# PEMBUATAN APLIKASI GAME PERTEMPURAN BAJAK LAUT MENGGUNAKAN METODE *INDEPENDENT GRAPHIC OBJECT* PADA PONSEL ANDROID

## TUGAS AKHIR

Diajukan Untuk Memenuhi Sebagian Persyaratan Guna Memperoleh Gelar
Sarjana Teknik Informatika Pada Fakultas Teknologi Industri
Universitas Islam Indonesia


Oleh :

**AULIA DIAN PERDANA**

No. Mahasiswa : 07523021

Program Studi : Teknik Informatika

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**YOGYAKARTA**

**2012**

# PEMBUATAN APLIKASI GAME PERTEMPURAN BAJAK LAUT MENGGUNAKAN METODE *INDEPENDENT GRAPHIC OBJECT* PADA PONSEL ANDROID

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Sebagian Persyaratan Guna Memperoleh
Gelar Sarjana Teknik Informatika Pada Fakultas Teknologi Industri
Universitas Islam Indonesia


Oleh :

**AULIA DIAN PERDANA**

No. Mahasiswa : 07523021

Program Studi : Teknik Informatika



**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**YOGYAKARTA**


**2012**

## LEMBAR PENGESAHAN

# PEMBUATAN APLIKASI GAME PERTEMPURAN BAJAK LAUT MENGGUNAKAN METODE *INDEPENDENT GRAPHIC OBJECT* PADA PONSEL ANDROID

**LAPORAN TUGAS AKHIR**

Oleh :

Nama : Aulia Dian Perdana
NIM : 07523021

Yogyakarta, April 2012

Telah Diterima Dan Disetujui Dengan Baik Oleh :

Dosen pembimbing :

Affan Mahtarami, S.Kom., M.T



ii

## LEMBAR PENGESAHAN PENGUJI

# PEMBUATAN APLIKASI GAME PERTEMPURAN BAJAK LAUT MENGGUNAKAN METODE *INDEPENDENT GRAPHIC OBJECT* PADA PONSEL ANDROID

## TUGAS AKHIR

Oleh :

Nama : Aulia Dian Perdana

NIM : 07523021

Telah Dipertahankan Di Depan Sidang Penguji Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Informatika

Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 24 April 2012

**Tim Penguji**

Affan Mahtarami,S.Kom.,MT.

Ketua

Izzati Muhimmah,ST.,M.Sc.,Ph.D

Anggota I

Yudi Prayudi,S.Si.,M.Kom

Anggota II

Mengetahui,

Ketua Jurusan Teknik Informatika

Universitas Islam Indonesia

(Yudi Prayudi,S.Si.,M.Kom)



iii

# LEMBAR PERNYATAAN KEASLIAN HASIL TUGAS AKHIR

Saya yang bertanda tangan dibawah ini ;

Nama : Aulia Dian Perdana

No. Mahasiswa : 07 523 021

Menyatakan bahwa seluruh komponen dan isi dalam laporan Tugas Akhir ini adalah hasil karya sendiri. Apabila dikemudian 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, April 2012

( Aulia Dian Perdana )

**PERSEMBAHAN**

***Bapak dan Mamah***

*Terima kasih doa dan pengharapannya untuk putra tercinta.*

***Kakak Tersayang***

*Terima kasih bimbingannya untuk adik tercinta.*

***Teman-teman***

*Terima kasih canda tawanya yang mengsisi hari-hari penulis.*

## MOTTO

*Jangan Malas Untuk Mencapai Impian (Anonim)*

*Berdoalah Kepada Allah Setelah Berusaha (Anonim)*

# KATA PENGANTAR

*Assallamu'alaikum w.wb.*

Alhamdulillah penulis panjatkan puji dan syukur atas kehadirat Allah SWT, yang telah memberikan rahmat, hidayah serta karunia-Nya, sehingga laporan tugas akhir di Jurusan teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia dapat penulis selesaikan dengan baik. Tidak lupa juga kami panjatkan shalawat serta salam kepada junjungan nabi besar Muhammad SAW yang dengan syafa'atnya kami mengharapkan keselamatan baik di dunia maupun di akherat kelak.

Tugas akhir ini merupakan salah satu syarat yang harus dipenuhi untuk memperoleh gelar sarjana di jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Melalui tugas akhir ini penulis medapatkan pengalaman baru tentang bagaimana membuat suatu aplikasi *game* yang baik beserta dengan dokumentasinya, sehingga dapat dikembangkan lebih lanjut. Selama membuat sistem ini penulis penulis tidak mengalami kesulitan yang berarti karena banyak teman-teman yang membantu dan mendukung penulis.

Dalam pelaksanaan tugas akhir ini, penulis mendapatkan bantuan dari beberapa pihak. Untuk itu, pada kesempatan kali ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Allah SWT yang selalu membimbing setiap langkah kami.
2. Ayahnda Sunyono,S.Ag, Ibunda Hadijah dan keluarga tercinta atas dorongan serta doanya.
3. Bapak Yudi Prayudi, S.Si., M.Kom, selaku ketua jurusan Teknik Informatika Fakutas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Affan Mahtarami, S.Kom.,M.T, selaku dosen pembimbing tugas akhir yang telah memberikan pengarahan dan bimbingan selama proses tugas akhir dan penulisan laporan ini.

Penulis menyadari bahwa dalam penulisan laporan ini masih belum sempurna, karena keterbatasan kemampuan dan pengalaman penulis. Oleh karena itu penulis mengharapkan kritik dan saran untuk dapat membantu dan membangun penulis dimasa yang akan datang. Akhir kata penulis berharap agar laporan ini bisa bermanfaat bagi semua pihak.

*Wassalamu'alaikum wr.wb.*

Yogyakarta, 2012

Penulis

# SARI

*Aplikasi game telah digemari sebagian besar pengguna ponsel android setahun terakhir ini. Game pertempuran banyak diminati oleh ponsel gamer karena memiliki tantangan yang beragam. Selain itu juga mudah dipahami dan dimainkan oleh segala kalangan usia. Sejalan dengan perkembangan teknologi game, platform dan konsol untuk menjalankan aplikasi game juga ikut berkembang. Platform yang paling baru dan memiliki tren cukup positif dikalangan ponsel gamer adalah sistem operasi android. Android merupakan sistem operasi (OS) untuk ponsel yang berbasis Linux. Android menyediakan platform sumber terbuka untuk para pengembang yang menciptakan aplikasi mereka sendiri. Maka pada Tugas Akhir ini penulis mencoba membuat sebuah game pertempuran bajak laut dengan metode grafis independent graphic object. Tujuan dari penelitian ini merancang dan membangun aplikasi game pertempuran bajak laut dengan animasi yang menarik dan memiliki fitur yang memberikan ketegangan dan menyenangkan. Alur cerita berlatar di lautan dengan tujuan menghancurkan perompak. Metode pembuatan aplikasi game ini menggunakan sprite, yaitu gambar dua dimensi yang ditampilkan berurutan kedalam layar untuk membentuk efek yang dinamis. Sedangkan untuk merancang sistem permainan dan membentuk keseluruan aplikasi menggunakan bahasa pemgrograman java. Hasil dari penelitian ini adalah aplikasi game dengan genre battle tentang pertempuran bajak laut yang dapat dijalankan diponsel android. Aplikasi ini dapat dijalankan pada ponsel dengan interaksi layar sentuh maupun tombol. Sehingga dapat disimpulkan bahwa penelitian ini mampu memberikan porsi genre tersendiri untuk game dengan animasi sprite dan pemrograman java.*

*Kata Kunci : gamer, platform, sprite, java, battle, genre, independent graphic object*

# TAKARIR

| | |
|---|---|
| Android | : Sistem operasi ponsel buatan google. |
| Android Market | : Istilah untuk tempat pembelian software android buatan google sebelum berubah nama ke Google Play. |
| *Collision Detection* | : Deteksi tubrukan terhadap dua objek atau lebih dalam sebuah *game*. |
| *Game* | : Istilah untuk menyebut *software* permainan. |
| *Game Engine* | : Suatu *software* khusus dengan fungsi dan *library* bawaan untuk mengembangkan *software* permainan. |
| *Graphic Effects* | : Efek visual dalam game agar animasi terlihat mendekati nyata. |
| *Genre* | : Jenis atau aliran dalam menentukan tipe permainan sebuah game. |
| *Kernel* | : Suatu perangkat lunak yang menjadi bagian utama dari sebuah sistem operasi. |
| *Platform* | : Dasar penopan sistem kerja suatu komputer. |
| Sprite | : Kumpulan gambar 2D yang bergerak bergantian untuk menghasilkan efek animasi. |
| *Sound Effects* | : Efek suara berupa bunyi yang dirancang khusus untuk mendukung permainan. |
| *Touchscreen* | : Antarmuka layar sentuh pada ponsel. |
| *Tool* | : Software khusus untuk membantu mengembangkan game seperti teks editor, atau *image processor*. |
| *Upgrade* | : Istilah untuk tindakan penggantian sebagian atau keseluruhan perangkat sistem dengan perangkat sejenis yang memiliki spesifikasi yang lebih tinggi. |

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# SARI

*Aplikasi game telah digemari sebagian besar pengguna ponsel android setahun terakhir ini. Game pertempuran banyak diminati oleh ponsel gamer karena memiliki tantangan yang beragam. Selain itu juga mudah dipahami dan dimainkan oleh segala kalangan usia. Sejalan dengan perkembangan teknologi game, platform dan konsol untuk menjalankan aplikasi game juga ikut berkembang. Platform yang paling baru dan memiliki tren cukup positif dikalangan ponsel gamer adalah sistem operasi android. Android merupakan sistem operasi (OS) untuk ponsel yang berbasis Linux. Android menyediakan platform sumber terbuka untuk para pengembang yang menciptakan aplikasi mereka sendiri. Maka pada Tugas Akhir ini penulis mencoba membuat sebuah game pertempuran bajak laut dengan metode grafis independent graphic object. Tujuan dari penelitian ini merancang dan membangun aplikasi game pertempuran bajak laut dengan animasi yang menarik dan memiliki fitur yang memberikan ketegangan dan menyenangkan. Alur cerita berlatar di lautan dengan tujuan menghancurkan perompak. Metode pembuatan aplikasi game ini menggunakan sprite, yaitu gambar dua dimensi yang ditampilkan berurutan kedalam layar untuk membentuk efek yang dinamis. Sedangkan untuk merancang sistem permainan dan membentuk keseluruan aplikasi menggunakan bahasa pemgrograman java. Hasil dari penelitian ini adalah aplikasi game dengan genre battle tentang pertempuran bajak laut yang dapat dijalankan diponsel android. Aplikasi ini dapat dijalankan pada ponsel dengan interaksi layar sentuh maupun tombol. Sehingga dapat disimpulkan bahwa penelitian ini mampu memberikan porsi genre tersendiri untuk game dengan animasi sprite dan pemrograman java.*

*Kata Kunci : gamer, platform, sprite, java, battle, genre, independent graphic object*

# BAB I
# PENDAHULUAN

## 1.1 LATAR BELAKANG

Menurut perusahaan *Mobile Analytics* yang dipublikasikan oleh Flurry menunjukkan bahwa aplikasi ponsel telah melebihi penggunaan *web* untuk konsumsi sehari-hari. Aplikasi ponsel yang paling dominan digunakan adalah *game*, kemudian disusul aplikasi jejaring sosial. Peningkatan pengguna aplikasi ponsel ini meningkat setahun terakhir. Hal ini dipicu oleh makin maraknya penjualan ponsel pintar di dunia.

Berdasarkan permintaan konsumen yang selalu meningkat terhadap aplikasi ponsel, faktor lain seperti peningkatan infrastruktur, harga ponsel pintar yang terjangkau dan semakin bersaingnya harga tarif layanan data oleh beberapa penyedia internet membuat peluang bisnis bagi pengembang lokal semakin terbuka. Disisi lain aplikasi ponsel *game* memiliki ceruk pasar tersendiri yang sangat potensial untuk terus dikembangkan.

Pertumbuhan industri *game* termasuk masih baru dan bergerak secara signifikan 10 tahun belakangan ini. Pada awalnya dimulai pada pertengahan 1990-an muncul beberapa studio *game* dan perorangan yang membuat *game* untuk tujuan hobi atau komersial. Selanjutnya pada era 2000-an awal banyak sekali pertumbuhan industri *game* di Indonesia. Perkembangan industri *game* ini tidak diketahui secara pasti pencetus dan pengembang pertama kali. Hal ini disebabkan tidak ditemukan fakta otentik terkait studio *game* atau perorangan yang memulai mengembangkan *game* di Indonesia. Karena itu bisa dikatakan pada era 1990-an perkembangan *game* dimulai karena faktor hobi dan sedikit sekali ditujukan untuk keperluan komersial. Selain itu, konsumen pada era tersebut masih sedikit dan tidak terlalu antusias terhadap perkembangan *game*.

*Game* pertempuran banyak diminati oleh ponsel *gamer* karena memiliki tantangan yang beragam. Selain itu juga mudah dipahami dan dimainkan oleh segala kalangan usia. Hal ini karena alur cerita yang dimiliki *game* pertempuran

secara umum bertujuan untuk mengalahkan musuh. Pertumbuhan aplikasi *game* semakin memiliki potensi besar untuk terus dikembangkan dibeberapa *platform*. Maka dari itu, banyak sekali jenis *game* ponsel yang mengadopsi *game* konsol. Pada *game* pertempuran atau disebut dengan *battle game*, ponsel *gamer* difokuskan untuk mengalahkan musuh dengan alur yang disediakan dan memiliki tingkat kesulitan beragam.

Sejalan dengan perkembangan teknologi *game, platform* dan konsol untuk menjalankan aplikasi *game* juga ikut berkembang. Platform yang paling baru dan memiliki tren cukup positif dikalangan ponsel *gamer* adalah sistem operasi android.

Android merupakan sistem operasi (OS) untuk ponsel yang berbasis Linux. Android menyediakan *platform* sumber terbuka untuk para pengembang yang menciptakan aplikasi mereka sendiri. Selanjutnya aplikasi tersebut bisa digunakan dibeberapa ponsel. Metode pembuatan game *independent graphic object* akan menghasilkan *sprite* yaitu berupa kumpulan gambar 2D yang bergerak bergantian sehingga memberikan efek animasi. Selanjutnya untuk membangun *game* secara keseluruhan maka aplikasi ini akan dibangun dan ditulis menggunakan bahasa pemrograman java.

Dalam ilmu grafis komputer, *sprite* adalah gambar dua dimensi atau animasi yang ditampilkan berurutan kedalam layar untuk membentuk efek yang dinamis. Potongan gambar tersebut ditampilkan secara bergantian pada satu pergerakan dalam sebuah adegan. Proses inilah yang membentuk efek animasi sehingga terlihat gambar akan bergerak secara dinamis. Metode pembuatan *game* menggunakan *sprite* terlihat lebih ringan karena memerlukan memori yang lebih sedikit daripada menggunakan efek 3D (tiga dimensi). Selain itu, dengan beberapa teknik tertentu maka efek yang dihasilkan menyerupai efek 3D.

Salah satu contoh *game* menggunakan *sprite* pada android market adalah *Sky Kitty*. *Game* tersebut masuk pada *genre casual* karena operasi permainannya yang simpel. Cerita *game* ini adalah karakter kucing luar angkasa yang berusaha mengumpulkan poin dengan menangkap benda-benda yang berjatuhan dari langit. Namun tidak terdapat informasi nyawa dan karakter kucing tidak bisa menyerang

untuk menghindari musuh yang juga berhatuhan. Aplikasi yang menggunakan sprite lainnya adalah *Skydive, Fancy Snake dan Smooth Snake*. Aplikasi *game* pertempuran sedikit sekali ditemukan. Maka dari itu mengingat semakin banyaknya ponsel berbasis OS Android, serta potensi yang besar untuk menciptakan aplikasi *game* ponsel. Maka pada Tugas Akhir ini penulis mencoba membuat sebuah *game* pertempuran bajak laut dengan metode grafis *independent graphic object*.

## 1.2 RUMUSAN MASALAH

Berdasarkan uraian latar belakang di atas, masalah yang dirumuskan adalah sebagai berikut :

1. Bagaimana membangun *game* pertempuran menggunakan metode *independent graphic object* dari potongan gambar 2D menjadi efek animasi.
2. Bagaimana implementasi bahasa java untuk pengembangan *game* yang diterapkan pada ponsel android.
3. Bagaimana cara membangun *game* yang beroperasi pada ponsel android layar sentuh menggunakan navigasi buatan.

## 1.3 BATASAN MASALAH

Batasan masalah diperlukan agar penelitian dapat berjalan secara terarah dan mempermudah skema persoalan yang akan diselesaikan. Selain itu, batasan masalah berfungsi sebagai penegasan agar rumusan masalah yang telah dijabarkan dapat diselesaikan dengan baik. Berikut ini adalah batasan masalah yang dipergunakan dalam penelitian :

1 Aplikasi *game* akan berjalan pada sistem operasi Android versi 1.6 dan setelahnya.

2 Kontroler bisa menggunakan tombol fisik pada ponsel maupun tombol buatan pada layar sentuh.

3 Permainan bersifat pemain tunggal dan *Offline*.

4 Menggunakan sistem skoring dan bertujuan menghancurkan musuh.

## 1.4 TUJUAN PENELITIAN

Adapun tujuan penelitian yang diharapkan adalah :

1. Merancang dan membangun aplikasi *game* pertempuran bajak laut dengan animasi yang menarik dan memiliki fitur yang memberikan ketegangan dan menyenangkan. Alur cerita berlatar di lautan dengan tujuan menghancurkan perompak.
2. Memanfaatkan dan mempelajari pembuatan *game* pertempuran (*battle*) menggunakan bahasa java agar mudah diterapkan pada banyak ponsel Android dan mengadopsi efek animasi *independent graphic object* yaitu *sprite*.

## 1.5 MANFAAT PENELITIAN

Berdasarkan penelitian yang akan dijalankan, maka diharapkan akan memberikan manfaat sebagai berikut :

a. Dalam Dunia Akademik

   Dapat digunakan sebagai referensi untuk mahasiswa lainnya dalam pebuatan suatu *game* khususnya *game* yang memiliki genre *battle* tanpa *game engine*.

b. Dalam Dunia *Game Development*

Memberikan motivasi dan inisiatif kepada *game developer* dalam pembuatan game khususnya *game* jenis *battle*. Pembuatan *game* tanpa *game engine* membuat kita lebih bebas dan leluasa memaksimalkan ide yang dimiliki. Sedangkan menggunakan *game engine* biasanya terpaku pada *tools* dan fungsi yang dimiliki *game engine*. Walaupun pengembangan *game* menggunakan *game engine* masih bisa digunakan untuk pengembangan *game* sederhana, namun kita harus mempelajari dulu fungsi-fungsi dan skrip bawaan dari *game engine* tersebut. Selain itu, kita baru bisa mengimplementasikan algoritma yang kita inginkan ke dalam *game engine* tersebut setelahnya. Pada dasarnya, pembuatan *game* yang membutuhkan *game engine* adalah jika ditemukan permasalahan yang tidak bisa

dikerjakan dengan cara sederhana. Atau untuk mempersingkat waktu pembuatan dengan hasil yang sama dengan pengembangan tanpa *game engine*.

## 1.6 METODOLOGI PENELITIAN

Metodologi yang diperlukan dalam pembuatan *game* ini meliputi pengumpulan data dan pembangunan sistem meliputi perancangan dan implementasi dari sistem yang akan dibangun.

### 1.6.1 PENGUMPULAN DATA

Metodologi ini dilakukan dengan cara mengumpulkan data dari beberapa referensi *game battle* yang relevan dengan permasalahan yang dihadapi, mempelajari komponen-komponen yang terdapat dalam *game battle* tersebut dengan memainkan *game* yang memiliki jenis yang mirip untuk menyegarkan ide.

### 1.6.2 PEMBANGUNAN SISTEM

Pada pembangunan *game* ini akan mempergunakan metodologi seperti berikut :

1. Analisis Kebutuhan Sistem *Game Battle.*

   Mengumpulkan informasi mengenai hal-hal apa saja yang dibutuhkan dalam pembuatan *game battle,* salah satunya adalah penggunaan SDK Android dan *Sprite*.
2. Perencanaan dan Perancangan *Game Battle.*

   Perencanaan tentang alur dan sistem permainan, variabel-variabel yang dibuat dan skenario *game*.
3. Mendesain antarmuka *Game Battle.*

   Membuat sketsa desain antarmuka setiap bagian yang akan diterapkan pada *game battle* nantinya.
4. Pengkodean.

   Tahap-tahap yang berkaitan dengan persiapan pembuatan *game* selanjutnya adalah implementasi dari variabel-variabel yang sudah ada,

maka dari itu pada tahap ini pengkodean terdiri dari beberapa langkah yang harus dilakukan untuk membentuk sistem *game* yang diinginkan :

a. Pergerakan Objek dan Aksi

   Membangun pergerakan objek dan aksi yang dapat dilakukan objek.

b. *Collision Detection*

   Mendeteksi sebuah tubrukan jika terdapat objek yang bersentuhan dengan batasan tertentu.

c. NPC *(Non-Player Character)*

   Membangun objek yang tidak dikendalikan oleh pemain dan akan menjadi lawan dalam permainan.

d. Sistem Pertempuran

   Membangun sistem pertempuran antara objek yang dimainkan dan lawan dalam permainan.

e. Kecerdasan Buatan Musuh untuk Bertempur

   Membangun lawan permainan yang bisa bergerak dan melakukan aksi tanpa dikontrol pemain.

5. Pengujian Sistem dan Revisi.

   Pada tahap ini akan dilakukan uji coba terhadap *game* yang dibuat pada awalnya menggunakan SDK Android Emulator kedalam sebuah ponsel menggunakan sistem operasi Android. Setelah itu dilakukan analisa berupa pengujian apakah semua aspek yang dijalankan sesuai yang diharapkan. Pada tahap selanjutnya, *game* memerlukan tahap revisi dan penyempurnaan agar bisa digunakan secara maksimal.

## 1.7 SISTEMATIKA PENULISAN

Dalam penyusunan laporan Tugas Akhir, disusun per bab dan berurutan untuk mempermudah pembahasannya. Secara garis besar, sistematika dari penulisan terdiri atas lima bab, yaitu :

**BAB I PENDAHULUAN**

Diawali dengan penjelasan mengenai latar belakang masalah, kemudian dilanjutkan dengan rumusan masalah, batasan masalah, maksud dan tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

**BAB II LANDASAN TEORI**

Membahas lebih dalam mengenai landasan teori yang dapat membantu para pembaca dalam memahami implementasi yang akan dilakukan. Landasan teori tersebut mencakup tentang *game, game development, sprite,* sistem operasi Android, perangkat lunak dan bahasa pemrograman yang digunakan.

**BAB III METODOLOGI**

Bagian ini terdiri dari pembahasan komponen-komponen yang akan dibangun dalam *game battle*, analisis kebutuhan dan perancangan sistem yang akan dibuat. Pembahasan sistem *game battle* mencakup pembahasan tentang komponen-komponen *game battle* yang akan dibuat. Analisis kebutuhan membahas tentang kebutuhan komponen-komponen *game battle*, penerapan pada ponsel yang menggunakan sistem operasi Android, kebutuhan antarmuka serta kebutuhan perangkat lunak dan perangkat keras yang akan digunakan.

Sedangkan perancangan komponen-komponen *game battle* membahas tentang metode perancangan, hasil perancangan dan perancangan antarmuka.

**BAB IV HASIL DAN PEMBAHASAN**

Berisi implementasi komponen-komponen *game battle* dan analisis kinerja sistem. Implementasi merupakan tahap dimana sistem siap dioperasikan pada tahap sebenarnya, sehingga akan diketahui apakah sistem yang dibuat sesuai dengan yang telah direncanakan. Sedangkan analisis kinerja sistem berisi pengujian program yang telah diimplementasikan kedalam sebuah *game battle* yang bisa berjalan pada *mobile phone* bersistem operasi Android.

## BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan dari hasil implementasi dan analisa kinerja yang telah dikembangkan. Serta saran-saran yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dari sistem yang telah dikembangkan.

# BAB II
# LANDASAN TEORI

## 2.1 PENGERTIAN *GAME*

*Game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan *game*. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan. (Nilwan,1995).

### 2.1.1 KLASIFIKASI *GAME*

Klasifikasi *game* menurut Sibero (2009) berdasarkan jenis *platform* atau alat yang digunakan terbagi menjadi :

1. ***Arcade games,*** yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah atau tempat khusus dan memiliki *box* atau mesin yang memang khusus di desain untuk jenis *video games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa masuk dan menikmati, seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil beserta transmisinya.
2. ***PC Games,*** yaitu *game* yang dimainkan menggunakan *Personal Computers* dan menggunakan sistem operasi seperti Windows atau MACINTOSH.
3. ***Console games,*** yaitu *game* yang dimainkan menggunakan konsol tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.
4. ***Handheld games,*** yaitu *game* yang dimainkan dikonsol khusus video *game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.
5. ***Mobile games,*** yaitu *game* yang dapat dimainkan atau khusus untuk ponsel bersistem operasi atau PDA.

Sedangkan menurut Grace (2005) klasifikasi *game* berdasarkan jenis permainan secara umum adalah :

1. ***Action Shooting,*** video *game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga *timing*, inti dari *game* jenis ini adalah menembak.
2. ***Fighting,*** jenis ini memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus, hafal caranya dan lancar mengeksekusinya, pengenalan karakter dan *timing* sangatlah penting. Berbeda seperti *game* aksi pada umumnya yang umumnya hanya melawan *Artificial Intellegence* atau istilah umumnya melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya. Seri *Street Fighter, Tekken, Mortal Kombat, Soul Calibur* dan *King of Fighter*.
3. ***Action Adventure,*** *Game* jenis ini sudah berkembang jauh hingga menjadi jenis campuran *action beat-em up*, dan ditahun 2000-an, jenis ini cenderung untuk memiliki visual 3D dan sudut pandang orang ketiga. *Tomb Rider, Grand Theft Auto* dan *Prince of Persia* termasuk didalamnya.
4. ***Adventure,*** *Game* murni petualangan ini lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.
5. ***Simulation,*** *game* jenis ini seringkali menggambarkan dunia didalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Seperti mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. *Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: *Sim City, The Sims,* dan *Tamagotchi.*

6. ***Role Playing,*** *game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh atau peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain,biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman maupun makhluk peliharaan.
7. ***Strategy,*** kebalikan dari *game* jenis *action* yang berjalan cepat dan perlu refleks secepat kilat, video *game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. *game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusal pelatihan tempur, tergantung dari tema ceritanya. Pemain *game* strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan *game action*. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumberdaya uang, besi, kayu dan minyak hingga ke pembelian dan peng-*upgrade*-an pasukan atau teknologi. *Game* jenis ini terbagi atas:
    a. ***Real time Strategy,*** *game* berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: Starcraft, Warcraft , dan Command and Conquer.
    b. ***Turn based Strategy,*** *game* yang berjalan secara bergiliran, saat kita mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya, layaknya catur. contoh: *Front*

*Mission, Super robot wars, Final Fantasy tactics, Heroes of might and magic,* dan *Master of Orion*. Sebenarnya ada yang memilah lagi menjadi jenis tactical dan strategi, namun untuk menggabungkannya karena perbedaannya hanya ada di masalah skala dan kekomplekan dalam manajemen sumber dayanya saja.

8. ***Puzzle,*** *game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video *game* petualangan maupun *game* edukasi. Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman.
9. ***SimulationVehicle,*** *game* jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, meskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut.
10. ***Sports,*** bermain sport di PC atau konsol. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater.

Berdasarkan jenis permainan yang terdapat pada Android Market :

1. ***Casual,*** kemungkinan besar menurut penulis, jenis dengan jumlah *game* terbesar yang terdapat pada android market adalah *casual game*. Sebenarnya tidak terdapat istilah yang baku mengenai *casual game* pada android market. Terdapat alur permainan yang mengajak kita untuk meluangkan beberapa menit saja, namun biasanya sangat adiktif. Mekanisme berjalannya *game* sangat simple, seperti melemparkan kertas

kedalam tong sampah, menebas buah yang berlompatan dilayar dan biasanya hanya memiliki satu tombol interaksi saja.

2. ***Brain dan Puzzle,*** permainan yang mengandalkan konsentrasi dan kerja otak berada dijenis *game* ini. Tidak seperti jenis *game* lain yang menuntut kerjasama jari-jemari, pada *game* ini tidak banyak animasi yang ditampilkan. *Game* pada jenis ini misalnya seperti menyusun pecahan-pecahan *puzzle* yang memerlukan kreatifitas dan percobaan dalam setiap sesi *game*. Salah satu yang paling digemari mungkin adalah permainan catur.
3. ***Action* dan *Arcade,*** pada jenis ini, potensi GPU sistem operasi android benar-benar dimaksimalkan. Banyak teknologi pendukung pembuatan *game* seperti visualisasi 3D dan *game engine*. Karena beraneka ragam permainan pada jensi ini, maka muncullah banyak sub-jenis. Kita mengenal *game first person shooter*, *third person shooter* diberbagai *platform*. Namun terdapat juga *game* dengan sentuhan efek 2D bergaya aksi, seperti *game* radiant yang pada saat pengembangan awal hanya menggunakan efek sederhana.
4. ***Racing,*** terinspirasi untuk mengendarai mobil dari ponsel android, lahirlah jenis *game* racing yang sebagian besar pemain harus mengendarai kendaraan untuk menyelesaikan balapan. Untuk memberikan efek adiktif dan memudahkan pemain mengontrol instrument dalam kendaraan, biasanya *accelerometer* berperan besar disini. Ponsel digoyang kekiri dan kekanan untuk mengendalikan kendaraan yang terdapat pada layar.
5. ***Sports,*** olahraga dikehidupan nyata bisa diadopsi pada ponsel android sesuai cabang olahraga yang disukai. Aturan permainan pada cabang olahraga juga diterapkan pada *game* jenis ini. Namun tidak seluruhnya. Salah satu contoh *game* paling digemari adalah sepakbola. Pangsa pasar pemain *game* ini banyak sekali karena dikehidupan nyata juga cabang olahraga sepakbola memiliki peminat yang cukup banyak.
6. ***Cards* dan *Casino,*** jenis *game* ini mengakomodasi pemain yang tidak mau lepas dari permainan seperti kartu, domino dan permainan dalam casino

dalam dunia nyata. Pada jenis ini kecerdasan buatan sangat ditonjolkan untuk mengatur jalannya permainan. Semakin tinggi tingkat kecerdasan permainan ini biasanya semakin digemari. Maka dari itu, permainan jenis ini membutuhkan baris kode yang banyak namun tidak memiliki efek animasi yang bagus.

7. ***Tower Defense,*** didunia PC dan strategi *game* yang sekarang semakin marak, platform android juga memiliki jenis yang sama. *Tower defense* bahkan menjadi *game* dengan penjualan terbaik di android market sejak pertama rilis. Namun tidak seperti di PC, permainan biasanya tidak terlalu luas dan dibatasi dengan kemampuan hardware masing-masing ponsel. Ide ceritanya biasanya terdapat iblis yang mengirim utusan berupa monster untuk menyerang kastil yang kita dirikan. Tugas kita adalah mempertahankan kastil dari serangan yang dilancarkan sehingga terjadi peperangan untuk menghancurkan musuh. Setiap musuh yang kita bunuh, dikonversi menjadi nilai untuk men-upgrade kastil kita. Konsep yang sangat sederhana namun harus cerdas untuk menyeimbangkan timing yang tepat untuk memenangkan *game* jenis ini.
8. ***Innovation,*** beberapa *game* yang tidak termasuk dalam kategori di android market, namun biasanya memaksimalkan fasilitas perangkat keras ponsel android dikategorikan dalam jenis ini. Permainan ini terkadang melibatkan GPS atau camera untuk alur permainannya. Salah satu contoh yang mudah dipahami adalah *game* menangkap hantu. Tujuan permainan adalah menangkap hantu disekitar kita menggunakan kamera dan GPS untuk melacaknya. Dalam permainan kita harus bergerak ke arah yang dimaksud dan menangkapnya menggunakan tombol yang telah disediakan.

### 2.1.2 KOMPONEN *GAME*

Beberapa komponen *game* yang merupakan bagian penting dari jalannya *game* ponsel diantaranya:

1. **Skenario**

   Ide cerita menghasilkan rangkaian skenario yang dituangkan dalam alur permainan. Skenario biasanya diakhiri dengan satu tujuan yang harus dicapai. Cara yang dilakukan itulah yang menjadi rangkaian skenario *game*.

2. ***Sound Effects*** **(Efek Suara)**

   Efek suara adalah bunyi yang dihasilkan untuk melengkapi animasi dalam adegan *game*. Ketika permainan dijalankan, maka aplikasi *game* ini akan memberikan suasana hidup dengan bunyi yang terdengar ketika peluru ditembakkan. Bunyi yang lain adalah ketika terdapat tubrukan antar objek sehingga permainan akan terkesan lebih seru. Beberapa bunyi tersebut seperti bunyi ledakan, dua objek tubrukan dan sebagainya..

3. ***Graphic Effects*** **(Efek Grafis)**

   Animasi yang dibangun untuk mempercantik *game* ini menggunakan *sprite* , yaitu berupa kumpulan gambar 2D yang bergerak bergantian sehingga memberikan efek animasi. Pada umumnya *game* 2D tidak memerlukan efek animasi untuk tampilannya, namun pada *game* bajak laut ini tidak demikian. Beberapa konsep *image-looping* diterapkan dengan sedemikian rupa sehingga efek perpindahan *image* terlihat hidup. Hal inilah yang menyebabkan animasi terlihat lebih hidup.

4. **Pergerakan Objek**

   Objek dalam *game* akan ditampilkan menggunakan *sprite* yang dihasilkan oleh fungsi dalam kode java. Dalam *game* ini dirancang pergerakan objek sesuai skenario *game*. Salah satunya manuver yang mampu dilakukan karakter. Objek dalam *game* juga memiliki kondisi yang dibatasi ruang. Konsep ini menghasilkan istilah *collision detection* atau deteksi tubrukan. Deteksi yang dimaksud hanya terhadap objek yang dibutuhkan untuk mendukung skenario permainan.

5. **Status *Gameplay***

   Ketika *gamer* melakukan operasi saat aplikasi *game* dijalankan, maka membutuhkan informasi pendukung yang layak *gamer* ketahui. Salah

satunya kondisi karakter seperti nyawa atau informasi perolehan skor. Untuk menghasilkan informasi tersebut, maka dirancang *gameplay* atau misi apa saja yang harus dilakukan. Tujuan untuk menyelesaikan misi dilandasi oleh perancangan sistem pertempuran yang diinginkan.

**6. Elemen Tampilan *Game***

Sebagian *game* yang diproduksi pada masa sekarang memiliki semua elemen umum yaitu berupa tampilan pada layar. semakin lengkap elemen tampilan *game* makin tampak professional karya yang dibuat.

**Elemen-elemen pokok dalam sebuah *game***

Bila di perhatikan secara rinci fungsi tampilan dari berbagai *game* ada yang serupa atau mirip satu sama lain, baik fungsi maupun bentuknya. Walaupun *genre* satu sama lain berbeda namun elemen itu tetap memiliki fungsi dan maksud yang sama. Di bawah ini adalah keterangan dalam urutan acak dari berbagai elemen yang ada dalam sebuah *game*,

***a. Title Screen***

Judul adalah elemen pertama. Tidak sederhana yang kita kira. Membuat judul dari sebuah *game* tidaklah mudah. Harus singkat namun menyiratkan isi dari *game* itu. Menarik perhatian tetapi tetap mudah diingat. Judul yang dibuat harus menggoda calon pembeli karena tersedia ratusan bahkan ribuan judul *game* yang bisa mereka pilih. Jadi memilih judul yang menarik tidaklah mudah untuk kriteria seperti itu.

***b. Menu Screen***

Judul memang penting namun untuk membuatnya lebih menarik adalah tampilan menu grafis yang sesuai dengan judulnya. Mendukung operasi apa saja yang ditawarkan secara umum.

***c. Credits***

*Credits* menunjukan bahwa *developer* yang membuatnya disertai dengan penghargaan kepada pihak yang membantu pembuatan *game*. Pada *game* masa kini *credits* berisikan banyak individu yang

terlibat. Mulai dari *designer*, *programmer*, *sound engineer*, 2D dan 3D *artist* dan sebagainya.

***d. Intro***

Untuk intro bisa digunakan berbagai media. Bisa teks, grafik, maupun animasi. Tujuannya adalah untuk membuaka mood atau perkenalan sebelum player memulai *game*.

***e. User Interface***

Saat *game* dijalankan, maka terdapat tombol navigasi yang bisa dijalankan. Tampilan ini menandakan *gamer* harus melakukan operasi menggunakan *user interface* tersebut. Pada ponsel android mungkin hanya menggunakan layar saja. Namun untuk kebutuhan pengembangan awal melibatkan komponen *hardware* seperti *keyboard*.

***f. Mouse Cursor***

Bentuk *cursor* untuk *mouse* kini bisa bervariasi sesuai dengan tema dari *game* yang dibuat. Untuk kebutuhan *end user* pada ponsel, tampilan ini tidak diperlukan.

***g. Exit Screen***

Tampilan akhir apabila *gamer* telah selesai memainkan *game* disusun disini. Tampilan ini menandakan *game* telah berhenti dijalankan dan kembali pada sistem operasi.

## 2.2 PENGEMBANGAN *GAME*

Pengembangan *game* adalah serangkaian proses dimana sebuah *game* dibuat. Istilah ini mengacu kepada perkembangan *video game* pada umumnya, namun pada pengembangan *game* pada Android mengadopsi keadaan yang sama. Pengembangan *game* dilakukan oleh seorang *developer*, bisa 1 orang atau 1 perusahaan besar. Biasanya, *game* komersial berskala besar dibuat oleh tim pengembang dalam sebuah perusahaan yang mengkhususkan pada *game* komputer atau konsol. Sedangkan *game* yang dibuat untuk keperluan Tugas Akhir ini merupakan rintisan 1 orang dalam bentuk mini *game*.

### 2.2.1 Langkah Non-Teknis dalam Pembuatan Game Pada Android

1. **Membentuk Mekanisme Permainan Utama**

   Langkah yang paling mudah membentuk skema permainan adalah membuat desain *game* terhadap objek yang bergerak. Pembuatan gambar bisa menggunakan editor gambar seperti photoshop atau gimp. Semua komponen utama dalam desain *game* seperti latar, objek bergerak, karakter, panel skor, dan navigasi dirancang kasar pada tahap ini.

2. **Menyusun Alur Cerita Kasar Karakter Utama**

   Permainan harus dimulai dengan sebuah cerita dan berakhir dengan cerita pula. Alur cerita sederhana bisa diaplikasikan pada permainan yang akan kita bangun. Pada umumnya cerita sederhana mudah dipahami dan tidak membuat pemain merasa bingung karena harus menghafal skenario permainan yang harus diikuti. Untuk kelas permainan sederhana, cerita yang simpel tapi menarik layak diperhatikan. Lebih baik mengambil segmen yang paling banyak disukai banyak orang. Banyak cerita yang terlalu bagus namun ketika diaplikasikan dalam pembuatan *game* berubah menjadi membingungkan. Maka dari itu, susunlah alur cerita sesuai ide permainan dengan sudut pandang karakter utama.

3. **Menggambar Sketsa Grafis Berdasarkan Alur Cerita Dan Karakter**

   Sementara cerita apik tentang angkasa, monster, peperangan dunia akan menjadi sangat menarik. Tanpa didukung sumber daya yang mencukupi, maka tidak akan menghasilkan *game* yang menarik. Desain harus sesuai dengan alur cerita dan karakter yang terdapat pada *game*. Komposisi warna juga harus diperhatikan agar sejuk dipandang terus menerus namun tidak membosankan. Sketsa grafis sebaiknya satu tema.

4. **Membuat Sketsa Gambar Yang Akan Ditampilkan Pada Layar.**

   Setelah sketsa berdasarkan alur utama terbentuk, penuangan ide pada editor gambar menjadi hasil akhir objek yang akan dibangun. Gambar seperti karakter utama sudah terbentuk berupa potongan *sprite*. Selanjutnya untuk mempermudah penyusunan ide, sebaiknya disusun

menyerupai diagram transisi yang akan ditampilkan pada layar. Semua elemen desain dan transisi dijelaskan urutan kejadiannya agar mudah dipahami dan terorganisir ketika masuk dalam pengkodean.

### 2.2.2 Langkah Teknis dalam Pembuatan *Game* pada Android

1. **Menentukan *Genre Game***

   Menentukan apakan *game* tersebut akan mempunyai *genre* Casual seperti Antigen yang dirilis oleh Battery Powered Game, Puzzle seperti Super Tumble yang dirilis oleh Camel Games atau U Connect yang dirilis oleh BitLogik, Action dan Arcade seperti Exzeus yang dirilis oleh HyperDevBox. *Game* dengan pengelompokan *genre* bisa ditemui dengan mudah di Android Market.

2. **Menentukan *Tool* yang Digunakan**

   Biasanya *game* dibuat dengan bahasa pemograman yang populer. Namun untuk pengembang pemula terdapat beberapa pilihan seperti bahasa java. Selain itu software yang khusus dibuat untuk membuat *game* dengan *genre* tertentu juga disediakan, terdapat software yang menggunakan bahasa pemograman dalam pembuatan *game* atau sama sekali tidak membutuhkan pemograman dalam pembuatannya. Sehingga hanya perlu menggunakan *mouse* untuk mengatur jalannya *game*, karakter jagoan, musuh dan beberapa komponen *game* lainnya. Software untuk membuat *game* sangat membantu dalam membuat sebuah *game*. Dalam perkembangannya, munculah *game engine* untuk memberikan keleluasaan kepada pembuat *game* sehingga lebih dinamis.

   Namun pada pemrograman di ponsel android, bahasa java masih menjadi poin penting mengingat android berjalan menggunakan bahasa java.

3. **Proses Pembuatan *Game***

   Komponen pembentuk *game* telah disiapkan diawal pembuatan *game*. Pada tahap ini merupakan proses panjang yang dilakukan oleh pembuat *game* untuk menghasilkan suatu *game* yang akhirnya dapat dimainkan. Mulai dari teknologi bahasa pemrograman yang dipilih, efek animasi yang

digunakan seperti gambar 2D saja atau sudah menggunakan OPENGL ES. Kemampuan sumber daya layak diperhatikan untuk permulaan pembuatan *game*.

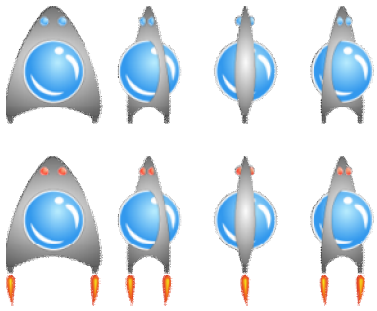
4. **Pengujian *Game***

   Program yang telah selesai harus diuji untuk menentukan apakan layak dimainkan. Pengujian awal dilakukan oleh pembuatnya sendiri yang dinamakan *alpha test* untuk menentukan kesalahan logika dalam alur permainan. Setelah melewati tahap ini maka dilakukan *beta test* yaitu pengujian yang dilakukan oleh orang lain. Pada tahap ini, diharapkan terdapat saran, ide dan masukan untuk mengembangkan program permainan selanjutnya.

## *2.3 SPRITE*

*Sprite* adalah kumpulan gambar 2D yang bergerak bergantian sehingga memberikan efek animasi. Selanjutnya untuk membangun *game* secara keseluruhan maka aplikasi ini akan ditulis menggunakan bahasa pemrograman Java.

Dalam ilmu grafis komputer, *sprite* adalah gambar dua dimensi atau animasi yang ditampilkan berurutan kedalam layar untuk membentuk efek yang dinamis. Potongan gambar tersebut ditampilkan secara bergantian pada satu pergerakan dalam sebuah adegan. Proses inilah yang membentuk efek animasi sehingga terlihat gambar akan bergerak secara dinamis. Metode pembuatan *game* menggunakan *sprite* terlihat lebih ringan karena memerlukan memory yang lebih sedikit daripada menggunakan efek 3D (tiga dimensi). Selain itu, dengan beberapa teknik tertentu maka efek yang dihasilkan menyerupai efek 3D. Gambar 2.1 adalah contoh sederhana *sprite* yang dibuat menggunakan *image processor* seperti Corel Draw.



**Gambar 2.1. Contoh Sprite Sederhana Kapal Luar Angkasa**

Perkembangan dunia grafis memungkinkan pembuat *game* mengambil *sprite* yang bersifat gratis di situs penyedia gambar *sprite* sepeti di http://www.spriters-resource.com/. Pengolah grafis yang diperlukan bisa menggunakan Adobe Photoshop CS2 ataupun menggunakan software khusus seperti RPG Maker VX. Dibawah ini adalah contoh *sprite* satu karakter Megaman yang komplek.

**Gambar 2.2. Contoh Sprite Megaman Satu Karakter**

Metode pembuatan *game* menggunakan *sprite* lebih dikhususkan pada *mobile phone* atau *device* yang memerlukan memory yang lebih sedikit daripada menggunakan efek 3D (tiga dimensi) yang memakan *resource* terlalu banyak.

## 2.4 BAHASA JAVA

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang

terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal.

Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum atau non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

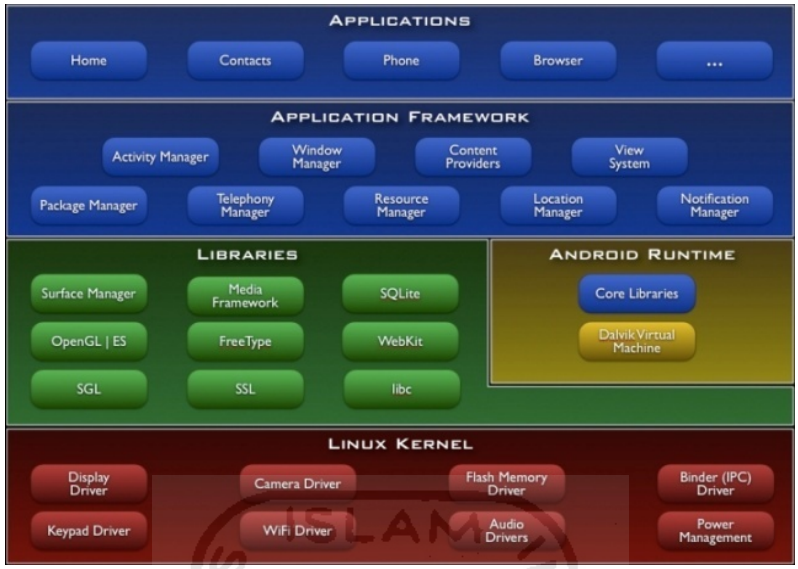
Pada jalur khusus seperti pembuatan *game* dengan menggunakan *sprite*, java dimungkinkan untuk pembuatan *class* dalam struktur *game*. Selain itu membuat animasi lebih hidup dengan *sprite* dan memberikan implementasi dari logika permainan yang telah dibuat sebelumnya.

## 2.5 SISTEM OPERASI ANDROID

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

### 2.5.1 ARSITEKTUR ANDROID

Arsitektur android secara jelas dapat dilihat pada gambar 2.3 yang tersusun dari lapisan-lapisan. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi.




**Gambar 2.3. Arsitektur Android**

Berikut ini penjelasan dari susunan lapisan-lapisan tersebut jika di lihat dari lapisan dasar hingga lapisan teratas:

1. ***Linux Kernel***

   Tumpukan paling bawah pada arsitektur Android ini adalah kernel. Google menggunakan kernel Linux versi 2.6 untuk membangun sistem Android, yang mencakup memory management, security setting, power management, dan beberapa driver hardware.

2. ***Android Runtime***

   Lapisan setelah Kernel Linux adalah Android *Runtime*. Android *Runtime* ini berisi *Core Libraries* dan Dalvik Virtual Machine.

   Setiap aplikasi yang berjalan pada Android berjalan pada processnya sendiri, dengan instance dari Dalvik Virtual Machine. Dalvik telah dibuat sehingga sebuah piranti yang memakainya dapat menjalankan multi Virtual Machine dengan efisien. Dalvik VM dapat mengeksekusi file dengan format Dalvik Executable (.dex) yang telah dioptimasi untuk menggunakan minimal memory footprint. *Virtual Machine* ini *register-based*, dan menjalankan *class-class* yang dikompilasi menggunakan

*compiler* java yang kemudian ditransformasi menjadi format .dex menggunakan "dx" tool yang telah disertakan.

Dalvik Virtual Machine (VM) menggunakan kernel Linux untuk menjalankan fungsi-fungsi seperti *threading* dan *low-level memory management.*

***3. Libraries***

Bertempat di level yang sama dengan Android *Runtime* adalah *Libraries.* Android menyertakan satu set *library-library* dalam bahasa C/C++ yang digunakan oleh berbagai komponen yang ada pada sistem Android. Kemampuan ini dapat diakses oleh *programmer* melewati Android application framework. Sebagai contoh Android mendukung pemutaran format *audio, video,* dan gambar.

***4. Applications Framework***

Lapisan selanjutnya adalah *application framework,* yang mencakup program untuk mengatur fungsi-fungsi dasar *smartphone*. *Application Framework* merupakan serangkaian *tool* dasar seperti alokasi *resource smartphone,* aplikasi telepon, pergantian antar – proses atau program, dan pelacakan lokasi fisik telepon. Para pengembang aplikasi memiliki aplikasi penuh kepada tool-tool dasar tersebut, dan memanfaatkannya untuk menciptakan aplikasi yang lebih komplek. Programmer mendapatkan akses penuh untuk memanfaatkan API-API (*Android Protocol Interface*) yang juga digunakan *core applications.*

***5. Application***

Di lapisan teratas adalah aplikasi itu sendiri. Di lapisan inilah anda menemukan fungsi-fungsi dasar *smartphone* seperti menelepon dan mengirim pesan singkat, menjalankan *web browser*, mengakses daftar kontak, dan lain-lain. Bagi rata-rata pengguna, lapisan inilah yang paling sering diakses. Mereka mengakses fungsi-fungsi dasar tersebut melalui *user interface*.

# BAB III
# METODOLOGI

## 3.1 DESAIN *GAME*

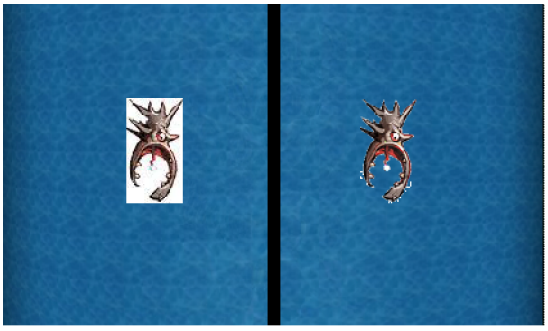
### 3.1.1 KONSEP *GAME*

Aplikasi *game* ini merupakan implementasi untuk menghasilkan sebuah *game* pertempuran dengan latar belakang lautan lepas. Sebuah kapal jagoan akan melakukan baku tembak dengan kapal musuh yaitu kapal perompak. Alur permainan ini mirip dengan *game* legendaris pada ponsel nokia layar *monochrome* dekade lalu bernama *Space Impact*. Letak perbedaannya adalah latar belakang *game* ini tidak pada luar angkasa namun pada lautan lepas. Tujuan kapal jagoan adalah menghancurkan kapal perompak, namun harus melalui rintangan yang telah diberikan agar pertempuran menarik. Aplikasi *game* ini berjalan pada ponsel pintar dengan sistem operasi android.

Animasi yang dibangun untuk mempercantik *game* ini menggunakan *sprite*; yaitu berupa kumpulan gambar 2D yang bergerak bergantian sehingga memberikan efek animasi. Pembuatan *sprite* dilakukan menggunakan pengolah gambar Adobe Photoshop dengan merancang gambar setiap objek. Objek tersebut mewakili satu karakter yang akan ditampilkan pada layar ponsel. Perancangan objek lain seperti *background* dan antarmuka menu juga menggunakan metode yang sama dengan perancangan objek *sprite*. Hasil dari perancangan objek tersebut adalah gambar yang mendukung transparansi dengan format .png.

PNG (*Portable Network Graphics*) adalah salah satu format penyimpanan citra yang menggunakan metode pemadatan yang tidak menghilangkan bagian dari citra tersebut. Gambar yang dihasilkan memiliki ukuran yang relatif besar daripada format JPG (*Joint Photographic Experts Group*) karena detail dari *pixel* yang ditampilkan tidak dikompresi. Pada gambar 3.1 adalah perbandingan gambar yang mendukung transparansi dan tidak mendukung transparansi. Jika gambar disusun dalam dua *frame*, maka akan terlihat perpotongan gambar format .png

mengikuti desain objek asli, sedangkan format .jpg mengikuti resolusi objek dan terlihat kotak setiap sisinya.



**Gambar 3.1. Perbandingan JPG (kiri) dan PNG (kanan).**

Gambar tersebut ditampilkan dalam dua *frame* untuk *background* sebagai *frame* pertama dan gambar monster sebagai *frame* kedua. Rancangan *sprite* inilah yang dikembangkan transisinya, agar ketika mendapat perintah untuk bergerak kekiri maka terdapat pergantian objek dari objek diam ke objek dengan manuver kekiri. Proses pergantian tersebut dilakukan dengan cepat dan interval waktu yang konsisten sehingga menimbulkan efek animasi transisi yang diharapkan. Sehingga dalam *game* ini akan dibuat gambar setiap karakter menggunakan format .png.

Selanjutnya untuk membangun *game* secara keseluruhan maka aplikasi ini akan ditulis menggunakan bahasa pemrograman Java. Hal yang menarik pada pembangunan *game* ini adalah sama sekali tidak menggunakan *game engine.* Kondisi ini bisa dimanfaatkan oleh pengembang lain dengan kemampuan mengerti bahasa java dan tidak terpaku pada fitur yang telah disediakan oleh *game engine*. Sedangkan keuntungan lainnya adalah *game* bisa dikembangkan lebih luwes dan tidak kaku.

### 3.1.2 KARAKTER DAN ALUR PERMAINAN

1. KARAKTER UTAMA

Karakter utama dalam *game* ini adalah sebuah kapal yang melintasi lautan lepas untuk menghancurkan perompak yang telah menguasai samudera. Kapal

laut ini memiliki proyektil dan nyawa agar bisa melakukan pertempuran dengan musuh, atau dalam hal ini kapal perompak.

Rintangan yang menghadang harus dihindari agar kapal tetap berjalan dan menembak perompak. Kapal jagoan memiliki manuver untuk bergerak menghindari ringtangan yang ada.

2. KARAKTER MUSUH

a. Perompak

Kapal yang harus dihancurkan adalah kapal perompak yang telah menguasai samudera selama bertahun-tahun. Kapal perompak memiliki kemampuan untuk menembak kapal jagoan dan bergerak menghindari proyektil yang ditembakkan oleh kapal jagoan. Pergerakkan kapal ini lincah dan membuat alur permainan bertambah seru.

b. Sekutu Kapal Perompak

Sekutu perompak berusaha menabrakkan dirinya ke kapal jagoan, hal ini bertujuan mengganggu pergerakkan kapal jagoan agar tidak mudah mengalahkan perompak. Sekutu ini terdiri dari objek dengan bentuk monster dan batuan karang. Pergerakkannya maju menuju kapal jagoan dengan jumlah yang sangat banyak.

3. ALUR PERMAINAN

Pertempuran yang disuguhkan pada *game* ini ditampilkan sebagai inti permainan. Sebuah kapal jagoan berada di lautan lepas menghadapi hadangan sekutu perompak yang berusaha menabrak kapal jagoan. Kapal jagoan harus menghindari sekutu tersebut agar selamat. Kapal jagoan akan menembakkan proyektil yang bisa mengenai sekutu perompak dan sekaligus menghancurkan kapal perompak itu sendiri sebagai musuh utama.

Laju sekutu mengarah ke kapal jagoan sehingga kapal perompak bisa dengan leluasa menembak. Animasi pada lautan lepas juga diperlihatkan pada aplikasi *game* ini, berupa arah buih yang melaju bersamaan dengan laju awak kapal perompak.

Pemain yang bisa mengalahkan kapal perompak dan menghancurkannya, akan mendapatkan nilai akumulasi. Setiap kapal jagoan menembakkan proyektil dan mengenai sasaran maka akan mendapatkan poin. Hal ini digambarkan pada

fitur pada *game* ini dengan menampilkan skor yang akan terus bertambah jika mencapai sebuah kondisi. Beberapa kondisi nilai yang dimaksud adalah :

1. Sekutu perompak berhasil ditembak maka mendapatkan 100 poin.
2. Terjadi tubrukan dengan sekutu perompak mendapat 100 poin dan nyawa kapal jagoan berkurang 1 poin.
3. Berhasil menembak perompak mendapatkan 500 poin.
4. Poin akan ditambahkan terus pada poin yang telah didapatkan selama *game* ini berjalan.

### 3.1.3 FITUR GAME

*1. SOUND EFFECTS*

Ketika permainan dijalankan, maka aplikasi *game* ini akan memberikan suasana hidup dengan bunyi yang terdengar ketika peluru ditembakkan. Bunyi yang lain adalah ketika terdapat tubrukan antar objek sehingga permainan akan terkesan lebih seru. Beberapa bunyi tersebut seperti bunyi ledakan, dua objek tubrukan dan sebagainya.

*2. GRAPHIC EFFECTS*

Pada umumnya *game* 2D tidak memerlukan efek animasi untuk tampilannya, namun pada *game* bajak laut ini tidak demikian. Beberapa konsep *image-looping* diterapkan dengan sedemikian rupa sehingga efek perpindahan *image* terlihat hidup. Hal inilah yang menyebabkan animasi terlihat lebih hidup. Penggunaan sprite menimbulkan transisi lebih halus dari objek 2D tanpa menggunakan *sprite*. Rancangan *sprite* yang akan dihasilkan dijelaskan pada rincian tabel 3.1 dibawah ini.

**Tabel 3.1. Rancangan jumlah *sprite* aplikasi game.**

| Jumlah | Objek | Operasi / Fungsi |
|---|---|---|
| 3 | Karakter Perahu | Posisi *idle*, gerak kiri dan kanan. |
| 3 | Karakter Perompak | Posisi idle, gerak kiri dan kanan. |
| 6 | Monster | Posisi diam |
| 3 | Latar Lautan | Background |
| 1 | Panel | Status |
| 2 | Karang | Arus laut |
| 2 | Proyektil | Arah maju |
| 5 | Antarmuka Menu | Informasi game |

3. TOMBOL NAVIGASI BUATAN

Ponsel android didesain untuk digunakan dengan layar sentuh untuk mengoperasikan menu didalamnya. *Keypad* yang sudah umum digunakan sebagai navigasi terkadang tidak semua dijumpai pada ponsel android. Maka dari itu, untuk mengoperasikan arah pada *game* dan sebagai ganti dari tombol fisik, pada *game* ini diciptakan tombol navigasi buatan yang akan muncul pada layar sehingga pengguna bisa memainkan *game* ini tidak hanya pada ponsel android yang memiliki layar fisik saja. Namun bisa digunakan pada ponsel tanpa tombol fisik sama sekali. Keypad ini direpresentasikan dengan 2 buah sprite yang berfungsi sebagai tombol dan tembak.

4. PANEL INFORMASI

Selain logika pemrograman *game* yang telah dirancang, terdapat informasi mengenai status dari permainan yang sedang berlangsung yang harus ditampilkan juga. Konsep skoring dirancang agar *game* ini lebih informatif dari sisi eksak. Nilai yang didapatkan setelah menembak musuh, nyawa yang masih dimiliki serta jumlah peluru dipaparkan dipanel informasi. Semua hal itu ditampilkan secara informatif dan *realtime* agar pengguna lebih tahu kondisi kapalnya.

## 3.2 METODOLOGI ANALISIS

Metode analisis merupakan sebuah metode untuk menguraikan aplikasi pertempuran bajak laut yang mempunyai *genre* pertempuran menjadi komponen-komponen untuk diidentifikasi dan dievaluasi permasalahannya. Sistem yang dianalisis adalah sistem yang berisikan segala sesuatu yang berkaitan dengan aplikasi *game* yang akan dibangun, dimulai dari komponen-komponen *game* pertempuran kemudian diterapkan pada ponsel yang bersistem operasi android. Pada tahap ini merupakan fase terpenting ketika memasuki perancangan program, apabila terdapat kesalahan dalah tahap ini maka pada langkah selanjutnya tidak bisa dilakukan proses yang benar. Maka dari itu dilakukan metode sebagai pedoman dalam mengembangkan sistem yang akan dibangun.

## 3.3 ANALISIS KEBUTUHAN

Dari metode dan langkah yang dilakukan maka hasil yang diperoleh adalah sebagai berikut.

### 3.3.1 ANALISIS KEBUTUHAN *INPUT*

Input adalah suatu masukan dan berupa data yang telah ada yang dibutuhkan oleh perangkat lunak sehingga dapat mencapai tujuan yang diinginkan. Salah satu kasus yaitu memilih menu pada tampilan saat akan mulai menjalankan program. Adapun masukkan yang dimaksud untuk *game* pertempuran bajak laut pada ponsel android ini menggunakan layar sentuh (*touchscreen*) dan juga bisa menggunakan tombol fisik yang terdapat pada ponsel tertentu.

### 3.3.2 ANALISIS KEBUTUHAN FUNGSI DAN KINERJA

Fungsi dan kinerja yang dibutuhkan pada aplikasi ini adalah :

1. Membaca input melalui tombol fisik dan layar sentuh ponsel.
2. Melakukan gerakan yang dilakukan objek dalam hal ini musuh untuk menjalankan skenario yang telah dibuat.
3. Pertempuran secara langsung kepada objek dalam hal ini musuh yang mempengaruhi pemain seperti nyawa, ketersediaan proyektil, dan batasan nyawa untuk hancur.

### 3.3.3 ANALISIS KEBUTUHAN *OUTPUT*

Output dari rancangan program ini adalah sebuat aplikasi *game* aksi pertempuran 2D yang bertujuan mengalahkan musuh. Objek dari aplikasi *game* ini adalah komponen-komponen *game* aksi yang dibangun dalam satu skenario atau permasalahan yang harus diselesaikan.

### 3.3.4 ANALISIS KEBUTUHAN PERANGKAT KERAS

Pembuatan aplikasi *game* ini membutuhkan alat untuk mengolah data yang berbentuk teks, gambar dan suara agar menghasilkan efek animasi yang

mendukung alur permainan *game*. Maka dari itu perangkat keras yang digunakan setidaknya menangani masalah pembuatan aplikasi *game* dan selebihnya akan ditanamkan pada sisi pengguna.

Perangkat keras yang digunakan dalam pembuatan aplikasi *game* ini adalah komputer yang memiliki spesifikasi sebagai berikut :

1. Prosesor minimal 1 Ghz atau setara.
2. Kartu grafis minimal 256 MB.
3. RAM minimal 2 GB.
4. Piranti *input* seperti *keyboard* dan *mouse*.
5. Piranti *output* seperti monitor dan *speaker*.

Sedangkan perangkat keras yang digunakan dalam penggunaan oleh *gamer* memiliki spesifikasi sebagai berikut :

1. Ponsel bersistem operasi android minimal 1.6 (Donut).
2. Piranti input ponsel dengan layar sentuh dan tombol fisik untuk navigasi.
3. Piranti output ponsel berupa layar dengan resolusi minimal 320x240 pixel dan speaker.
4. Prosesor ponsel minimal 600 Mhz.
5. Internal memory untuk ruang kosong minimal 10 MB.

### 3.3.5 ANALISIS KEBUTUHAN PERANGKAT LUNAK

Pembuatan aplikasi *game* ini juga membutuhkan perangkat lunak untuk tahapan pembuatan dan menjalankan aplikasi. Sama halnya dengan kebutuhan perangkat keras, kebutuhan perangkat lunak memiliki kategori yang menjadi dua bagian. Pertama adalah perangkat lunak yang digunakan untuk membuat aplikasi dan kedua adalah perangkat lunak yang digunakan untuk penggunaan aplikasi.

Perangkat lunak yang dibutuhkan pada pembuatan aplikasi adalah komputer yang memiliki spesifikasi sebagai berikut :

1. Sistem Operasi, sistem operasi yang digunakan untuk membuat aplikasi *game* ini adalah Ubuntu 9.10 Karmic Koala.
2. Eclipse Galileo, merupakan *text editor* untuk menuliskan bahasa java agar mudah dikompilasi membentuk sebuah file yang akan diinstall di ponsel

android. Penggunaan eclipse mempermudah pembuatan aplikasi *game* karena mendukung ADT Plugin untuk android. Hal ini membuat proses *debug* menjadi lebih mudah.

3. Android Development Tools (ADT), ADT adalah sebuah *plugin* untuk eclipse yang didesain memberikan kemudahan yang mengintegrasikan lingkungan editor agar sesuai dengan pembuatan aplikasi android.[1] Plugin ini terpisah dan bertujuan untuk mempercepat membuat aplikasi baru dalam bentuk *projects*, membuat tampilan antarmuka, menambahkan komponen berdasarkan Android Framework API, keperluan men-*debug* aplikasi menggunakan Android SDK Tools dan mengkompilasi kedalam file berekstensi .apk atau android package kit.
4. SDK Android, SDK Android adalah Software Development Kit yang menjadikan *developer* bisa membuat dan menjalankan aplikasi pada platform berbasi android.[2] SDK Android terdiri dari contoh *projects* dengan kode sumber, *development tools*, *emulator*, dan *library* yang dibutuhkan. Aplikasi ditulis menggunakan bahasa pemrograman java dan akan dijalankan pada Dalvik. Dalvik adalah virtual mesin yang dikhususkan untuk dipasangkan pada level teratas kernal linux.
5. Adobe Photoshop CS2, Adobe Photoshop digunakan untuk membentuk sprite yang beraneka ragam agar objek yang akan ditampilkan pada layar terlihat menarik. Seperti halnya ketika membuat kapal, efek ledakan dan komponen pendukung *game* lain.

Sedangkan perangkat lunak yang digunakan untuk menjalankan aplikasi *game* adalah ponsel android dengan spesifikasi sebagai berikut :

1. Sistem Operasi, sistem operasi yang digunakan mulai dari android 1.6 (Donut), android 2.2 (Froyo) dan setelahnya seperti versi 2.3 (Gingerbread).

[1] http://developer.android.com/sdk/eclipse-adt.html diakses tanggal 27 September 2011.
[2] http://www.webopedia.com/TERM/A/Android_SDK.html diakses tanggal 27 september 2011.

## 3.4 PERANCANGAN PERANGKAT LUNAK

### 3.4.1 METODE PERANCANGAN

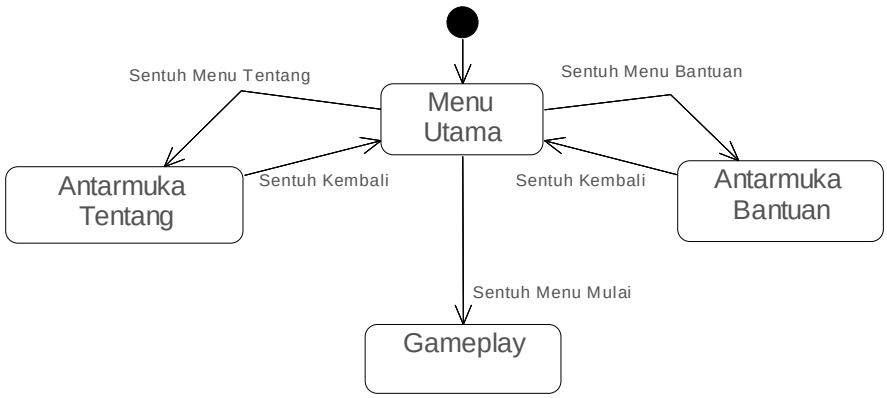
Pembuatan *game* ini didahului dengan merancang alur kegiatan yang harus dilakukan oleh pengguna yang menjalankan. Perancangan *game* ini digambarkan dengan *finite statem machine* atau dikenal dengan istilah FSM. Proses jalannya aplikasi *game* digambarkan kedalam diagram FSM secara umum berdasarkan aksinya. Hal ini dimaksudkan untuk memudahkan *gamer* memahami cara menjalankan aplikasi *game*.

### 3.4.2 HASIL PERANCANGAN

Hasil perancangan berkaitan erat dengan hasil pada tahap analisis. Hal itu disebabkan pada tahap analisis telah menggunakan metode, fungsi-fungsi yang digunakan, perangkat lunak yang digunakan dan antarmuka yang diharapkan.

## 3.5 PERANCANGAN *FINITE STATE MACHINE*

*Finite State Machine* (FSM) digunakan untuk menggambarkan kondisi-kondisi yang terdapat pada saat *game* dimainkan. FSM memberikan keterangan aksi yang dilakukan berikut kondisi yang terjadi jika aksi tersebut dilakukan. Pembuatan FSM pada *game* ini disusun berdasarkan urutan secara umum dan mendetail terhadap sebuah kondisi. Masing-masing diagram FSM tersebut dibuat berdasarkan kondisi yang ada pada *game*. Dibawah ini adalah gambar 3.2 rancangan FSM aplikasi *game* secara umum.

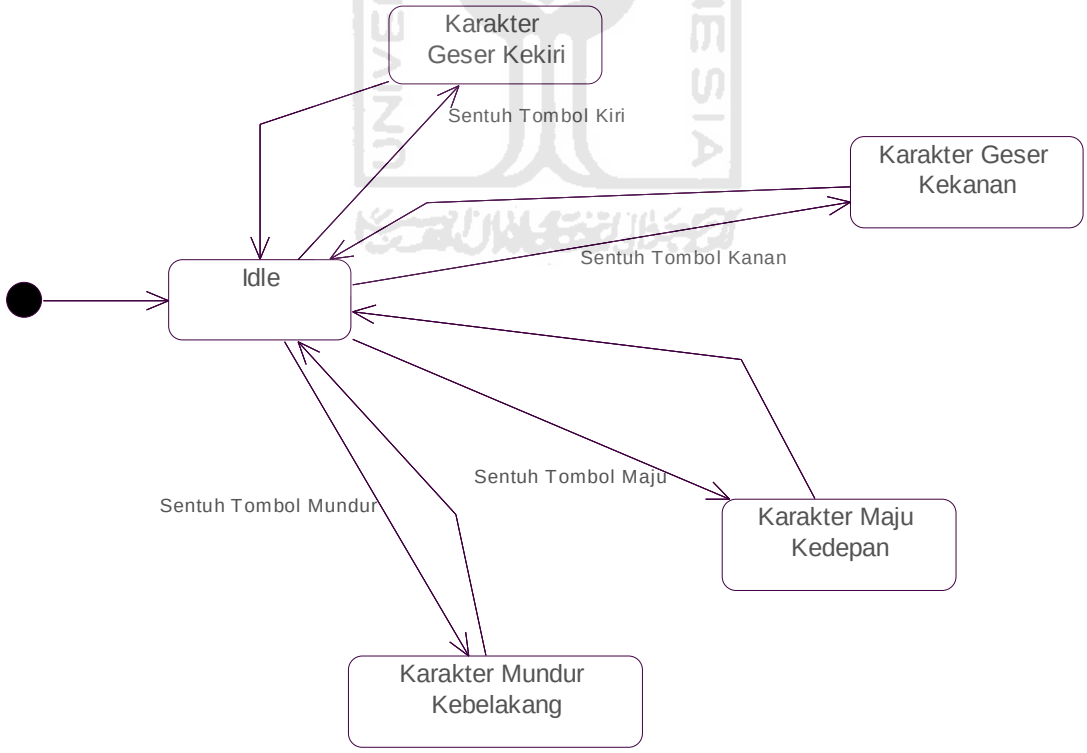



**Gambar 3.2. FSM Aplikasi Game Secara Umum Berdasarkan Kondisi.**

Diagram FSM diatas menjelaskan tentang kondisi-kondisi apa saja yang terjadi pada aplikasi *game* yang akan dibuat secara umum. Pengguna bisa melihat rangkuman kondisi *game* secara umum berikut aksi-aksi apa saja yang harus diakukan oleh aplikasi ini mulai dari awal sampai dengan selesainya *game*. Akan tetapi, alur tersebut masih telalu umum dan tidak dijelaskan secara detil sehingga harus dilanjutkan oleh diagram FSM pendukung yang lain.

### 3.5.1 *STATE MOVE* PADA GAMEPLAY

Penjelasan umum yang terdapat pada gambar sebelumnya bisa dipahami oleh sebagian orang, namun untuk lebih jelasnya akan disertakan diagram FSM untuk menjabarkan kondisi *move* pada *gameplay* serta aksi-aksi apa saja yang harus dilakukan. Berikut ini adalah gambar 3.3 dari diagram FSM untuk menjabarkan kondisi yang dimaksud.




**Gambar 3.3. State Move Pada Game Beserta Aksinya**

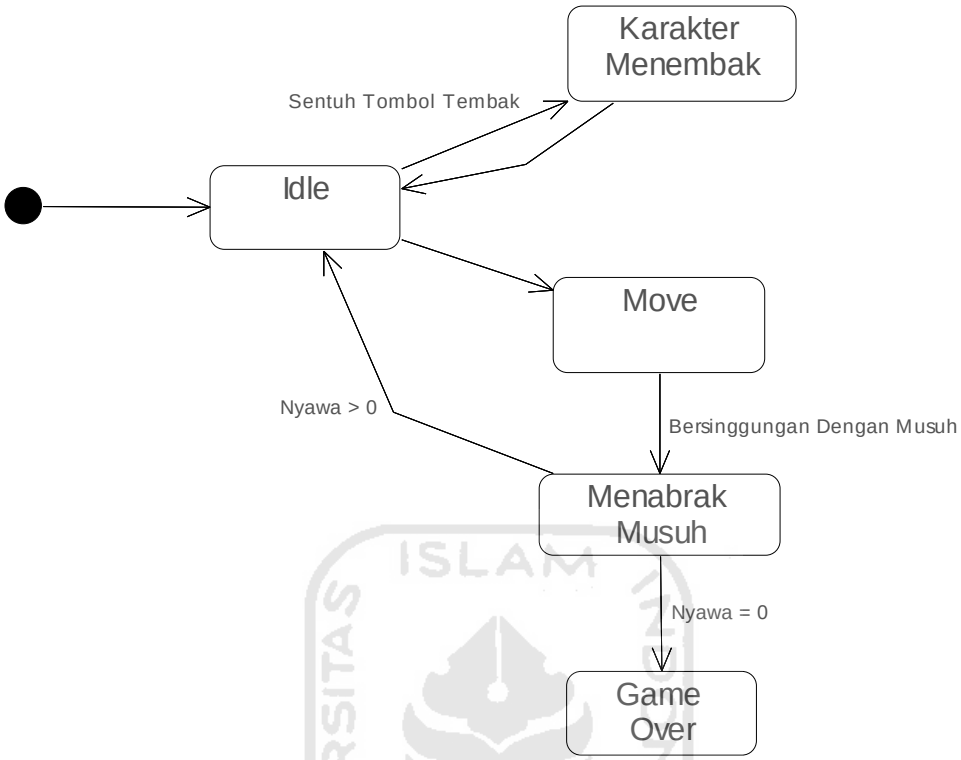
*Substate* dari state move adalah pergerakan kapal jagoan dari kondisi diam kearah samping kanan dan kiri, selain itu juga bergerak maju dan mundur. Kapal jagoan akan bergerak ke samping kiri jika pengguna menyentuh tombol navigasi buatan arah kiri. Kapal jagoan akan bergerak ke samping kanan jika pengguna menyentuh tombol navigasi buatan arah kanan. Kapal jagoan akan bergerak maju jika pengguna menyentuh tombol navigasi buatan arah maju. Serta, kapal jagoan akan bergerak ke belakang jika pengguna menyentuh tombol navigasi buatan arah belakang.

Batasan pada sisi pemrograman harus dilakukan agar pergerakan kapal terlihat nyata. Selain animasi sprite pada saat kapal begerak kearah yang diinginkan, pergerakkan juga dibatasi area layar. Maksudnya adala kapal tidak bisa bergerak terus-menerus jika sudah menyentuh pinggir layar. Logika semacam ini menghindari kapal terlihat menembus layar dan hilang setelah terus menerus diarahkan kesamping kiri misalnya.

### 3.5.2 *STATE ATTACK* PADA *GAMEPLAY*

Setelah kapal jagoan bisa melakukan gerakan yang dapat dikendalikan dengan sempurna. Maka kapal jagoan bisa menembak kapal musuh dengan proyektil yang dimiliki. Kondisi ini dikatakan kapal jagoan sedang menyerang musuh dengan beberapa cara; yaitu dengan menembak dan menabrakkan diri ke kapal musuh.

State *attack* pada penjelasan ini adalah kondisi menyerang dengan kombinasi tombol tembak dan arah navigasi. Menembak dilakukan dengan aksi sentuhan terhadap tombol tembak. Sedangkan menabrakkan diri ke kapal musuh disebut juga menyerang namun harus menerima konsekuensi nyawa berkurang. Berikut ini adalah gambar 3.4 dari diagram FSM untuk menjabarkan kondisi yang dimaksud.




**Gambar 3.4. State *Attack* Pada Game Beserta Aksinya**

Substate dari state attack adalah menembak dan menubruk. Dengan kemampuan menembak, kapal jagoan dapat menghancurkan musuh. Namun kondisi peluru yang tidak selalu ada atau habis menyebabkan kapal harus menabrakkan diri untuk memperoleh nilai. Namun poin utama pada pertempuran ini adalah kemampuan untuk menembak dengan cekatan dan lihai. Kemampuan tersebut lebih mengasyikkan dengan kombinasi menghindari tembakan musuh dan tubrukkan kapal yang terus bergerak maju.

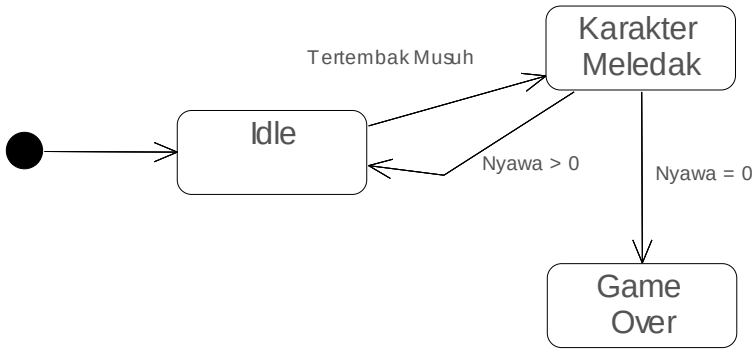
Sama halnya dengan batasan yang terdapat pada kondisi *move,* pada saat menembak dengan peluru, setiap kali menembak maka peluru berkurang sesuai jatah. Lebih jelasnya dijabarkan pada kondisi dibawah ini;

1. Jumlah peluru standar adalah 9 peluru dengan slot maksimal 9.
2. Seiring dengan bertambahnya poin, maka slot peluru semakin bertambah.

3. Slot peluru adalah daya tampung peluru yang dimiliki karakter utama, sehingga jumlah peluru tidak akan melebihi slot peluru.
4. Setiap penembakkan, jumlah peluru berkurang namun slot peluru tidak berkurang.
5. Penambahan peluru menunggu jeda beberapa saat, penambahan berlangsung bertahap satu-demi-satu sampai slot peluru terisi penuh.
6. Jumlah peluru sama dengan 0 maka karakter utama tidak bisa menembak namun masih bisa menghindari tubrukan.

### 3.5.3 *STATE EXPLODE* PADA GAMEPLAY

Keseimbangan pada alur cerita antara kapal jagoan yang menyerang dan diserang ditampilkan disini. Bukan hanya kapal jagoan saja yang dapat menyerang, namun kapal musuh juga memiliki kemampuan serupa yang tidak kalah hebat. Kapal bajak laut mampu menembak dan memerintahkan kapal sekoci anak buahnya untuk menabrakkan diri dengan kapal jagoan. Kondisi ini jika tidak dapat dihindari dengan baik maka bisa menyebabkan tubrukan dengan kapal jagoan. Jika terjadi tubrukan, maka terjadilah ledakan pada dua objek yang saling bersentuhan. Kondisi ini dijelaskan sebagai state *explode* atau karakter yang meledak. Berikut ini adalah gambar 3.5 dari diagram FSM untuk menjabarkan kondisi yang dimaksud.




**Gambar 3.5. State Karakter Meledak Pada Game Beserta Aksinya**

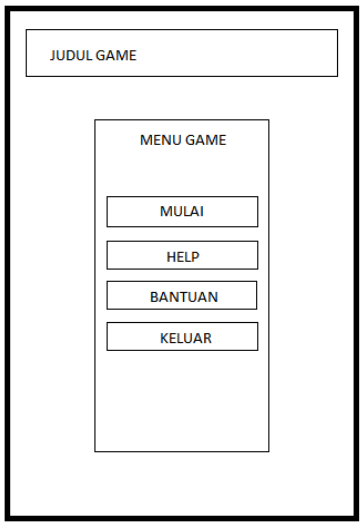
## 3.6 PERANCANGAN ANTARMUKA

Pengguna yang berinteraksi dengan sistem membutuhkan antarmukan untuk melakukan operasi-operasi yang harus dilakukan. Maka dari itu, pada tahap pengembangan sistem perlu dilakukan antarmuka sedemikian rupa sehingga pengguna dapat memaksimalkan fitur yang ditawarkakan oleh sistem tersebut. Pada kasus pembuatan *game* ini, antarmukan dibuat semenarik mungkin sehingga pengguna merasa nyaman. Selain itu juga tidak mengesampingkan nilai interaksi yang informatif dan mudah dipahami.

Merancang antarmuka merupakan awal dari sebuah sistem dapat dioperasikan dengan baik. Oleh karena itu dapat dikatakan kemudahan pengguna dalan menggunakan sistem diukur dari komunikasi yang terjalin pada saat melakukan beberapa operasi pada sistem.

### 3.6.1 ANTARMUKA MENU UTAMA

Ketika aplikasi *game* dijalankan untuk pertama kali, maka pengguna akan disuguhkan tampilan menu utama sebelum bisa menggunakan *game* secara utuh. Menu utama memberikan beberapa pilihan yang dapat dijalankan sesuai dengan menu pilihan yang tertera pada layar. Pada aplikasi *game* ini terdapat 4 pilihan menu yang bisa dipilih, antara lain : mulai, bantuan, about dan keluar. Tampilan rancangan antarmuka untuk menu utama tersebut dapat dilihat pada gambar 3.6 dibawah ini.



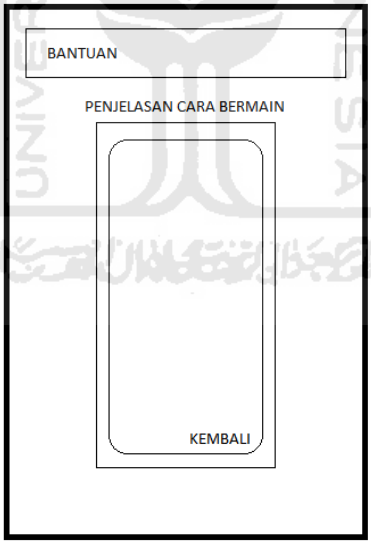

**Gambar 3.6. Rancangan Tampilan Menu Utama**

Setelah itu pengguna bisa memilih salah satu menu untuk melakukan operasi yang telah disediakan. Pada dasarnya menu tersebut mewakili beberapa operasi yang telah dikelompokkan sesuai sifatnya.

### 3.6.2 ANTARMUKA MENU BANTUAN

Pada menu urutan kedua dari atas setelah menu mulai, terdapat menu ***Bantuan***. Menu bantuan berisi cara bermain *game* berikut penjelasan yang harus dipahami pengguna untuk memainkan *game* dengan benar. Penggunaan tombol fisik oleh pada device dan penggunaan tombol buatan pada layar sentuh dijelaskan fungsinya dengan detail pada menu bantuan ini.

Setelah penjelasan dilanjutkan dengan tautan untuk kembali kehalaman sebelumnya yaitu tautan *kembali*. Letaknya pada urutan kanan bawah sehingga pengguna diharapkan telah memahami penjelasan pada bagian isi. Rancangan menu bantuan bisa dilihat pada gambar 3.7 dibawah.

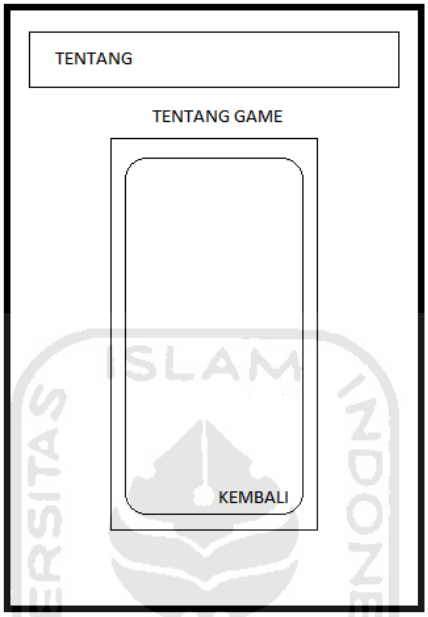



**Gambar 3.7. Rancangan Tampilan Menu Bantuan**

### 3.6.3 ANTARMUKA MENU TENTANG

Pada menu urutan ketika dari atas setelah menu mulai dan bantuan, terdapat menu ***Tentang***. Menu tentang akan ditampilkan judul Tugas Akhir dan nama Pemilik Game Aplikasi beserta Nomor mahasiswa kemudian Logo Universitas Islam Indonesia.

Setelah penjelasan mengenai pemilik *game* dan identitas Tugas Akhir dilanjutkan dengan tautan untuk kembali kehalaman sebelumnya yaitu tautan *kembali*. Letaknya pada urutan kanan bawah sama seperti pada menu bantuan. Rancangan menu tentang ditampilkan pada gambar 3.8 dibawah.

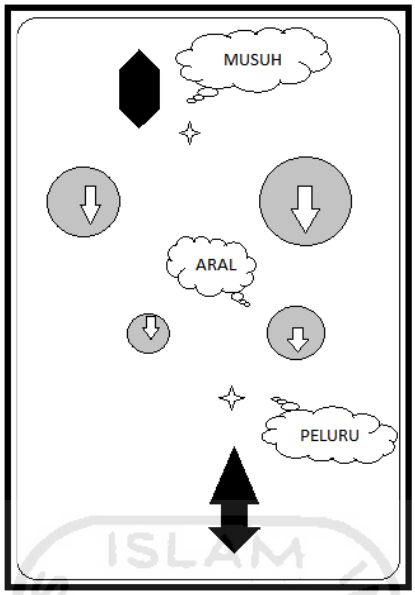



**Gambar 3.8. Rancangan Tampilan Menu Tentang**

### 3.6.4 ANTARMUKA *GAMEPLAY*

Proses utama dari aplikasi *game* ini adalah ketika pengguna memilih menu mulai pada tampilan menu utama. Setelah itu aplikasi utama yaitu *gameplay* permainan bisa langsung dimainkan. Perancangan *game*play pada tahap ini dikembangkan pada saat pengembangan *game* sehingga alur permainan dapat diperluas.

*Gameplay* dirancang sedemikian rupa sehingga tingkat kemudahan pengoperasian tidak terlalu sulit. Selain itu perancangan tombol navigasi virtual juga dilakukan pada tahap ini karena akan berjalan berbarengan dengan permainan. Proses penggabungan tombol dan navigasi juga harus selaras dengan fungsi ergonomis *game* sehingga tidak akan saling mengganggu. Berikut ini adalah rancangan tampilan *game*play bajak laut pada gambar 3.9.




**Gambar 3.9. Rancangan Tampilan Gameplay Ketika Bermain**

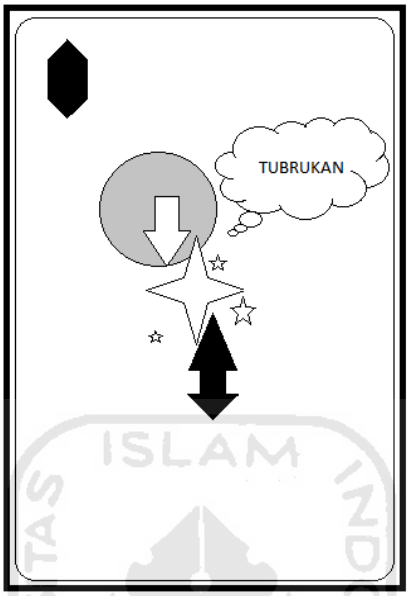
Rancangan tersebut adalah gambaran beberapa objek yang akan ditampilkan pada aplikasi final nantinya. Proses utama ini adalah layar yang paling lama ditampilkan. Pada bagian atas layar, terdapat karakter berbentuk diamond yang mewakili musuh yang harus dihancurkan.

Pergerakan musuh pada *game*play ini dikontrol oleh algoritma agar bergerak dengan sendirinya. Selain itu, karakter musuh pada aplikasi *game* ini bisa menembakkan peluru pada karakter yang kita mainkan. Untuk menambah ketegangan pada aplikasi *game* ini, maka terdapat karakter pengganggu dan pemecah konsentrasi.

Ikon berbentuk lingkaran dan terdapat tanda panah kearah bawah adalah karakter pengganggu. Karakter ini berjumlah sangat banyak dan bergerak kearah karakter yang kita mainkan. Jika terjadi tubrukan antara karakter pengganggu dan karakter yang kita mainkan maka nyawa kita akan berkurang.

Karakter yang kita mainkan mempunyai nyawa berjumlah Sembilan yang akan berkurang setiap terkenan tubrukan dengan karakter pengganggu. Selain itu, karakter yang kita mainkan harus menghindari peluru dari karakter musuh agar nyawa tidak berkurang.

Tampilan tubrukan pada saat karakter penganggu bersentuhan dengan karakter yang kita mainkan dirancangan pada tampilan gambar 3.10 dibawah ini.




**Gambar 3.10. Rancangan Tampilan Gameplay Ketika Bermain**

Terlihat pada gambar rancangan tubrukan terjadi pecahan yang berfungsi sebagai animasi dari deteksi tubrukan yang terjadi. Efek ini memungkinkan pengguna tidak merasa bosan dan menjadikan *game* lebih atraktif.

Tubrukan bisa dihindari dengan menggerakkan karakter yang kita mainkan ke samping kanan dan kiri. Selain itu karakter pengganggu bisa ditembak dengan peluru yang kita punya. Hal ini untuk menghindari tingkat kesulitan yang tinggi dan menambah tantangan pengguna untuk menghancurkan karakter pengganggu.

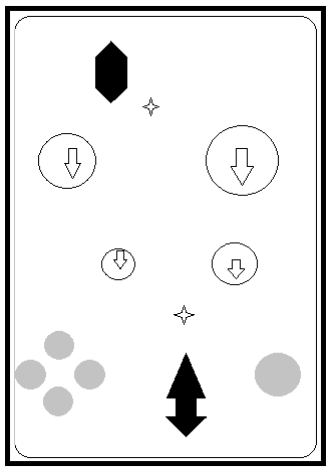
### 3.6.5 ANTARMUKA NAVIGASI KAPAL

Pada beberapa *device* yang memiliki tombol fisik lengkap seperti tombol navigasi arah dan pilihan, *game* ini tidak akan bermasalah pada saat mengoperasikan segala fiturnya. Selain itu pada tahap pembuatan *game* menggunakan emulator memungkinkan menggerakkan karakter kebeberapa arah menggunakan tombol fisik. Namun pada saat implementasi, yaitu pemasangan *game* ke *device* tertentu, tombol buatan navigasi sangat diperlukan.

Pengertian tombol buatan navigasi adalah tombol yang muncul pada layar sentuh menyerupai arah dan menggantikan fungsi tombol fisik. Pengguna bisa menyentuh tombol buatan ini secara langsung pada layar. Tombol ini berada pada tampilan *game*play seolah bertumpukkan. Rancangan tombol buatan navigasi dan tombol tembak dijelaskan pada gambar 3.11 dibawah ini.


**Gambar 3.11. Rancangan Tampilan Tombol Navigasi Buatan**

Sehingga pada saat *game* dioperasikan maka tampilan tersebut akan tampil bersama dengan *game* dan mempunyai fungsi untuk mengoperasikan karakter yang dimainkan. Rancangan akhir dari *game*play setelah ditambahkan dengan fitus navigasi buatan ini ditampilkan pada gambar 3.12.



**Gambar 3.12. Rancangan Tampilan Akhir Gameplay dengan Navigasi**

### 3.6.6 ANTARMUKA PANEL STATUS KAPAL

Pada saat *game* berjalan dan pengguna memainkan karakter, terdapat informasi yang bisa dilihat sebagai informasi dari status karakter. Fitur pada *game* ini adalah menampilkan nilai / score yang akan terus bertambah jika mencapai sebuah kondisi. Beberapa kondisi nilai yang dimaksud adalah ;

1. Karakter pengganggu berhasil ditembak maka mendapatkan 100 poin.
2. Terjadi tubrukan dengan karakter pengganggu mendapat 100 poin dan nyawa karakter kita berkurang 1 poin.
3. Berhasil menembak karakter musuh mendapatkan 100 poin.
4. Poin akan ditambahkan terus pada poin yang telah didapatkan selama *game* ini berjalan.

Informasi tambahan yang terletak pada panel adalah jumlah peluru dan nyawa. Sama seperti kondisi yang telah dirancang pada perolehan poin, nilai yang dimiliki sebagai nyawa dan jumlah peluru bersifat dinamis. Jumlah nyawa dan jumlah peluru akan berkurang selama penggunaan. Berikut adalah kondisi informasi peluru karakter;

1. Jumlah peluru standar adalah 9 peluru dengan slot maksimal 9.
2. Seiring dengan bertambahnya poin, maka slot peluru semakin bertambah.
3. Slot peluru adalah daya tampung peluru yang dimiliki karakter utama, sehingga jumlah peluru tidak akan melebihi slot peluru.
4. Setiap penembakkan, jumlah peluru berkurang namun slot peluru tidak berkurang.
5. Penambahan peluru menunggu jeda beberapa saat, penambahan berlangsung bertahap satu-demi-satu sampai slot peluru terisi penuh.
6. Jumlah peluru sama dengan 0 maka karakter utama tidak bisa menembak namun masih bisa menghindari tubrukan.

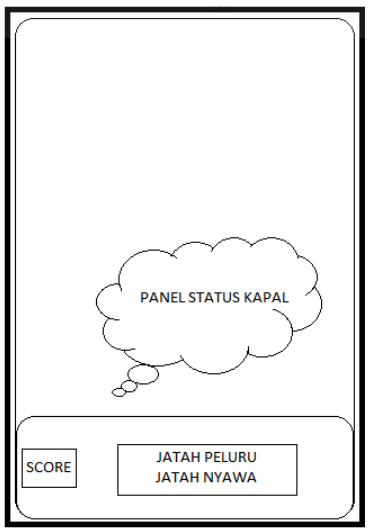
Selain informasi jumlah peluru yang telah dijelaskan diatas, terdapat mekanisme yang hampir mirip dengan kondisi jumlah peluru. Hal yang dimaksud adalah jumlah nyawa karakter.

Kondisi yang serupa diterapkan pada nyawa yang berfungsi sebagai batasan karakter tetap bisa dimainkan atau tidak. Dalam hal ini, jika jumlah nyawa

kurang dari 0 maka *game* tidak bisa dilanjutkan atau *game over*. Maka dari itu, pengguna harus secerdik dan tangkas mungkin menghindari tubrukan dari karakter pengganggu dan tembakan dari karakter musuh. Kondisi dari informasi jumlah nyawa adalah sebagai berikut ;

1. Jumlah nyawa awal pada saat mulai permainan adalah 9 nyawa dengan slot maksimal 9.
2. Seiring dengan bertambahnya poin, maka slot nyawa semakin bertambah.
3. Slot nyawa adalah daya tampung nyawa yang dimiliki karakter utama, sehingga jumlah nyawa tidak akan melebihi slot nyawa.
4. Setiap terjadi tubrukan dengan karakter pengganggu dan terkena tembakan oleh karakter musuh maka jumlah nyawa berkurang namun slot nyawa tidak berkurang.
5. Penambahan nyawa menunggu jeda beberapa saat, penambahan berlangsung bertahap satu-demi-satu sampai slot nyawa terisi penuh.
6. Jika nyawa memiliki nilai 0, maka karakter harus hanya memiliki sisa 1 waktu hidup dan setelahnya ketika terkena tubrukan atau tembakan maka *game over*.

Rancangan dari panel yang kondisinya terangkum diatas ditampilkan pada gambar 3.13 dibawah ini.




**Gambar 3.13. Rancangan Tampilan Panel Status Karakter**

### 3.6.7 ANTARMUKA GAME OVER

Setelah melewati beberapa kondisi yang telah disediakan oleh *game*. Maka terdapat alur batasan dimana kondisi pengguna tidak dapat melanjutkan permainan. Pada kasus ini adalah kondisi *game* selesai atau dikenal dengan istilah *game over*. Kondisi yang terdapat pada *game* pertempuran bajak laut ini adalah ketika nyawa yang terdapat pada panel status kurang dari 0 atau tidak memiliki nyawa lagi. Setelah kondisi tersebut, maka muncullah tampilan *game* over.

Rancangan pada *game* aplikasi bajak laut untuk tampilan *game* over ini dijelaskan pada gambar 3.14 dibawah ini.


**Gambar 3.14. Rancangan Tampilan Game Over**

Antarmuka *game* over sangat sederhana agar memberikan efek kejutan pada pengguna bahwa permainan telah berakhir. Selanjutnya jika pengguna ingin mulai permainan lagi, ditambahkan tautan untuk bisa kembali ke Menu Utama.

## 3.7 RENCANA PENGUJIAN

Pengujian terhadap aplikasi game terbagi menjadi dua bagian *platform*. *Platform emulator* yang digunakan pada tahap pengembangan dan *platform* sebenarnya yaitu ponsel android yang akan menjalankan aplikasi *game* setelah terlebih dahulu diinstal. Pada emulator pengembangan *game*, versi android yang digunakan adalah android 1.6 dan android 2.2. Sedangkan untuk ponsel android akan digunakan ponsel dengan android terbaru dengan versi 2.3.

Setelah pengujian dilakukan dan dianalisa kinerja game pada perangkat baik emulator maupun ponsel android, maka akan dilakukan pengukuran kualitas game menggunakan metode playtesing. Metode ini banyak dilakukan oleh *game developer* untuk memperoleh masukan dari *gamer*. Selain itu dapat diukur berdasarkan variabel pengukuran yang telah ditentukan sebelumnya.

*Playtesting* adalah ujicoba yang dilakukan oleh *game developer* untuk aplikasi *game* yang telah dikembangkan agar diketahui apakah terdapat *bugs* dan sekaligus mengukur kinerjanya. Pada proses *playtest* aplikasi *game* pertempuran ini dilakukan menggunakan metode kuisioner kepada responden yang telah memainkan aplikasi *game*. Responden dipersilahkan memainkan *game* dan menjawab pertanyaan yang telah disiapkan untuk diformulasikan menjadi kelebihan dan kekurangan *game*.

# BAB IV
# HASIL DAN PEMBAHASAN

Hasil dan pembahasan menjelaskan mengenai implementasi perangkat lunak yang meliputi batasan implementasi, implementasi komponen pengembangan game, implementasi antarmuka, analisa kinerja *platform* dan *playtesting*.

## 4.1 BATASAN IMPLEMENTASI

Implementasi merupakan tahap saat aplikasi siap untuk dioperasikan sesuai dengan tahap sebenarnya. Pada tahap ini dipaparkan sejauh mana aplikasi *game* sesuai dengan perancangan pada tahap sebelumnya. Selain itu, aplikasi dijelaskan dalam bentuk tampilan dan penjelasan yang muncul dalam *game*.

## 4.2 IMPLEMENTASI KOMPONEN PENGEMBANGAN *GAME*

Komponen game menghasilkan satu rangkaian pengembangan aplikasi, mulai dari pembuatan awal sampai dengan pengujian. Komponen itu ialah efek suara, efek grafis menggunakan sprite untuk animasi, pergerakkan objek dalam *game*, deteksi tubrukan, status karakter, pergerakkan musuh dan skema pertempuran. Implementasi komponen game tersebut menghasilkan satu aplikasi utuh yang sebenarnya, yaitu aplikasi yang dapat berjalan pada ponsel android.

Komponen *game* seperti *spritesheet* diambil dari situs penyedia *spritesheet* gratis maupun dari *game engine* seperti RPG Maker VX. Sedangan untuk efek suara dan lagu tema selain diambil dari RPG Maker VX juga merupakan gubahan dari lagu tema film *pirates of the carribean.*

### 4.2.1 PENGGUNAAN SPRITE UNTUK ANIMASI *GAME*

Animasi yang dibangun untuk mempercantik *game* ini menggunakan sprite, yaitu berupa kumpulan gambar 2D yang bergerak bergantian sehingga memberikan efek animasi. Sementara pada saat teknologi *game* dibangun menggunakan efek 3D yang canggih, pada android sendiri sesungguhnya memiliki keunggulan dengan memanfaatkan library seperti OPEN-GL ES. Namun biasanya pemanfaatan teknologi itu tidak sejalan dengan sumber daya yang mendukung, seperti beratnya *running* emulator pada komputer ketika pengembangan menggunakan efek 3D, serta pemanfaatan *software* 3D lain secara terpisah. Maka dari itu, jika dilihat

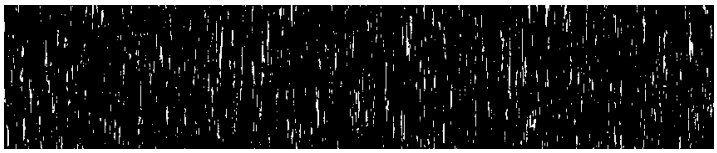
pangsa pasar untuk permainan klasik yang tidak sedikit, penggunaan sprite masih diperlukan untuk memenuhi *gamer* yang ingin memainkan *game* ringan dan bernostalgia pada *game* klasik.

### 4.2.2 LATAR PERMAINAN *GAME*

Pada tahap pembuatan ini dirancang implementasi pembuatan latar permainan untuk objek-objek yang akan bergerak diatasnya. Pembuatan latar akan bernuansa lautan agar menyatu dengan tema pertempuran dilautan. Latar permainan membutuhkan *spritesheet* sebanyak 3 *sheet* yang di-*draw* terus menerus memenuhi layar. Penggunaan 3 *sheet* ukuran kecil untuk satu ukuran layar *game* lebih baik karena tidak memerlukan *load buffer memory* yang besar daripada menggunakan 1 *sheet* ukuran besar sekaligus. Berikut ini adalah latar permainan menggunakan gambar *spritesheet* berurutan untuk ditampilkan pada layar dari atas sampai bawah pada gambar 4.1.

**Gambar 4.1 *Sprite* Latar Permainan**

Latar tersebut tidak bergerak untuk mengurangi *load memory* yang besar, sedangkan untuk menghasilkan efek animasi lautan yang bergerak sesuai arus, maka akan diberikan sprite arus yang bergerak dari sumbu y atas menuju sumbu y bawah. Jumlah potongan sprite tersebut proporsional dihasilkan oleh fungsi initArus() agar menghasilkan arus lautan yang mirip dengan kondisi lautan. Berikut adalah sprite arus lautan pada gambar 4.2.



**Gambar 4.2 *Sprite* Ombak Lautan**

Sprite tersebut memiliki kecepatan konstan yang dihasilkan oleh fungsi drawArus yang didraw terus menerus pada layar sejumlah pixel yang telah didefinisikan pada numArus().

Sedangkan untuk animasi dasar lautan yang bergerak perlahan, dibutuhkan *sprite* batuan karang. Dengan asumsi bahwa dasar lautan bergerak lebih lambat dari ombak yang berada pada permukaan, maka transisi pergerakan batuan karang membutuhkan fungsi baru yang memiliki kecepatan lebih lambat. Perpaduan batuan karang yang bergerak lambat dan ombak dipermukaan yang bergerak lebih cepat pada latar menghasilkan animasi yang dinamis. Berikut ini adalah gambar 4.3 batuan karang sebagai dasar lautan.

**Gambar 4.3 *Sprite* Batuan Karang Dasar Laut**

Sprite tersebut akan terlihat menyatu dengan warna biru lautan karena transparansi diturunkan, sehingga menghasilkan latar permainan dinamis yang diharapkan. Dibawah ini adalah bagian baris kode java untuk menggambar latar yang terdiri dari 3 *spritesheet* lautan :

```
1   void loadBackground() { bg = new Bitmap[3];
2   int[] idx = new int[] {
3   R.drawable.bg1, R.drawable.bg2, R.drawable.bg3 };
4           for (int i = 0; i < 3; i++) {
5                 bg[i] = getImage(idx[i]);          }
```

Baris kode diatas akan memanggil 3 sheet gambar latar yang akan ditampilkan sesuai koordinat kepadatan layar (*density*). Koordinat menggunakan *pixel* dihindari pada pembuatan game ini karena menghindari *ratio* dan jumlah *pixel* pada ponsel android yang berbeda-beda. Sedangkan koordinat kepadatan tidak berpengaruh terhadap letak asli koordinat sesungguhnya. Selanjutnya gambar latar tersebut ditampilkan menurut koordinat sumbu x dan sumbu y pada baris kode berikut ini :

```
1   public void drawBackground(Canvas c) {
2     c.drawBitmap(bg[0], 0.0f, 0.0f, mBitmapPaint);
3     c.drawBitmap(bg[1], 0.0f, 190.0f, mBitmapPaint);
4     c.drawBitmap(bg[2], 0.0f, 380.0f, mBitmapPaint);
5   }
```

drawBitmap berurutan menggambar array bg[] sesuai koordinat sumbu x 0.0f dan sumbu y 0.0f sampai sumbu x 0.0f dan sumbu y 380.0f. Koordinat sumbu x dan y menggunakan tipe data float (f) agar lebih presisi. Sedangkan untuk arah arus ombak sesuai sumbu y baris kode java-nya sebagai berikut :

```
1   void drawArus(Canvas canvas) {
2     lajuArus();
3     for (int a = 0; a < numArus; a++) {
4     canvas.drawRect(arusPosX[a], arusPosY[a], arusPosX[a] + 2,
5                     arusPosY[a] + 2, arusColor[a]);
6           }
7     }
```

drawArus akan membangkitkan fungsi lajuArus() agar pixel yang digambar bergerak sesuai sumbu y secara terus menerus dan acak. Lebar dan panjang arus sebesar 2 *density* cukup untuk memberikan efek animasi pergerakan arus. Dan yang terakhir adalah pergerakan laju karang yang berfungsi sebagai dasar lautan. Mekanisme pergerakan batuan karang tersebut diimplementasikan pada kode java berikut ini :

```
1   public void lajuKarang() {
2     for (int i = 0; i < numKarang; i++) {
3     if (karangPosY[i] + 1 > canvasHeight - (karangSpeed * 2))
4   {
5     karangPosY[i] = 0;
6     karangPosX[i] = 0} else {
7     karangPosY[i] += karangSpeed; }}}
```

lajuKarang() memiliki perulangan terhadap jumlah karang yang dibutuhkan untuk menentukan pergerakan karang. Munculnya karang pada sumbu y diinisiasi pada koordinat sumbu y atas dan akan berakhir pada sumbu y bawah. Sedangkan percepatannya dihasilkan dari karangSpeed yang telah didefinisikan sebelumnya, dikali 2 density agar pergerakan terasa halus.

### 4.2.3 PERGERAKKAN OBJEK DALAM *GAME*

Implementasi pergerakkan objek dan aksi dalam *game* menggunakan potongan gambar sprite dari beberapa frame. Karakter utama dalam *game* ini adalah sebuah perahu yang melintasi lautan lepas untuk menghancurkan perompak yang telah menguasai lautan. Perahu laut ini memiliki proyektil dan nyawa agar bisa melakukan pertempuran dengan musuh, dalam hal ini adalah kapal perompak. Berikut ini adalah gambar 4.4 sprite perahu karakter utama dan perompak.

**Gambar 4.4 *Sprite* Perahu Karakter Utama dan Perompak**

*Sprite Sheet* Perahu diatas tidak ditampilkan pada layar dengan metode *pixel movement* seperti pembuatan game pada umumnya, mekanisme menggunakan metode tersebut mengambil sebagian gambar dari beberapa gambar dengan menempatkan koordinat pada sumbu x dan y sehingga yang akan muncul hanya berupa sebuah gambar karakter. Metode yang digunakan pada pembuatan game ini menggunakan metode *paint* yang dipanggil oleh setiap *layout* untuk menentukan apa yang ditampilkan pada layar berdasarkan file gambar.

Selanjutnya terdapat fungsi yang memiliki perintah untuk menggambar pada layar sesuai konsep animasi *game*. Setiap gambar perahu dipotong sesuai resolusinya menjadi tiga bagian sebagai objek yang akan ditampilkan pada layar. *Sprite sheet* tersebut merupakan gambar dengan ekstensi PNG yang mendukung transparansi. Dibawah ini adalah bagian baris kode java untuk menggambar objek perahu berupa gambar/bitmap ke dalam layar :

```
1   c.drawBitmap(perahu[currentPerahuState], perahuPosX,
2   perahuPosY, mBitmapPaint);
```

Karakter tersebut bisa bermanuver kekiri dan kekanan ketika tombol navigasi disentuh sehingga membutuhkan baris kode untuk masing-masing kondisi

(state). Pada saat state perahu kekiri maka akan ditampilkan gambar perahu paling kiri, dan jika state perahu kekanan maka akan ditampilkan gambar perahu paling kanan. Dibawah ini adalah bagian baris kode java untuk menangani pergerakan karakter tersebut :

```
1  if ((tx >= 10 && tx <= 30) && (ty >= 440 && ty <= 460)) {
2  perahuDx = -1;
3  currentPerahuState = StatePerahu.stateLeft;
4                       }
5  if ((tx >= 70 && tx <= 90) && (ty >= 440 && ty <= 460)) {
6  perahuDx = 1;
7  currentPerahuState = StatePerahu.stateRight;
```

*State* dengan perahu bergerak kekiri terjadi ketika koordinat tombol navigasi memenuhi kondisi dan tentunya harus diiringi dengan berpindahnya perahu dari koordinat sebelumnya menuju koordinat baru. Perpindahan kekiri misalnya, koordinat terhadap sumbu x akan dikurangi terus menerus sampai mencapai koordinat x baru dengan sumbu y tetap atau telah mencapai batas layar kiri. Penanganan logika tersebut diimplementasikan kedalam bagian baris kode java dibawah ini :

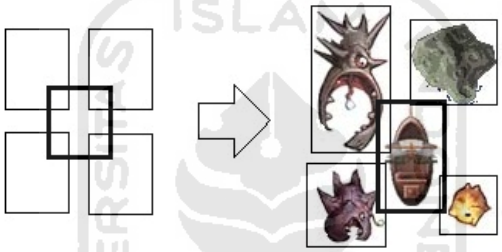
```
1   if (perahuPosY <= borderwidth || perahuPosY >= (height –
2   perahuHeight - panelHeight)) {
3                 perahuDy = 0;
4                 perahuPosY = perahuOldY;
5                 }
6   if (perahuPosX >= (width - borderwidth - perahuWidth)
7   || perahuPosX <= borderwidth) {
8                 perahuDx = 0;
9                 perahuPosX = perahuOldX;
10                }
```

Selanjutnya untuk pergerakkan kekanan, penanganannya menyesuaikan dengan koordinat terhadap sumbu yang berubah. Seperti jika pergerakkan karakter keatas maka terdapat perubahan terhadap sumbu y dengan koordinat sumbu x tetap.

Karakter perompak atau objek yang tidak dikendalikan oleh *gamer* juga memiliki metode yang hampir sama. Namun pergerakkan karakter tersebut dikendalikan oleh sebuah fungsi yang telah didefinisikan.

### 4.2.4 DETEKSI TUBRUKAN

Tubrukan yaitu sebuah kondisi ketika terdapat dua objek yang bersentuhan menghasilkan sebuah ledakan, akibat dari ledakan menyebabkan dua objek tersebut hancur. Dalam *game* ini, tubrukan terjadi ketika perahu karakter utama bersentuhan dengan monster yang harus dihindari selama berlayar. Pada implementasi pemrograman java, terdapat fungsi pengecekan secara berkala pada *game looping* untuk mendeteksi apakah terdapat koordinat sumbu X dan sumbu Y kedua objek yang berhimpitan. Apabila terdapat kondisi yang telah memenuhi syarat yang dimaksud, maka terjadilah ledakan dan objek yang bersinggungan hilang. Dibawah ini adalah konsep tubrukan menggunakan visualisasi gambar pada gambar 4.5.



**Gambar 4.5 Konsep Cek Tubrukan secara *Looping***

Berikut ini adalah bagian baris kode java terhadap pengecekan yang dipanggil pada *game looping* tersebut :

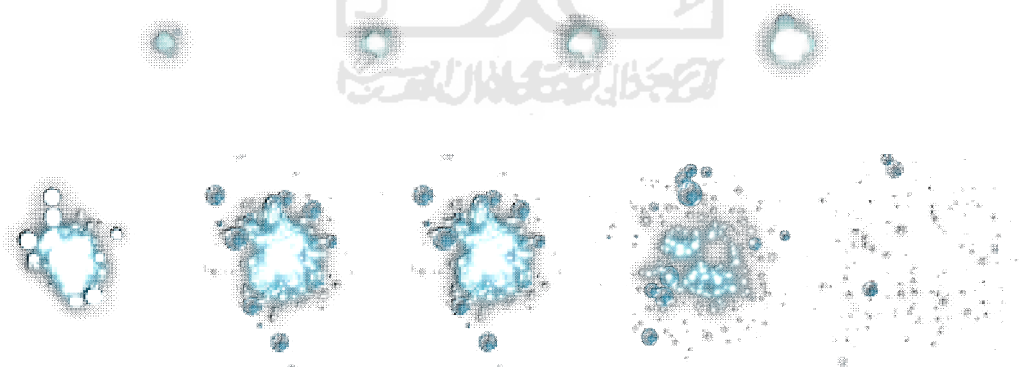
```
1   public void cekTubrukan() {
2   for (int i = 0; i < maxMonster; i++) {
4     if (monsterPosX[i] >= 0) {
5           if (monsterStat[i]
6                 & perahuPosX + perahuWidth > monsterPosX[i]
7                 & perahuPosX < monsterPosX[i] +
8   monsterWidth[monsterType[i]]
9                 & perahuPosY + perahuHeight > monsterPosY[i]
10                & perahuPosY < monsterPosY[i] +
11  monsterHeight[monsterType[i]]) {
12                hitPerahu();
13                hilangkanMonster(i);}
14                }
15          }
16    }
```

Objek monster yang harus dihindari oleh perahu karakter utama ditampilkan pada layar menggunakan potongan *sprite sheet* pada gambar 4.6.

**Gambar 4.6 *Sprite Sheet* Monster dalam *Game***

Monster diatas dibangkitkan secara acak sesuai jumlah *sprite sheet* kemudian digambar pada layar. Monster tersebut bergerak kearah perahu karakter utama untuk dihindari agar tidak terjadi tubrukan. Konsep yang hampir sama juga digunakan untuk pergerakan monster seperti arah gerak arus.

Pada saat kondisi tubrukan terpenuhi, maka terjadilah ledakan sebagai hasil dari hancurnya monster. Mekanisme ini terjadi dengan menghilangkan objek monster dan menggantinya sesuai koordinat monster menggunakan *sprite sheet* ledakan. Animasi ledakan ini terdiri dari beberapa *sprite sheet* yang terdapat pada gambar 4.7.



**Gambar 4.7 *Sprite Sheet* Ledakan dalam *Game***

*Sprite sheet* ledakan berurutan berganti dari efek pertama sampai terakhir kemudian menghilang. Efek ini akan menghasilkan transisi yang halus sehingga animasi ledakan bisa dilihat pada layar seketika tubrukan terjadi. Implementasi ledakan pada pemrograman java terlebih dahulu didefinisikan *sprite sheet* tersebut

kedalam bitmap. Berikut ini adalah bagian baris kode java pendefinisian bitmap ledakan dalam sebuah array.

```
1   int ids[] = new int[] {
2   R.drawable.ledak1, R.drawable.ledak2, R.drawable.ledak3,
3   R.drawable.ledak4, R.drawable.ledak5, R.drawable.ledak6,
4   R.drawable.ledak7, R.drawable.ledak8, R.drawable.ledak9,
5   R.drawable.ledak10};
6
7   animasiLedakan = new Bitmap[ledakanFrame + 1];
8   for     (int i = 0; i <= ledakanFrame; i++) {
9     animasiLedakan[i] = getImage(ids[i]);
10    }
```

animasiLedakan kemudian akan terpanggil ketika monster bersentuhan dengan perahu karakter utama jika sumbu y dan atau sumbu x monster dan perahu karakter utama bernilai sama. Proses menggambar sprite sheet tersebut diimplementasikan pada baris kode java dibawah ini :

```
1   c.drawBitmap(
2     animasiLedakan[ledakanFrame - monsterIndexFrame[i]],
3     monsterPosX[i] + (monsterWidth[monsterType[i]] –
4     animasiLedakanWidth)/ 2,
5     monsterPosY[i] + (monsterHeight[monsterType[i]] –
6     animasiLedakanHeight)/ 2,
7     mBitmapPaint);
```

Pergantian *sprite sheet* berhenti ketika jumlah *frame* yang telah didefinisikan sebelumnya tercapai dan mengakhiri rangkaian ledakan yang diimplementasikan pada *game*.

### 4.2.5 PANEL STATUS KARAKTER

Fitur yang tidak kalah penting sebelum *gameplay* adalah informasi apa saja yang layak ditampilkan dan diketahui *gamer*. Informasi itu dirangkum dalam panel status perahu karakter utama yang menampilkan skor, jumlah proyektil dan nyawa. Panel status ini adalah informasi yang ditampilkan ketika memenuhi kondisi tertentu. Misalnnya skor akan menampilkan nilai 100 ketika perahu berhasil menembak monster atau terjadi tubrukan. Kemudian informasi tentang jatah proyektil agar pengguna lebih bijak dalam menembak. Dan terakhir adalah nyawa perahu. Mekanisme berubahnya skor, jumlah proyektil dan nyawa telah dijelaskan

pada bab sebelumnya. Berikut adalah bagian baris kode java untuk menampilkan status yang dimaksud :

```
1   public void drawSkor(Canvas c) {
2     String s;
3     int my;
4     int height = getHeight();
5     mTextPaint.setARGB(255, 255, 255, 255);
6     mTextPaint.setTextSize(10.0f);
7
8     my = height - panelHeight + 12;
9     s = "Proyektil: " + currentProyektil + "/" + maxProyektil;
10    c.drawText(s, 90, my, mTextPaint);
11    c.drawRect(new Rect(150, my, 140 + (currentProyektil * 8),
12    my - 10),
13    mProyektilBarPaint);
14
15    my += 12;
16    s = "Nyawa: " + currentNyawa + "/" + maxNyawa;
17    c.drawText(s, 90, my, mTextPaint);
18    c.drawRect(new Rect(150, my, 140 +
19    (currentNyawa * 8), my - 10),mNyawaBarPaint);
20
21    my = 20;
22    mTextPaint.setARGB(255, 255, 0, 0);
23    mTextPaint.setTextSize(20.0f);
24    s = "Skor: " + skor;
25    c.drawText(s, 2, my, mTextPaint);
26
27    }
```

Jumlah nyawa dan proyektil akan ditampilkan diatas frame panel yang berupa gambar bitmap *spritesheet*. Sedangkan skor ditampilkan pada pojok kiri atas. Berikut adalah *sprite* panel status perahu pada gambar 4.8.

**Gambar 4.8 *Sprite* Panel Status untuk Informasi Perahu**

### 4.2.6 PERGERAKAN MUSUH

Implementasi pergerakan musuh yaitu, perompak dapat berpindah kekanan dan kekiri dan secara bersamaan menembakkan proyektil kearah karakter utama. Kondisi pergerakkan kanan dan kiri hampir sama dengan pergerakkan karakter utama, namun secara otomatis dibangkitkan oleh sebuah fungsi perompakStep. Dibawah ini adalah bagian baris kode java pergerakkan tersebut :

```
1  void perompakStep() {
2     for (int i = 0; i < maxPerompak; i++) {
3           if (perompakStat[i]) {
4  perompakDx[i] = perompakMoveStep[perompakMoveIndex[i]][0];
5  perompakDy[i] = perompakMoveStep[perompakMoveIndex[i]][1];
6           if (perompakDx[i] > 0) {
7  currentPerompakState[i] = StatePerahu.stateLeft;
8           } else if (perompakDx[i] < 0) {
9  currentPerompakState[i] = StatePerahu.stateRight;
10          } else {
11 currentPerompakState[i] = StatePerahu.stateCenter;
12                       }
13 if (perompakMoveStep[perompakMoveIndex[i]][2] == 1) {
14          byur();
15 }
16 perompakMoveIndex[i]++;
17 if (perompakMoveIndex[i] >= perompakStepCount) {
18          perompakMoveIndex[i] = 0;
19                       }
20                }
21          }
22    movePerompak();}
```

Perompak akan terus bergerak berdasarkan index sumbu x yang mengakibatkan perpindahan kekiri dan kekanan dan membangkitkan fungsi movePerompak untuk implementasi pergerakan berdasarkan lebar layar dan koordinat sumbu x. Perpindahan terhadap sumbu y tidak diimplementasikan karena perompak hanya akan berpindah kekanan dan kekiri. Sedangkan perompak akan menembakkan bom yaitu sebutan lain untuk 3 buah proyektil yang dimiliki perompak pada perahu karakter utama.

Bom ditembakkan secara otomatis ketika perompak berhenti untuk berbalik arah, pada saat itu fungsi drawBom akan dijalankan untuk menggambar *sprite* bom pada layar sesuai sumbu y. Baris kode java untuk fungsi tersebut adalah sebagai berikut :

```
1  public void drawBom(Canvas canvas) {
2  for (int j = 0; j < maxPerompak; j++) {
3     if (perompakStat[j]) {
4     for (int i = 0; i < bomLevel[j] * 10; i++) {
5           if (bomPosX[j][i] > 0) {bomPosY[j][i] += bomSpeed;
6                 if (bomPosY[j][i] < borderwidth
7                       | perahuHit(bomPosX[j][i],
8  bomPosY[j][i])) {bomPosX[j][i] = -1;
9     hitPerahu();
10    } else {canvas.drawBitmap
11 (bom, bomPosX[j][i],bomPosY[j][i], BitmapPaint);}}}}}
      }
```

*Canvas* digunakan untuk menampung bitmap dan fungsi-fungsi lain yang berhubungan dengan proses *drawing*. drawBom yang telah didefiniskan sebelumnya dengan bitmap bom *sprite* yaitu peluru yang ditembakkan perompak, bom tersebut akan digambar pada layar sesuai sumbu y. Bom seolah-oleh bergerak menuju kebawah dan akan dan akan menjalankan fungsi hitPerahu jika sumbu y bom sejalur dengan sumbu y perahu. Fungsi hitPerahu menyebabkan perahu karakter utama meledak dan kehilangan nyawa.

### 4.2.7 SKEMA PERTEMPURAN KARAKTER

Sistem pertempuran karakter utama secara garis besar adalah kemampuan menembak dan menghindari serangan. Pertempuran yang dijelaskan pada bab sebelumnya sebenarnya merupakan beberapa kombinasi dari tembak menembak dan tubrukan yang terjadi pada layar. Ketika karakter utama menembak, dengan kondisi jatah proyektil masih ada, *gamer* telah menyentuh tombol tembak sehingga membangkitkan fungsi tembak yang menggambar proyektil meluncur searah dengan sumbu y. Dibawah ini adalah implementasi baris kode java untuk menggambar bitmap ketika proyektil ditembakkan oleh perahu karakter utama :

```
1  public void drawProyektil(Canvas canvas) {
2  for (int i = 0; i < proyektilLevel * 10; i++) {
3     if (proyektilPosX[i] > 0) {
4           proyektilPosY[i] -= proyektilSpeed;
5     if (proyektilPosY[i] < borderwidth | monsterHit(i)|
6     perompakHit(proyektilPosX[i], proyektilPosY[i])) {
7     proyektilPosX[i] = -1;
8     } else {
9  canvas.drawBitmap(proyektil, proyektilPosX[i],
10 proyektilPosY[i],    mBitmapPaint);}}}}
```

Fungsi drawProyektil dijalankan dan menyebabkan sprite proyektil bergerak sesuai sumbu y dengan kecepatan konstan. Apabila sumbu y antara proyektil dan monster bernilai sama maka akan membangkitkan monsterHit, dan jika sumbu y antara proyektil dan perompak bernilai sama maka akan membangkitkan perompakHit. Masing-masing fungsi yang dibangkitkan tadi memiliki kondisi yang berbeda. Jika monsterHit menyebabkan skor bertambah 100 poin dan menyebabkan monster hancur, maka jika perompakHit akan menyebabkan skor bertambah 500 dan perompak hancur.

Ketika karakter utama terkena proyektil musuh atau monster yang bertubrukan, maka akan terjadi pengurangan nyawa yang dimiliki karakter utama. Setiap terjadi tubrukan baik itu proyektil mengenai perahu atau monster maka nyawa berkurang satu dari jatah yang disediakan. Jatah pada saat game dimulai sebanyak 9 dan akan bertambah sesuai bertambahnya skor. Baris kode java pengurangan nyawa karakter utama ketika menerima proyekti musuh sebagai berikut :

```
1  public void hitPerahu() {
2           crash.play();
3           perahuShield = maxNyawa;
4           currentNyawa--;
5  }
```

Selanjutnya proyektil karakter utama juga berkurang seiring dengan tembakan yang dilesakkan. Implementasi untuk mengurangi jatah proyektil pada kode java adalah dengan menyimpan variabel sementara (f) untuk dicek kondisinya setiap fungsi tembak() dijalankan. Jika jumlah proyektil masih diatas nol maka setiap menembak akan berkurang jumlah proyektilnya sampai nol. Apabila jumlah sama dengan nol, maka fungsi tembak() tidak dijalankan karena menggunakan percabangan while, dimana deskripsi program terletak disitu. Dibawah ini adalah bagian baris kode java pengurangan jatah proyektil saat menembak :

```
1  public void tembak() {
2     int n = 0, f = -1;
3     while (n < proyektilLevel * 10 && proyektilPosX[n] >= 0)
4           n++;
5           if (n < proyektilLevel * 10)
6           f = n;
7     if (f >= 0) {proyektilPosX[f] = perahuPosX +
8  ((perahuWidth - proyektilPerahuWidth) / 2);
9           proyektilPosY[f] = perahuPosY;
10          currentProyektil--;
11          blast.play();
12          }
13    }
```

Sebaliknya skor akan bertambah ketika musuh terkena proyektil dari karakter utama, baik itu monster ataupun musuh. Implementasi pada penambahan skor pada kode java terbagi menjadi dua; yaitu fungsi agar monster tersebut hilang dan skor bertambah. Apabila telah terdeteksi tembakan dari proyektil, maka fungsi

akan menjalankan hilangkanMonster sesuai indeksnya, dan sekaligus menambah skor 100 untuk perahu karakter utama. Dibawah ini adalah baris kode java penambahan skor saat monster terkena serangan :

```
1  if (monsterIndexFrame[i] < 0) {
2  hilangkanMonster(i);
4  skor += 100;
5  }
```

Sedangkan untuk skor pada saat proyektil mengenai perompak bernilai lebih besar dan diatur pada kode terpisah. Penambahan skor sebanyak 500 ditambahkan kedalam variabel skor setelah perompak terkena proyektil, sekaligus menjalankan animasi ledakan. Bagian baris kode java untuk penambahan skor ketika proyektil karakter utama mengenai perompak adalah sebagai berikut :

```
-------------------------------------------------------------
1  if (blx + proyektilPerahuWidth >
2  perompakPosX[i]& blx < perompakPosX[i] + perompakWidth
3  & bly + proyektilPerahuHeight > perompakPosY[i]
4  & bly < perompakPosY[i] + perompakHeight) {
5                          perompakStat[i] = false;
6                          perompakKillAnim[i] = true;
7                          skor += 500;
8                          visiblePerompak--;
9                          return true;}
```

Pada saat perompak terkena proyektil maka skor bertambah 500 poin sedangkan proyektil terkena monster maka hanya ditambahkan poin 100. Terdapat perbedaan karena tingkat kesulitan menembak monster lebih mudah daripada menembak perompak.

## 4.3 IMPLEMENTASI ANTARMUKA

Implementasi antarmukan merupakan hasil tampilan pembuatan *game* yang sesuai dengan rancangan antarmuka pada bab sebelumnya. Pada saat pembuatan selesai, maka dapat dihasilkan tampilan yang nyata hasil dari perancangan yang telah dibuat pada tahap sebelumnya.

### 4.3.1 MENU UTAMA

Menu utama adalah menu ketika aplikasi *game* ini dijalankan dan tampil pertama kali untuk merangkum operasi apa saja yang bisa dipilih *gamer*. Dalam menu utama ini terdapat 4 pilihan yang bisa dipilih oleh *gamer* yaitu; mulai main,

bantuan, tentang dan keluar. Tampilan menu utama aplikasi game ini dapat dilihat pada gambar 4.9.


**Gambar 4.9 Bitmap Menu Utama *Game* Kill The Pirates**

*Gamer* bisa langsung menyentuh menu yang tersedia sesuai pilihan yang diinginkan. Dan setiap menu mewakili operasi yang dimaksud.

### 4.3.2 MENU BANTUAN

Tampilan menu bantuan akan muncul setelah *gamer* memilih menu bantuan pada menu utama. Menu bantuan berisi tentang cara bermain dan informasi navigasi *gamer*. Selain itu tujuan penyelesaian *game* juga dijelaskan disini secara informatif. Tampilan menu bantuan dapat dilihat pada gambar 4.10.


**Gambar 4.10 Bitmap Menu Bantuan *Game* Kill The Pirates**

Menu bantuan dikelompokan menjadi 3 bagian informasi agar substansi diterima dengan mudah.*Gamer* dapat kembali ke menu utama dengan memilih tombol kembali pada bagian bawah menu bantuan.

### 4.3.3 MENU TENTANG

Tampilan menu tentang atau sering kita jumpai sebagai menu *about* muncul setelah gamer memilih menu tentang pada menu utama. Menu tentang menjelaskan kredit pengembang aplikasi *game* dan judul tugas akhir serta versi. Tampilan menu tentang dapat dilihat pada gambar 4.11.


**Gambar 4.11 Bitmap Menu Tentang *Game* Kill The Pirates**

*Gamer* dapat kembali ke menu utama dengan memilih tombol kembali pada bagian bawah menu tentang.

### 4.3.4 TAMPILAN *GAMEPLAY*

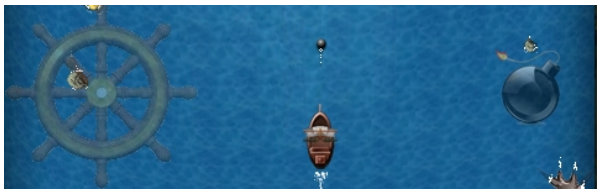
Tampilan *gameplay* adalah tampilan yang muncul setelah gamer memilih menu mulai main pada menu utama. Pada *gameplay*, *gamer* sudah masuk pada operasi permainan utama dan langsung dapat bermain. Pada tampilah ini terdapat perahu sebagai karakter utama yang harus menghancurkan musuh sesuai penjelasan *game* pada menu bantuan. Tampilan *gameplay* dapat dilihat pada gambar 4.12.


**Gambar 4.12 Tampilan *Gameplay* saat Aplikasi *Game* berjalan**

Pada tampilan *gameplay* ini gamer juga disuguhkan tombol navigasi buatan untuk menjalankan operasi pada perahu agar dapat bergerak dan menembak. Pada bagian bawah terdapat panel status perahu.

### 4.3.5 TAMPILAN NAVIGASI PERAHU

Tampilan ini membantu *gamer* yang menggunakan ponsel tanpa tombol navigasi fisik. Terdapat lima tombol yang terbagi menjadi dua bagian, yang pertama adalah tombol arah navigasi dan tombol menembak. Tampilan navigasi buatan dapat dilihat pada gambar 4.13.



**Gambar 4.13 Tampilan *Gameplay* Tombol Navigasi Buatan**

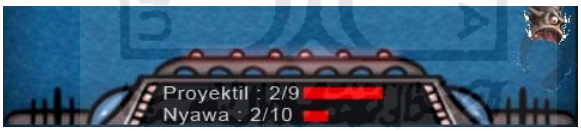
Dibawah ini adalah bagian baris kode java untuk menampilkan tombol navigasi buatan :

```
1   void drawPad(Canvas c) {
2           c.drawRect(40, 410, 60, 430, mPadPaint);
3           c.drawRect(40, 470, 60, 490, mPadPaint);
4           c.drawRect(10, 440, 30, 460, mPadPaint);
5           c.drawRect(70, 440, 90, 460, mPadPaint);
6           c.drawCircle(290, 450, 20, mPadPaint);}
```

Tombol tersebut transparan sehingga *gamer* tidak sepenuhnya kehilangan bagian yang harus dilihat dan tampilah *gameplay* tetap utuh secara penuh. Namun untuk merekam koordinat sentuhan yang terjadi pada tombol yang tersentuh, maka perlu diberikan kondisi agar koordinat navigasi berfungsi sebagaimana-mestinya. Koordinat tersebut mewakili fungsi gerakan dan tembak yang telah dijelaskan pada pergerakan objek dalam *game* sebelumnya.

### 4.3.6 TAMPILAN PANEL STATUS PERAHU

Tampilan panel status perahu terletak dibawah bagian *gameplay* dan akan terus menerus menampilkan informasi jumlah proyektil, skor dan nyawa. Tampilan utuh khusus untuk panel informasi ini adalah sprite tunggal yang diatasnya terdapat bilangan hasil dari operasi sistem skoring. Tampilan panel detail panel status perahu dapat dilihat pada gambar 4.14.




**Gambar 4.14 Tampilan Panel Status Perahu**

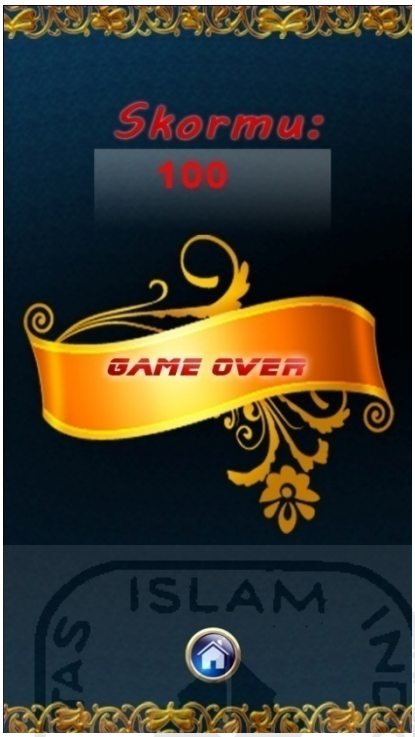
Untuk menampilkan skor dan status lainnya menggunakan sistem skoring yang dirancang pada bagian baris kode java dibawah :

```
1  s = "Proyektil:" + currentProyektil + "/" + MaxProyektil;
2  s = "Nyawa: " + currentNyawa + "/" + maxNyawa;
```

Baris kode diatas akan ditampilkan pada panel status perahu sesuai dengan bagiannya.

### 4.3.7 TAMPILAN *GAME OVER*

Tampilan *game over* otomatis akan muncul ketika *gamer* gagal menghancurkan musuh dan disebut kalah dalam pertempuran. Tampilan *game over* dapat dilihat pada gambar 4.15.




**Gambar 4.15 Tampilan Gameover**

Permainan berakhir ketika gamer melihat tampilan *game over* dan jika ingin melanjutkan maka harus mulai dari awal pertempuran lagi.

## 4.4 Analisa Kinerja *Platform* dan *Playtesting*

### 4.4.1 Pengujian *Game* pada Ponsel Android

Setelah pembuatan game pertempuran bajak laut menggunakan sprite pada sistem operasi android selesai menggunakan emulator dalam pengembangannya, maka game akan diimplementasikan pada ponsel yang menggunakan sistem operasi android sebagai implementasi akhir. Aplikasi game menghasilkan file APK (*Android Package Kit*) tunggal dan dapat diinstall pada ponsel secara langsung tanpa tambahan software lain. Berikut ini adalah foto pada gambar 4.16 ponsel android yang sedang menjalankan aplikasi game pertempuran bajak laut.


**Gambar 4.16 Galaxy Mini GT-5570 Menjalankan Kill The Pirates**

Ponsel tersebut adalah Samsung galaxy mini GT-5570 yang memiliki sistem operasi android froyo 2.2, frekuensi prosesor 600 Mhz dengan GPU Adreno 200, aplikasi game ini dapat berjalan dengan baik dengan resolusi layar 240x320 dan kualitas gambar baik. Aplikasi game sendiri terinstall dimemori internal ponsel dan berukuran 4,2 MB secara keseluruhan. Sedangkan foto pada gambar 4.17 berikut adalah pengujina pada ponsel *game*.


**Gambar 4.17 SE Xperia Play R800i Menjalankan Kill The Pirates**

Pengujian terakhir adalah pada ponsel gaming Sony Ericsson Xperia Play dengan fitur *high end* sebagai ponsel *gaming*. Ponsel ini memiliki sistem operasi android gingerbread 2.3.2 dengan frekuensi prosesor 1 Ghz dan GPU Adreno 205. Resolusi *High Definition* sampai dengan 854x480 pixel tampil sangat jernih dan tanpa cacat. Aplikasi *game* berjalan sangat lancar dan nyaman dimainkan karena layar yang lebar dan *hardware* yang mendukung.

Proses pengujian tidak menemukan kendala berarti karena teknologi yang digunakan adalah bahasa java sehingga *gamer* tidak perlu menambahkan software apapun ke ponsel androidnya. Dapat disimpulkan bahwa pada ponsel dengan dukungan hardware yang mendukung maka aplikasi *game* ini berjalan sangat baik dari segi grafis.

### 4.4.2 *Playtesting* Aplikasi *Game*

*Playtesting* dilakukan dengan metode kuesioner kepada mahasiswa dan siswa sekolah menengah atas untuk selanjutnya disebut responden. Responden dari kalangan mahasiswa sebanyak 10 orang dan dari siswa sekolah menengah atas sebanyak 32 orang. Responden tersebut mewakili *sample* lapisan masyarakat dibawah umur 30 tahun. Responden diberi kebebasan untuk memberikan masukan terhadap *game*. Kuesioner terdiri dari 16 pertanyaan yang terbagi dalam 5 kategori. Hasil *playtesting* disajikan dalam bentuk data yang dikategorikan klasifikasinya berdasarkan *usability, visual game, gameplay, behavior, miscellanous* menurut buku The Art of Game Design. (Jesse Schell,2008)

Skala kualitatif jawaban dari kuisioner bervariasi tergantung dari pertanyaan yang dimaksud. Penomoran berdasarkan urutan kualitas menandakan ukuran terburuk sampai terbaik. Misalkan untuk ukuran tidak dan ya diwakili dengan urutan dari terkecil sampai terbesar, yaitu angka 1 – 4. Urutan 1 menandakan nilai yang rendah dan 4 menandakan ukuran tinggi. Begitu juga untuk ukuran tidak setuju dan setuju yang mempunyai metode pengukuran yang sama.

*a. Usability*

Hasil tabel 4.1 menunjukan responden yang telah memberikan penomoran sesuai pertanyaan yang mewakili penggunaan aplikasi *game* dari sisi *usability*. *Usability* dicerminkan dari sisi *gamer* ketika menggunakan aplikasi *game* ini.

Beberapa pertanyaan yang mewakili tingkat penggunaan ini dirangkum dalam tabel hasil responden *usability*.

**Tabel 4.1 Hasil Responden Usability**

| No | Pertanyaan | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Apakan game ini mudah dioperasikan ? | 8% | 10% | 2% | 80% |
| 2. | Apakah tampilan dan desain pada game ini menarik dan interaktif ? | 5% | | 90% | 5% |
| 3. | Apakah game ini memberikan penjelasan cara bermain yang jelas dan akurat ? | 5% | 5% | 1% | 89% |

*b. Visual Game*

Hasil tabel 4.2 menunjukan responden yang telah memberikan penomoran sesuai pertanyaan yang mewakili penggunaan aplikasi *game* dari sisi desain dan efek animasi visual. Pertanyaan pada kategori ini merepresentasikan mengenai efek dan animasi yang ditampilkan pada *game*.

**Tabel 4.2 Hasil Responden Visual *Game***

| No | Pertanyaan | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Apakah game diproduksi dengan efek grafis yang kamu harapkan ? | | 25% | 70% | 5% |
| 2. | Apakah informasi yang ditampilkan dalam game jelas ? | | | 70% | 30% |
| 3. | Apakan transisi pergerakan objek terlihat nyata ? | 25% | 5% | 70% | |

*c. Gameplay*

Hasil tabel 4.3 menunjukan responden yang telah memberikan penomoran sesuai pertanyaan yang mewakili penggunaan aplikasi game dari tujuan dan alur cerita *game*. Daftar pertanyaan pada kategori ini mewakili bagaimana pemahaman gamer terhadap tujuan game dan alur permainan.

**Tabel 4.3 Hasil Responden *Gameplay Game***

| No | Pertanyaan | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Apakah tujuan permainan jelas ? | | 10% | | 90% |

| | | | | | |
|---|---|---|---|---|---|
| 2. | Apakah game tersebut menarik dari sisi cerita yang ditawarkan ? | | 2% | 95% | 3% |
| 3. | Apakah game tersebut membosankan ? | | 30% | 70% | |
| 4. | Apakah sistem pertempuran pada game sesuai dengan tema game ? | 10% | | 80% | 10% |

*d. Behavior*

Hasil tabel 4.4 menunjukan responden yang telah memberikan penomoran sesuai pertanyaan yang mewakili penggunaan aplikasi *game* dari sisi kebiasaan yang umum terdapat pada sebuah *game*. Kebiasaan tersebut dipengaruhi oleh latar belakang responden terhadap *game*. Pertanyaan pada kategori ini untuk mengukur mudah tidaknya *game* ini diterima oleh masyarakat.

**Tabel 4.4 Hasil Responden *Behavior Game***

| No | Pertanyaan | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Apakah game tersebut cocok dimainkan didalam ruangan ? | | | 80% | 20% |
| 2. | Apakah game tersebut cocok dimainkan diluar ruangan ? | 72% | 3% | 10% | 15% |
| 3. | Apakah game tersebut cocok dimainkan saat menunggu/mengisi waktu luang ? | | | | 100% |
| 4. | Apakah game tersebut kelihatan asing atau aneh ? | 100% | | | |

*e. Miscellaneous*

Hasil tabel 4.4 menunjukan responden yang telah memberikan penomoran sesuai pertanyaan yang mewakili penggunaan aplikasi game dari sisi efek pendukung *game*. Aspek pendukung dirumuskan pada pertanyaan kategori ini untuk melihat kesan responden terhadap suara yang dihasilkan *game*.

**Tabel 4.5 Hasil Responden *Miscellaneous Game***

| No | Pertanyaan | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 1. | Apakah efek suara yang dihasilkan seirama dengan animasi game ? | | | | 100% |

| | | | | | |
|---|---|---|---|---|---|
| 2. | Apakah jenis musik pendukung seirama dengan tema game ? | | | 90% | 10% |

Responden memberikan penomoran terhadap kualitas yang telah ditentukan dalam kuisioner yang telah dikategorikan pada tabel sebelumnya. Untuk mengetahui umpan balik berupa masukan yang tidak tersedia dalam kuisioner, maka disusun masukan yang berhubungan dengan pengembangan game selanjutnya yang disusun dalam kategori keluhan dalam penggunaan *game*.

Dibawah ini adalah tabel 4.6 keluhan terhadap aplikasi *game* kill the pirates yang diformulasikan dari daftar jawaban pada saat *playtesting*.

**Tabel 4.6 Table Hasil Tes *Playtesting* Masukan Responden**

| No | Masalah | Deskripsi | Solusi | Komentar |
|---|---|---|---|---|
| 1 | Fungsi Navigasi | Gamer tidak terbiasa menggunakan interface layar sentuh | Tombol navigasi diberi arah dan penjelasan mendetail | Gamer yang sudah terbiasa seharusnya mampu menalar tombol apa yang harus diketahui |
| 2 | Sistem Skoring | Gamer tidak mengetahui kapan skor bertambah, perbedaan menembak monster dan perompak. | Penggunaan game dalam jangka waktu lama menambah kebiasaan dan akan tahu sendiri. | Biasakan main |
| 3 | Jatah nyawa dan proyektil bertambah, tapi nyawa berkurang | Gamer merasa asing dengan sistem nyawa dan proyektil | Penggunaan game dalam jangka waktu lama menambah kebiasaan dan akan tahu sendiri. | Biasakan main |
| 4 | Efek Grafis | Animasi tidak sehalus game 3D dan kasar | Animasi menggunakan bitmap sprite, penggunaan | Mengguna kan library OPENGL ES untuk tahap |

| | | | | |
|---|---|---|---|---|
| | | | efek 3D terlalu berat saat pengembangan | pengembangan selanjutnya |

Pembuatan game ini pada hasil akhirnya memiliki kelebihan dan kekurangan yang bisa dilihat dari performa, animasi dan tingkat ergonomis, pengelompokan ini diharapkan bisa dikembangkan pada tahap selanjutnya. Pada bagian ini terdapat intisari yang mewakili evaluasi tahap akhir terhadap pengujian aplikasi game.

A. Kelebihan *Game*

Kelebihan dari aplikasi game kill the pirates ini adalah :

1. Perbandingan pada *game* menggunakan *sprite* lainnya terletak pada panel informasi yang lengkap mengenai status perahu dan skoring yang jelas serta *realtime*.
2. Sistem pertempuran yang mudah dipahami oleh semua kalangan menyebabkan *game* ini mudah dimainkan.
3. Menu bantuan yang informatif membuat gamer mengetahui cara permainan berlangsung.
4. Efek animasi dan suara ketika game berlangsung seperti suara proyektil, ledakan membuat gamer tidak bosan dan membuat suasana lebih hidup.

B. Kekurangan *Game*

Kekurangan dari aplikasi game kill the pirates ini adalah :

1. Gamer harus terbiasa menggunakan ponsel layar sentuh dengan navigasi buatan. Sehingga pada permainan awal membutuhkan kebiasaan. Selain itu gamer harus mampu menempatkan jarinya sehingga tidak sulit untuk memainkan game karena layar tertutup jari yang melakukan aksi.
2. Alur cerita sangat sederhana karena dilakukan pada saat pengembangan sistem sehingga game bisa saja berakhir dengan cepat selain itu tidak memiliki informasi level sehingga gamer tidak mengetahui tingkatan kecepatan saat permaian berlangsung.
3. Teknologi animasi pada pengembangan android seperti OPEN GL ES tidak diimplementasikan sehingga efek animasi kurang halus.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Dari penelitian dan pembahasan yang telah dilakukan maka dapat diambil kesimpulan dari Pembuatan Aplikasi Game Pertempuran Bajak Laut Menggunakan *Independent Graphic Objek* (Sprite) adalah sebagai berikut :

1. Aplikasi game bergenre *action battle* berjudul kill the pirates memberikan porsi tersendiri untuk gamer ponsel android sebagai game pertempuran.
2. Hasil penelitian ini adalah aplikasi *game* kill the pirates yang menggunakan bahasa java dan objek 2D dan merupakan implementasi sebenarnya dari konsep pemrograman tanpa *game engine*.
3. Animasi menggunakan *sprite* mampu menghasilkan aplikasi *game* yang berjalan lancar pada emulator maupun ponsel android.

## 5.2 Saran

Berdasarkan hasil pembahasan dan kesimpulan yang ada maka saran-saran untuk Aplikasi Game Pertempuran Bajak Laut Menggunakan Independent Graphic Objek (Sprite) antara lain :

1. Pengembangan *game* selanjutnya mampu dimainkan oleh banyak pemain (*multiplayer*) menggunakan bluetooth.
2. Efek animasi game terlihat lebih halus jika menggunakan teknologi 3D OPEN-GL ES untuk setiap objek yang ditampilkan pada layar.

# DAFTAR PUSTAKA


Jesse Schell. 2008. *The Art of Game Design: A book of lenses*. Massachusetts : Morgan Kauffmann

Nilwan, Agustinus. 1996. *Pemrograman Animasi dan Game Profesional*. Jakarta : Elex Media Komputindo

Sibero, Ivan C. 2009. *Langkah Mudah Membuat Game 3D*. Yogyakarta : Mediakom

Zechner, Mario. 2011. *Beginning Android Games*. New York : Apress

Anonim.2012.*Pengertian Sistem Operasi Android* tersedia pada http://id.wikipedia.org/wiki/Android_(sistem_operasi)

Anonim. 2011.*Android Game Development Tutorial* tersedia pada http://p-xr.com/android-tutorial-how-to-paint-animate-loop-and-remove-a-sprite/

Flockton, Stephen. 2009. *2D Sprite Animation in Android* tersedia pada http://www.droidnova.com/2d-sprite-animation-in-android,471.html

Guerrero, Maximo. 2009. *Android Game Development using Sprite* tersedia pada http://warriormill.com/2009/10/adroid-game-development-part-1-gameloop-sprites/

Knudsen,jonathan.2003. *Creating 2D Action Games with the Game API* tersedia pada http://developers.sun.com/mobility/midp/articles/game/

Grace, lindsay. 2005. *Game Type and Game Genre.* Lecture Presentation.

Spring. 2005. *AI Game Programming Wisdom 2*. Ref: Chenney, CS679 lectures