dimulai dari *Splash Screen* yang kemudian akan menuju ke menu utama. Pada menu utama terdapat beberapa menu, antara lain: Mulai, Bantuan, Misi, dan Option. "Bantuan" menjelaskan tentang penggunaan tombol. Menu "Misi" menceritakan tentang misi dari *game* Serangan Alien. Menu "Option" terdiri dari submenu yang akan menentukan penghilangan musik dan penghilangan getar dalam aplikasi. Menu mode permainan dapat ditampilkan dengan memilih menu "Mulai". Pengguna aplikasi *game* dapat keluar dari aplikasi dengan memilih *CommandListener* "Keluar" yang ada pada menu utama.

3.3.4 Rancangan Antarmuka

Pada sub-sub bab berikut ini merupakan rancangan antarmuka dari aplikasi.

3.3.4.1 Rancangan antarmuka layar *splash*

Pada gambar 3.4 di bawah ini akan ditampilkan Layar *Splash*. Layar ini akan ditampilkan setelah *MIDlet* diciptakan dan sebelum masuk ke menu utama. Gambar 3.4 di bawah ini merupakan rancangan layar *splash*.

Gambar 3.4 Rancangan layar *splash*

3.3.4.2 Rancangan antarmuka menu utama

Di bawah ini merupakan rancangan antarmuka Menu Utama. Gambar 3.5 akan ditampilkan setelah layar *splash* ataupun setelah permainan telah berakhir (*game over*). Gambar 3.5 memiliki beberapa menu, antara lain : Mulai, Bantuan, Misi, dan Option. Masing-masing menu mempunyai metode-metode tersendiri. Pada Menu Utama ini terdapat juga *CommandListener*, yaitu Pilih dan Keluar. "Pilih" berarti memilih salah satu menu yang tersedia dan "Keluar" berarti keluar dari aplikasi *MIDlet*. Gambar 3.5 di bawah ini merupakan rancangan menu utama.

Menu Utama	
Mulai Bantuan Misi Option	
Pilih	Keluar

Gambar 3.5 Rancangan menu utama

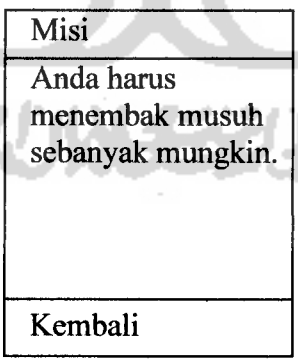
3.3.4.3 Rancangan antarmuka bantuan

Rancangan antarmuka Bantuan di bawah ini merupakan rancangan yang berisi informasi tombol-tombol pada telepon selular yang dapat digunakan untuk bermain *game*. *CommandListener* "Kembali" akan membawa pengguna aplikasi kembali ke menu utama. Gambar 3.6 di bawah ini merupakan rancangan bantuan.

Gambar 3.6 Rancangan bantuan

3.3.4.4 Rancangan antarmuka misi

Rancangan antarmuka Misi di bawah ini menceritakan tentang misi yang terdapat pada aplikasi *game* Serangan Alien. *CommandListener* "Kembali" akan membawa pengguna aplikasi kembali ke menu utama. Gambar 3.7 di bawah ini merupakan rancangan misi.

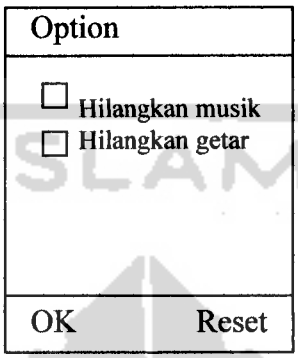


Gambar 3.7 Rancangan misi

3.3.4.5 Rancangan antarmuka option

Rancangan antarmuka Option di bawah ini hanya terdiri dari sebuah dua pilihan yaitu Hilangkan musik dan Hilangkan getar. Jika pengguna mencentang pilihan Hilangkan musik tersebut maka aplikasi *game* tidak menggunakan musik dan jika pengguna mencentang pilihan Hilangkan getar tersebut maka aplikasi

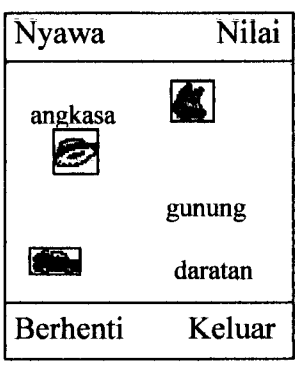
game tidak menggunakan efek getar. Tombol OK akan memproses *option* dan membawa pengguna ke halaman menu utama. Tombol Batal berarti membatalkan proses dan kembali ke menu utama. Gambar 3.8 di bawah ini merupakan rancangan option.



Gambar 3.8 Rancangan option

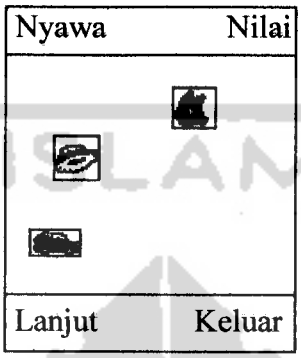
3.3.4.6 Rancangan antarmuka permainan

Rancangan antarmuka permainan merupakan rancangan ketika aplikasi berada pada mode permainan. Rancangan ini akan ditampilkan pada saat pengguna memilih menu "Mulai" pada menu utama. Pada rancangan ini akan ditampilkan *sprite* pada kelas GerakSprite yang mengatur gerakan mobil dan musuh. Selain itu juga akan ditampilkan kelas Layar yang menampung medan pertempuran. Pada bagian kanan atas layar akan ditampilkan nilai yang dikumpulkan oleh pengguna aplikasi. *String* "Nyawa" menunjukkan sisa nyawa yang dimiliki oleh pemain. *CommandListener* "Berhenti" menyatakan *game* dalam keadaan berhenti (*pause*). Gambar 3.9 di bawah ini merupakan rancangan permainan.



Gambar 3.9 Rancangan permainan

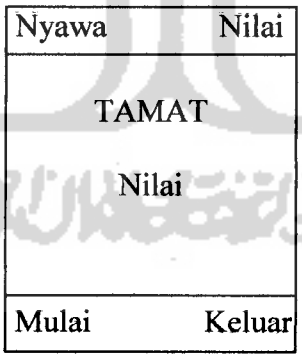
Pada saat *game* dalam keadaan *pause* maka *CommandListener* "Berhenti" diganti dengan *CommandListener* "Lanjut" yang akan melanjutkan mode permainan yang dihentikan sementara tadi. Gambar 3.10 di bawah ini merupakan rancangan permainan berhenti.



Gambar 3.10 Rancangan permainan berhenti

3.3.4.7 Rancangan antarmuka *game over*

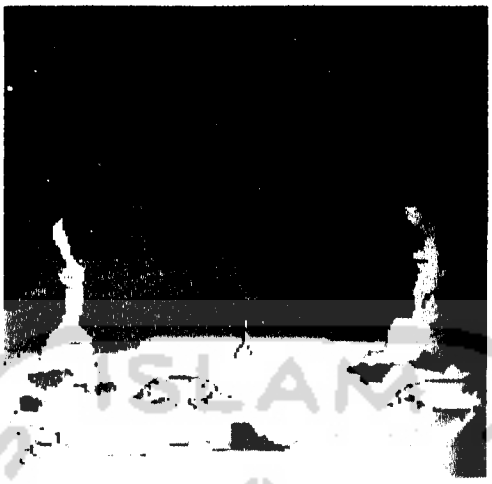
Jika *game* sudah dalam keadaan mode *game over*, maka *CommandListener* "Berhenti" diganti menjadi *CommandListener* "Mulai". *CommandListener* "Keluar" menyatakan keluar dari mode *game play* dan kembali ke menu utama. Gambar 3.11 di bawah ini merupakan rancangan antarmuka *game over.*



Gambar 3.11 Rancangan permainan mode *game over*

3.3.4.8 Rancangan Data Gambar

Sprite akan digunakan dalam pengembangan aplikasi *game* Serangan Alien. *Sprite* yang akan digunakan dalam aplikasi ini terdapat pada kelas GerakSprite. Gambar tersebut dalam format PNG *(Portable Network Graphic).* PNG merupakan format gambar satu-satunya yang didukung oleh J2ME. Kelas *Image* yang terdapat pada paket *javax.microedition.lcdui* dapat digunakan untuk membuat gambar dengan format PNG. *Sprite* yang digunakan adalah *sprite* biasa



Gambar 3.16 Rancangan medan tempur

3.4 Implementasi

Setelah melakukan tahapan analisis dan perancangan sistem, maka dilakukan tahapan selanjunya yaitu tahapan implementasi.

3.4.1 Spesifikasi Sistem Untuk Implementasi

Perangkat lunak yang digunakan dalam pengembangan *game* Serangan Alien adalah sebagai berikut :

1. Java SDK 1.5.0_06, merupakan alat dasar untuk pengembangan aplikasi berbasis Java. Perangkat ini dapat di-*download* dari http://www.java.sun.com.
2. EditPlus v2.1.2, merupakan editor pemrograman di lingkungan Windows. EditPlus v2.1.2 mendukung *syntax highlighting* untuk Java sehingga memudahkan penulisan kode program Java. Perangkat lunak ini dapat di-*download* dari http://editplus.com.
3. Windows XP, sebagai sistem operasi.
4. Perangkat simulasi WTK dari Sony Ericsson SDK 2.2.3, merupakan perangkat yang dapat digunakan untuk mensimulasikan program-program J2ME. Pada perangkat ini sudah tersedia MIDP 2.0.

3.4.2 Implementasi Halaman *Splash Screen*

Gambar 3.17 di bawah ini merupakan keadaan awal aplikasi yang berupa layar *splash* ketika dimainkan pada perangkat *emulator*. Layar *splash* akan muncul dalam bentuk sebuah gambar dan *progressbar*.

dengan memanggil metode *lanjut()*. Metode *lanjut()* berisi perintah untuk membatalkan penghitungan waktu dan langsung menuju halaman berikutnya.

3.4.3 Implementasi Halaman Menu Utama

Halaman menu utama merupakan halaman yang terdiri dsri menu-menu dari aplikasi permainan. Gambar 3.20 di bawah ini merupakan gambar halaman menu utama yang terdiri dari 4 buah menu pilihan yaitu: Mulai, Bantuan, Misi, dan Option.

Gambar 3.20 Menu utama

Masing-masing menu memiliki aksi-aksi tersendiri. Gambar 3.21 di bawah ini merupakan cuplikan program aksi dari masing-masing menu pilihan pada menu utama.

```
public void pilihMenu(int pil) {
    switch(pil){
        case 1 :
            if(canvas == null) {
            try {
                 canvas = new Layar(this, ubah);
                 canvas.mulai();
              }catch (Exception ioe) {
                  System.out.println(ioe);
              }
            } else {
                canvas.lagi();
            }
            Display.getDisplay(this).setCurrent(canvas);
            break;
        case 2 :
            Display.getDisplay(this).setCurrent(bantuan);
            break;
        case 3 :
            Display.getDisplay(this).setCurrent(misi);
            break;
        case 4 :
            try{
              display.setCurrent(new Opt(this).formUbah(ubah));
            }catch(Exception e){    }
            break;
    }
}
```

Gambar 3.21 Program aksi menu

Pada gambar 3.22 di atas terlihat bahwa masing-masing pilihan menu memiliki aksi-aksi tersendiri. Menu "Mulai" akan membawa pengguna aplikasi ke dalam mode permainan. Menu "Bantuan" merupakan halaman yang mendeskripsikan tentang tombol-tombol yang digunakan dalam *game*. Menu "Misi" merupakan menu yang mendeskripsikan misi dari *game* Serangan Alien. Menu "Option" berisi halaman yang terdiri dari 2 buah pilihan pengubahan pada aplikasi *game*.

3.4.4 Implementasi Halaman Menu Bantuan

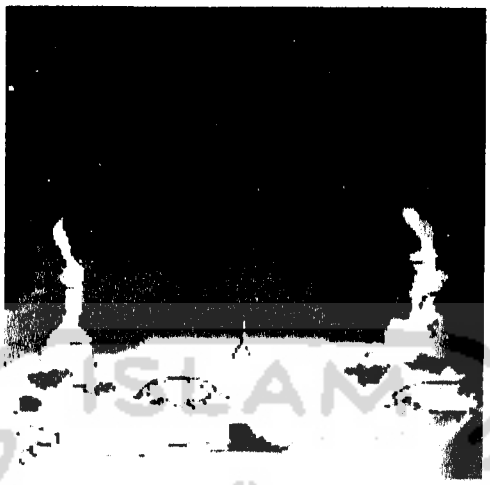
Menu "Bantuan" seperti yang terlihat pada gambar 3.23 terdiri dari 1 buah pilihan yaitu pilihan Kembali. Jika pilihan tersebut dipilih maka aplikasi akan kembali ke menu utama. Gambar 3.22 di bawah ini merupakan implementasi halaman menu bantuan.

pilihan tersebut dikosongkan maka aplikasi akan menggunakan musik dan efek getar. OK akan memproses *option* dan membawa pengguna ke halaman menu utama. Batal berarti membatalkan proses dan kembali ke menu utama. Gambar 3.24 di bawah ini merupakan halaman menu option.

Gambar 3.24 Menu Option

3.4.7 Implementasi Permainan

Halaman permainan pada gambar 3.25 di bawah ini merupakan halaman ketika aplikasi dalam keadaan mode *game play*. Halaman ini dapat ditampilkan dengan memilih menu "Mulai" pada menu utama. Mode permainan merupakan mode dimana pemain dan pasukan musuh bisa saling menembak. Pada mode ini terdapat mobil (pemain), musuh, nilai, nyawa, peluru, dan ledakan. Gambar 3.25 di bawah ini merupakan halaman permainan.



Gambar 3.26 Medan tempur

3.4.9 Mobil

Mobil dikendalikan oleh pemain *game* Serangan Alien. Mobil harus menembak mati pasukan musuh dan menghindari serangan musuh agar bisa memenangkan pertempuran. Mobil mempunyai kemampuan menembak. Tembakan oleh mobil dilakukan secara manual. Gambar 3.27 di bawah ini merupakan metode dasar *addMissile()* yang terdapat pada kelas Layar yang digunakan mobil untuk menembak musuh.

```
private void addMissile(GerakSprite movingsprite)
{
    for(int i = 0; i < 10; i++)
    {
        if(spritePeluru[i].isVisible())
        {
            continue;
        }
        switch(Math.abs(movingsprite.getXSpeed()))
        {
        case 3:
            spritePeluru[i].setFrame(1);
            spritePeluru[i].setPosition(movingsprite.getX() + 5, movingsprite.getY() + 21);
            spritePeluru[i].setXSpeed(movingsprite.getXSpeed() / 2);
            spritePeluru[i].setYSpeed(5);
            break;
```

Gambar 3.27 Cuplikan metode *addMissile()*

Pengguna aplikasi *game* Serangan Alien dapat menggerakkan mobil ke kanan (maju) dan ke kiri (mundur) secara manual. Metode-metode yang mengatur pergerakan mobil diatur pada kelas GerakSprite. Gambar 3.28 di bawah ini merupakan metode untuk menggerakkan mobil.

```
public int getXSpeed()
{
    return xSpeed;
}

public int getYSpeed()
{
    return ySpeed;
}

public void setXSpeed(int i)
{
    xSpeed = i;
}

public void setYSpeed(int i)
{
    ySpeed = i;
}
```

Gambar 3.28 Cuplikan metode untuk menggerakkan mobil

3.4.10 Musuh

Musuh merupakan sekumpulan pasukan yang ditugaskan untuk menembak mobil (pemain). Musuh terdiri dari alien, UFO, dan monster. Gambar 3.29 di bawah ini merupakan cuplikan metode untuk menampilkan musuh.

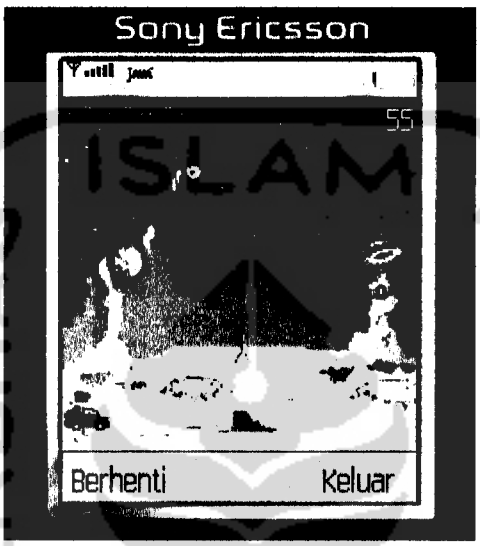
```
private void addAlien()
{
label0:
    switch(Math.abs(rand.nextInt() % 3))
    {
    default:
        break;

    case 0:
        for(int i = 0; i < 3; i++)
        {
            if(spriteAlien[i].isVisible())
            {
                continue;
            }
            placeSprite(spriteAlien[i]);
            spriteAlien[i].setVisible(true);
            break;
        }
        ..........
```

Gambar 3.29 Cuplikan metode untuk menampilkan musuh

3.4.11 Tabrakan

Tabrakan merupakan suatu kejadian dimana peluru bertabrakan dengan musuh atau mobil (pemain). Gambar 3.30 di bawah ini merupakan gambar tabrakan antara peluru dengan musuh.



Gambar 3.30 Tabrakan peluru dengan musuh

Setiap terjadi kejadian tabrakan maka telepon selular akan bergetar. Proses tabrakan akan menggunakan metode *collideWith(sprite[],boolean)* yang akan mendeteksi tabrakan antar *sprite*. Gambar 3.31 di bawah ini merupakan cuplikan program untuk mendeteksi tabrakan peluru yang ditembakkan mobil (pemain) dengan musuh.

```
.........
if(spriteAlien[il].isVisible() && spritePeluru[l].collidesWith(spriteAlien[il], false))
{
    try
    {
        Manager.playTone(54, 100, 100);
    }
    catch(Exception exception1) { }
    addExplosion(spriteAlien[il]);
    spriteAlien[il].setVisible(false);
    spritePeluru[l].setVisible(false);
    score += 10;
    break;
}
.........
```

Gambar 3.31 Cuplikan program tabrakan peluru dengan musuh

3.4.12 Suara

Pada aplikasi *game* Serangan Alien terdapat 2 buah suara yaitu suara musik dan suara nada. Musik merupakan pengiring *game* Serangan Alien. Musik

tersebut tidak ditampilkan pada layar. Gambar 3.34 di bawah ini merupakan gambar pada mode *game over.*

Gambar 3.34 Mode *game over*

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengujian Aplikasi

Untuk mengetahui kinerja *game* Serangan Alien perlu dilakukan pengujian aplikasi baik untuk mengetahui kemungkinan timbulnya kesalahan pada saat aplikasi dijalankan, pengujian terhadap *interface* yang dihasilkan maupun pengujian terhadap performa *game* Serangan Alien. Pengujian yang dilakukan dibagi menjadi 3, yaitu :

1. Pengujian pada *emulator*, dilakukan dengan menggunakan perangkat *emulator* yang berfungsi untuk mensimulasikan bagaimana aplikasi ini bekerja pada perangkat yang sebenarnya. *Emulator* yang digunakan adalah WTK dari Sony Ericsson SDK 2.2.3 yang dapat di-*download* dari situs resmi *vendor* Sony Ericsson (http://www.sonyericsson.com).
2. Pengujian pada telepon selular, dilakukan pada telepon selular Sony Ericsson W300i dan W810i untuk mengetahui apakah *game* Serangan Alien dapat berjalan pada perangkat telepon selular tersebut.
3. Pengujian oleh pengguna, dilakukan dengan pengisian kuisioner oleh pengguna setelah mencoba memainkan aplikasi *game* Serangan Alien. Pengujian oleh pengguna dilakukan untuk mengetahui penilaian subyektif pengguna tentang aplikasi *game* Serangan Alien tersebut.

4.1.1 Pengujian pada *emulator*

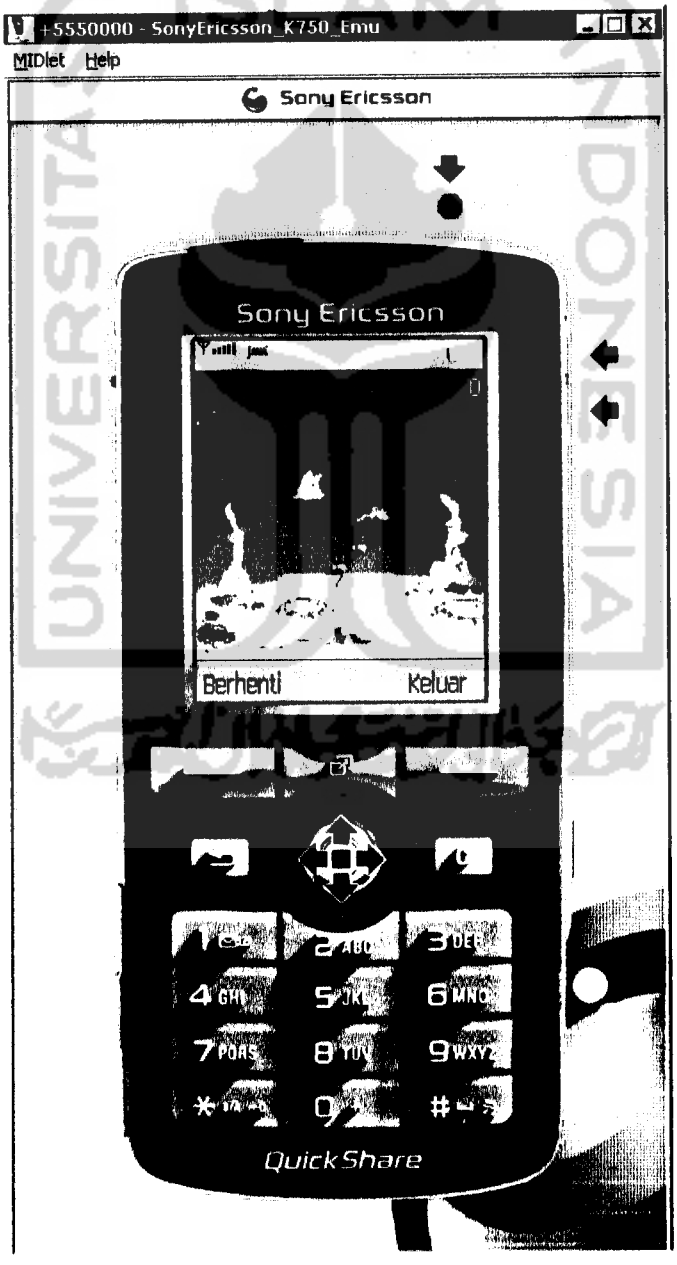
Pengujian pada perangkat *emulator* dilakukan pada perangkat WTK dari Sony Ericsson SDK 2.2.3 untuk jenis K750. Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah aplikasi *game* Serangan Alien tersebut dapat dijalankan pada perangkat *emulator* tersebut. Langkah-langkah pengujian aplikasi pada *emulator* ini adalah sebagai berikut :

1. Setelah menghasilkan berkas *.jar dan *.jad maka aplikasi bisa langsung dijalankan tanpa harus melalui KToolbar yaitu dengan cara

klik

Start ->All Programs ->Sony Ericsson ->J2ME SDK ->WTK2 ->Run MIDP Application.

2. Carilah tempat berkas *.jad disimpan, dalam hal ini Serangan Alien.jad, lalu aplikasi *game* Serangan Alien akan dijalankan.
3. Gambar 4.1 di bawah ini merupakan gambar *game* Serangan Alien pada *emulator*.



Gambar 4.1 Serangan Alien pada *emulator*

4.1.2 Pengujian pada perangkat telepon selular

Pengujian pada perangkat telepon selular dilakukan pada perangkat Sony Ericsson W300i dan W810i. Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah aplikasi *game* Serangan Alien tersebut dapat dijalankan pada perangkat telepon selular tersebut. Langkah-langkah pengujian pada telepon selular adalah sebagai berikut :

1. Salinlah berkas *.jar ke dalam *handset* pada *folder* tertentu.
2. Jalankan *handset* kemudian cari direktori dimana tempat berkas *.jar disimpan. Dalam hal ini Serangan Alien.jar.
3. *Install* berkas *.jar tersebut.
4. Cari dimana lokasi tempat peng*install*an dan mainkan aplikasi.
5. Gambar di bawah ini merupakan gambar hasil pengujian pada telepon selular Sony Ericcson W300i dan W810i.

Gambar 4.2 Serangan Alien pada Sony Ericcson W300i dan W810i

4.1.3 Pengujian oleh pengguna

Pengujian ini digunakan untuk mengetahui penilaian pengguna pada aplikasi yang telah dibuat. Maka untuk mengetahui penilaian pengguna pada *game* yang telah dibuat, sebanyak 10 orang diminta untuk memainkan *game* ini. Setelah mereka memainkan *game* ini, kuisioner dibagikan dan mereka diminta untuk mengisinya. Kuisioner ini digunakan untuk mengetahui kekurangan dari *game* yang telah dibuat. Untuk melihat seperti apa kuisionernya dapat dilihat pada lampiran B.

4.2 Hasil Pengujian dan Analisis

4.2.1 Pengujian pada *emulator*

Dari pengujian yang dilakukan pada perangkat *emulator* pada perangkat WTK dari Sony Ericsson SDK 2.2.3 untuk jenis K750, dapat disimpulkan bahwa aplikasi *game* Serangan Alien dapat berjalan dengan baik pada *emulator* di atas.

4.2.2 Pengujian pada perangkat telepon selular

Aplikasi *game* Serangan Alien bisa berjalan pada perangkat telepon selular di atas yang telah mendukung MIDP 2.0. Sony Ericsson W300i dan W810i merupakan jenis telepon selular yang tidak menggunakan sistem operasi. Pengujian pada telepon selular Sony Ericsson W810i menunjukkan bahwa aplikasi *game* Serangan Alien dapat berjalan dengan baik. Ukuran tinggi dan lebar layar sesuai dengan yang ditampilkan pada *emulator*. Hampir semua fungsionalitas yang berjalan pada *emulator* dapat juga berjalan pada Sony Ericsson W810i. *Game* Serangan Alien juga dapat berjalan pada Sony Ericsson W300i, akan tetapi terdapat beberapa kelemahan-kelemahan pada saat pengujian terhadap perangkat ini, yaitu lebar dan tinggi layar serta beberapa fungsionalitas *game* kurang berjalan dengan baik.

Secara umum *game* Serangan Alien dapat berjalan pada kedua perangkat di atas, walaupun terdapat beberapa kekurangan-kekurangan ketika diimplementasikan. Berdasarkan pengujian dari kedua telepon selular di atas, aplikasi *game* Serangan Alien lebih cocok jika diimplementasikan pada telepon selular Sony Ericsson W810i.

BAB V

SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan analisis dan pembahasan pada aplikasi *game* Serangan Alien dapat diambil beberapa kesimpulan antara lain :

1. Aplikasi *game* Serangan Alien telah berhasil dikembangkan dengan kemampuan-kemampuan sebagai berikut :
    a. Aplikasi *game* ini telah berhasil mendeteksi beberapa tabrakan yang terjadi yaitu antara peluru mobil (pemain) dengan pasukan musuh, peluru musuh dengan mobil.
    b. Aplikasi *game* ini bisa melakukan tembakan, baik itu tembakan dari mobil (pemain) maupun tembakan dari pasukan musuh.
    c. Aplikasi *game* ini menyediakan fungsi untuk mengubah *option*, yaitu menghilangkan musik dan menghilangkan getar.
    d. Aplikasi *game* ini menyediakan getaran jika terjadi suatu tabrakan.
    e. Pasukan musuh pada *game* ini mampu mendeteksi keberadaan mobil.
2. Aplikasi *game* ini dapat berjalan pada beberapa telepon selular yang telah mendukung MIDP 2.0 seperti Sony Ericsson W810i.
3. Aplikasi *game* Serangan Alien berukuran relatif kecil.

5.2 Saran

Aplikasi yang dikembangkan masih memiliki keterbatasan dan masih dapat dikembangkan lebih lanjut. Berikut beberapa saran untuk pengembangan lebih lanjut aplikasi *game* Serangan Alien antara lain :

1. Aplikasi *game* Serangan Alien akan lebih menarik jika dapat dimainkan dalam tampilan 3D (tiga dimensi).
2. Aplikasi *game* Serangan Alien akan lebih menarik jika dapat dimainkan dengan adanya tingkat atau level *game*.

3. Perlu dibuat aplikasi *game* yang mendukung jaringan dan *multiplayer*.
4. Perlu dibuat aplikasi *game* yang dapat berjalan dengan baik pada kebanyakan perangkat telepon selular.

DAFTAR PUSTAKA

[BAR04] Barbagallo, R. *Wireless Game Development in Java with MIDP 2.0.* Texas: Wordware, 2004.

[CRA82] Crawford, C. 1982. *The Art Computer Game Design,* http://erasmatazz.com/free/AoCGD.pdf, diakses tanggal 14 Juni 2007.

[DAY04] Day, B. 2004. *Wireless Game Development: Now and Future,* http://developers.sun.com/techtopics/mobility/getstart/articles/wirelessdev/ WirelessGameDevelopment.pdf, diakses tanggal 26 Mei 2007.

[FOR07] http://www.forum.nokia.com

[FOX02] Fox, D., Verhosek, R. *Micro Java™ Game Development.* Indiana: Addison Wesley, 2002.

[HAR05] Hariyanto, Bambang. *Esensi-esensi Bahasa Pemrograman Java: Disertai Lebih Dari 100 Contoh Program.* Bandung: Informatika, 2005.

[HAR04] Hartanto, Antonius A. *Pemrograman Mobile Java dengan MIDP 2.0.* Yogyakarta: Andi, 2004.

[KRA00] Kramer, W. 2000. *What Is a Game,* http://www.thegamesjournal.com/articles/WhatIsaGame.shtml, diakses tanggal 6 Juni 2006.

[KRO02] Kroll, M. *Java™ 2 Micro Edition Application Development.* Indiana: Sams, 2002.

[LAM04] Lam, J. 2004. *J2ME & Gaming,* http://www.jasonlam604.com, diakses tanggal 14 April 2007.

[MAH02] Mahmoud, Q. *Learning Wireless Java.* California: O'Reilly, 2002.

[NEU03] Neuenhofen, K. 2003. *Designing and Writing Java Action Games for Small Devices,* http://developers.sun.com/techtopics/mobility/blueprints/articles/game, diakses tanggal 28 April 2007.

[SCH02] Schildt, H. *The Complete Reference Java 2 Fifth Edition.* New York: McGraw-Hill, 2002.

LAMPIRAN A

FORMAT KUISIONER

KUISIONER

Petunjuk pengisian :

1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi)
2. Berikan tanda (√) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui

Apakah anda suka bermain *game*? (Ya / Kadang-kadang / Tidak)* Pilih salah satu

No	Pertanyaan	Nilai Jawaban anda				
		1	2	3	4	5
1.	Bagaimana mengenai tampilan antarmuka *game* ini?					
2.	Bagaimana tingkat kesulitan pada *game* ini?					
3.	Apakah anda merasa terhibur dengan memainkan *game* ini?					

Keterangan : Nilai 1 = Sangat kurang
Nilai 2 = Kurang
Nilai 3 = Sedang
Nilai 4 = Baik
Nilai 5 = Sangat baik

LAMPIRAN B
HASIL KUISIONER

KUISIONER

Petunjuk pengisian :

1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi)
2. Berikan tanda (√) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui

Apakah anda suka bermain *game*? (Ya)/ Kadang-kadang / Tidak)* Pilih salah satu

No	Pertanyaan	Nilai Jawaban anda				
		1	2	3	4	5
1.	Bagaimana mengenai tampilan antarmuka *game* ini?				√	
2.	Bagaimana tingkat kesulitan pada *game* ini?			√		
3.	Apakah anda merasa terhibur dengan memainkan *game* ini?					√

Keterangan : Nilai 1 = Sangat kurang
Nilai 2 = Kurang
Nilai 3 = Sedang
Nilai 4 = Baik
Nilai 5 = Sangat baik

KUISIONER

Petunjuk pengisian :

1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi)
2. Berikan tanda (√) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui

Apakah anda suka bermain *game*? (Ya / Kadang-kadang / Tidak)* Pilih salah satu

No	Pertanyaan	Nilai Jawaban anda				
		1	2	3	4	5
1.	Bagaimana mengenai tampilan antarmuka *game* ini?				✓	
2.	Bagaimana tingkat kesulitan pada *game* ini?		✓			
3.	Apakah anda merasa terhibur dengan memainkan *game* ini?					✓

Keterangan : Nilai 1 = Sangat kurang
Nilai 2 = Kurang
Nilai 3 = Sedang
Nilai 4 = Baik
Nilai 5 = Sangat baik

KUISIONER

Petunjuk pengisian :

1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi)
2. Berikan tanda (√) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui

Apakah anda suka bermain *game*? (Ya / Kadang-kadang / Tidak)* Pilih salah satu

No	Pertanyaan	Nilai Jawaban anda				
		1	2	3	4	5
1.	Bagaimana mengenai tampilan antarmuka *game* ini?				✓	
2.	Bagaimana tingkat kesulitan pada *game* ini?			✓		
3.	Apakah anda merasa terhibur dengan memainkan *game* ini?			✓		

Keterangan : Nilai 1 = Sangat kurang
Nilai 2 = Kurang
Nilai 3 = Sedang
Nilai 4 = Baik
Nilai 5 = Sangat baik

KUISIONER

Petu Petunjuk pengisian :

1 1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi)

2 2. Berikan tanda (√) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui

Apal Apakah anda suka bermain *game*? (Ya / Kadang-kadang / Tidak)* Pilih salah satu

No	No	Pertanyaan	Nilai Jawaban anda				
			1	2	3	4	5
1.	1.	Bagaimana mengenai tampilan antarmuka *game* ini?				√	
2.	2.	Bagaimana tingkat kesulitan pada *game* ini?			√		
3.	3.	Apakah anda merasa terhibur dengan memainkan *game* ini?			√		

Ket Keterangan : Nilai 1 = Sangat kurang

Nilai 2 = Kurang

Nilai 3 = Sedang

Nilai 4 = Baik

Nilai 5 = Sangat baik

KUISIONER

Petunjuk pengisian :

1. Berilah jawaban untuk setiap pertanyaan (jangan dikosongi)
2. Berikan tanda (√) dalam menjawab pertanyaan sesuai dengan apa yang anda ketahui

Apakah anda suka bermain *game*? (Ya / Kadang-kadang / Tidak)* Pilih salah satu

No	Pertanyaan	Nilai Jawaban anda				
		1	2	3	4	5
1.	Bagaimana mengenai tampilan antarmuka *game* ini?				✓	
2.	Bagaimana tingkat kesulitan pada *game* ini?			✓		
3.	Apakah anda merasa terhibur dengan memainkan *game* ini?				✓	

Keterangan : Nilai 1 = Sangat kurang
Nilai 2 = Kurang
Nilai 3 = Sedang
Nilai 4 = Baik
Nilai 5 = Sangat baik