BAB 3
ANALISIS DAN PERANCANGAN

3.1. Analisis Sistem

Analisis Analisis sistem merupakan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana, sehingga struktur logisnya menjadi lebih jelas. Pada analisis sistem ini akan mencakup analisis masalah yang berhubungan dengan game yang akan di bangun, mulai dari analisis terhadap game sejenis serta analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional.

3.1.1 Analisis Masalah

Game merupakan salah satu produk teknologi informasi yang cukup digemari saat ini, para pengelola industri berlomba-lomba untuk menciptakan *game* lebih menarik untuk para pemainya dengan cara meningkatkan kelincahan intelektual pada tingkat *gameplay*. Sehingga *game* tidak sekedar permainan untuk mengisi waktu luang melainkan cara untuk meningkatkan kreatifitas dan intelektual para penggunanya.

Game side scrolling adalah salah satu *genre game* dengan sudut pandang kamera seolah-olah di samping karakter, dan mengikuti pergerakan pemain yang pada umumnya bergerak dari sisi kiri menuju sisi kanan layar untuk mencapai target atau lokasi yang telah ditentukan. Alasan menerapkan side scrolling adventure game pada game yang akan dibangun adalah, karena game sejenis ini paling cocok untuk diterapkan pada umur 6 tahun ke atas, karena kontrol permainan yang tidak terlalu rumit dalam memainkannya [1].

Game tentu akan lebih menarik jika di dalamnya ada sebuah tantangan yang cukup sulit dengan memberikan sebuah kecerdasan buatan pada musuh yang berada dalam *game*. AI adalah sebuah kecerdasan buatan yang di tanamkan pada mesin atau komputer, agar seolah-olah menjadi lebih pintar dan bisa berfikir sendiri layaknya manusia. AI dibangun dari sebuah algoritma, dan algoritma yang di terapkan pada game ini adalah algoritma Djikstra. Algoritma ini merupakan

algoritma pencarian buta untuk mencari solusi dengan jalur terpendek. Algoritma ini nantinya akan di terapkan pada musuh yang berjalan dalam *game* in

3.1.1 Analisis *game* yang sejenis

Analisis *game* sejenis dilakukan untuk memberikan gambaran sistem agar lebih mudah dalam pembangunan dan sebagai pembanding untuk kesempurnaan *game* yang akan dicapai.

3.1.2.1 Game Mario Bros

Mario Bros adalah suatu permainan platform yang dikembangkan dan diterbitkan oleh Nintendo pada akhir 1985 untuk konsol Nintendo Entertainment System. Permainan ini membawa pengaruh yang besar pada perkembangan dunia hiburan rumahan dan merupakan salah satu permainan terlaris dengan penjualan lebih dari 40 juta salinan hingga saat ini. Dengan latar permainan yang cerah dan alur cerita yang berkembang, Super Mario Bros. berhasil mengubah wajah industri permainan video. Meskipun sering disalah persepsikan sebagai permainan platform bergulir (scrolling) pertama (yang pada kenyataannya paling tidak ada setengah lusin permainan serupa yang telah muncul sebelumnya), Super Mario Bros. bisa dianggap sebagai pelopor jenis permainan tersebut unruk media permainan konsol.

A. Pengenalan

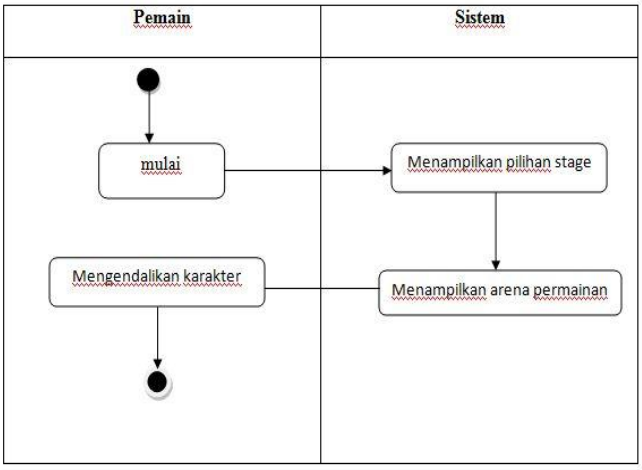
Game Mario bros sangat popular di Indonesia pada tahun 1990. *Game* ini di ciptakan oleh Shigeru Miyamoto dan merupakan *game* yang bisa kita mainkan pada consule seperti Nontendo dan Spica. Permainan Ini dikembangkan dan diterbitkan oleh Nintendo pada akhir 1985, tampilan game mario bros dapat dilihat pada gambar 3.1.



Gambar 3.1 *Game* Mario Bross

B. *Game*play

Super mario bros adalah *game* bertipe platform, pemain akan mendapati dirinya sebagai mario atau saudaranya luigi, dalam petualangan menyelamakan putri dari kura-kura raksasa bernama Bowser. Meski menyelamatkan putri terdengar cukup klise dan tidak original, untuk *game* semacam ini cerita tidaklah terlalu penting. dengan setting di mushrom kingdom, super mario bros memiliki banyak musuh yang inovatif dan menantang, alur permainan game mario bros dapat dilihat pada gambar 3.2.



Gambar 3.2 Alur permainan pada *game* mario bros

C. Skenario *Game*

Skenario Pada *game* ini pemain berperan sebagai Mario dan pemain kedua berperan sebagai kakak Mario yaitu Luigi. Tujuan utamanya dan menyelamatkan Princess Peach yang ditahan oleh Raja yang menyerupai buaya bernama Koppa. Mario memiliki kemampuan menghindar sekaligus mengalahkan musuhnya dengan cara melompat di atas musuh-musuhnya.

Musuh Mario Bros pada *game* ini adalah jamur pengkhianat, yang dikenal sebagai Goombas, dan tentara kura-kura dikenal sebagai Koopa Troopas. Jika menemukan box atau kotak yang memiliki tanda tanya kita dapat menyundulnya. Saat menyundulnya akan menemuan jamur ajaib. Mario juga dapat memperoleh 1-up melalui menemukan 1-up jamur dan dengan mengumpulkan 100 koin. Semua musuh memiliki satu persamaan dasar, loncat dan injak kepala mereka untuk membunuhnya.

Mario memiliki beragam kekuatan yang terus bertambah sampai sekarang. Mario biasanya memiliki tiga buah kekuatan dasar seperti, Jamur untuk membesar, bunga api untuk membuat Mario menembakkan api dan bintang yang membuat Mario tidak bisa dilukai oleh serangan apapun.

Permainan terdiri dari 4 sub dunia, yaitu:

1. Sub-dunia diatas tanah.
2. Dalam sebuah penjara bawah tanah atau air.
3. Serangakian platform tergantung tinggi di langit.
4. Benteng atau puri

Semua dunia terjadi di siang hari kecuali dunia ke 3 dan ke 6. Pada akhir setiap tingkat benteng, Mario berhadapan dengan Raja Koopa, melintasi sebuah jembatan di atas kolam lava. Di akhir dunia (dunia 6-8), Raja Koopa melemparkan palu serta sesekali semburan api napas. Raja Koopa dapat dikalahkan dalam salah satu dari dua cara,yaitu:

1. Dengan menyentuh kapak di ujung jembatan (sehingga menjatuhkan Raja Koopa ke dalam lava

2. Seperti Fiery Mario, melemparkan bola api pada dia untuk mengalahkan dia secara langsung dan mengungkapkan apa musuh menyamar.

Setelah menyelesaikan permainan, pemain diberi pilihan untuk memulai permainan lagi di '*Hard*' Mode, dimana semua Goombas digantikan oleh Buzzy Beetles (Koopa Troopa yaitu musuh yang tidak dapat dibunuh oleh bola api), dan semua musuh berjalan cepat. Selain itu lift juga berjalan lebih cepat 60% dari Mario pada mode biasa.

Kelebihan dari *game* Mario bros adalah :

1. *Game* ini adalah *game* yang sangat menarik dimasanya.
2. *Game* ini pada layar dan background mempunyai gerakan yang khas yaitu *double screen*, pada era itu umumnya hanya mempunyai single *screen*, sehinga *gameplay* pada umumnya berpindah *screen*, dan jika ada *game* yang memiliki *screen scrolling* yang sama dengan metode Mario bros.
3. *Game* ini merupakan *game* dengn penjualan terbanyak versi *Guiness Word of Record*.

Kekurangan dari game Mario bros ini adalah:

1. *Game* ini pada saat ini terlihat kurang begitu menarik karena peralihan gme semakin canggih pada saat ini.
2. *Game* Mario bros layar hanya bisa bergeser dari kiri ke kanan. *Player* utama dapat berjalan dan melompat serta dapat menginjak musuhnya
3. Musuh dalam *game* Mario bros hanya bisa berjalan secara *random* dan tidak dapat mengejar *player* utama.
4. Kualitas suara yang masih berformat midi

3.1.2 Analisis *game* yang dibangun

Ada berbagai macam *genre game* yang telah dikembangkan, maka dari itu diperlukan pemilihan *genre* yang sesuai dalam pembangunan aplikasi *game* ini. Pada pembangunan aplikasi *game* ini, Garuda adalah hewan yang terbang maka dari itu pemilihan *genre shooter* sejalan dengan hal itu. Untuk pemilihan *genre*

yang menampilkan luas kota adalah mutlak diperlukan *genre side-scrolling*. Kedua *genre* tersebut kemudian dikombinasikan sehingga menghasilkan sebuah *genre* baru yaitu *hybrid shooter side-scrolling. Genre* baru inilah yang akan diterapkan dalam pembangunan aplikasi *game* ini ini lah salah satu kelebihan dari *game destroyer* garuda.

Alur permainan *game* yang dibangun akan diterapkan dalam 2 mode yaitu mode pesawat dan mode robot pada tokoh utamanya yaitu Garuda F01. Hampir mirip dengan konsep Transformer yang membedakan mode karakter utama dari segi kemampuan dan kelebihan ketika dalam suatu mode tertentu. Ketika Garuda F01 dalam mode pesawat ia akan memiliki kemampuan layaknya pesawat yaitu seperti terbang dan memburu musuh yang berada di udara. Tetapi mode pesawat memiliki kelemahan yaitu ia tidak bisa melawan musuh yang bertebaran di darat karena keterbatasan kemampuan senjatanya yang hanya dapat menembak sasaran terbang. Kelemahan itu kemudian ditutupi dengan mode robot yang mana pada mode ini robot dapat berjalan dan berlari di daratan dan memiliki kemampuan menghancurkan musuh dengan kontak fisik. Adapun kelemahan pada mode robot adalah ia tidak bisa melawan serangan udara dari musuh terbang. Selain itu, karena robot ini mempunyai kode etik untuk tidak merusak bangunan kota, maka harus kembali ke mode pesawat untuk bermain aman.

Game akan menjadi menarik dengan menambah segmentasi pengguna, maka unsur pengenalan kota-kota besar di indonesia yang diterapkan akan dipadukan dengan fiksi ilmiah. Fiksi ilmiah akan menjadikan alur cerita *game* ini bermula di masa depan ketika Indonesia menjadi negara adikuasa dan segalanya telah serba modern. Indonesia memiliki skuadron pesawat garuda yang terkenal dengan kesaktiannya. Kota-kota terkenal di Indonesia memiliki wajah baru di masa depan dan hal itu ditampilkan dengan serba metalnya komponen infrastruktur kota dari mulai jalanan hingga monumen yang ada di sana. Pesawat garuda andalan tidak seperti pesawat biasa pada umumnya, tetapi mereka juga bisa berubah menjadi robot seperti pada Transformers.

3.1.3.1 *Storyline*

Tahun 2500 dunia sudah berubah, hampir semua negara menggunakan robot sebagai penggerak roda ekonominya termasuk juga di Indonesia. Di masa ini bumi mengalami penyerangan dari mahluk asing yang berniat menguasai bumi. Satu demi satu negara di hancurkan dan di kuasai oleh mahluk asing, hanya satu negara saja yang masih bisa mempertahankan negaranya dari serangan mahluk asing. Indonesia adalah satu-satunya negara yang masih mampu melindungi teritorial negaranya dari serangan mahluk asing karena memiliki alat militer berupa pesawat yang bisa berubah bentuk menjadi robot yaitu GARUDA F01.

GARUDA F01 adalah satu-satunya mesin tempur milik indonesia yang dapat menyaingi kemampuan mesin-mesin yang di miliki oleh para mahluk asing, dengan kemampuan khusus yang di miliki GARUDA F01 yaitu berubah menjadi robot maka mahluk asing pun menjadi sedikit kesulitan dalam menguasai Indonesia.

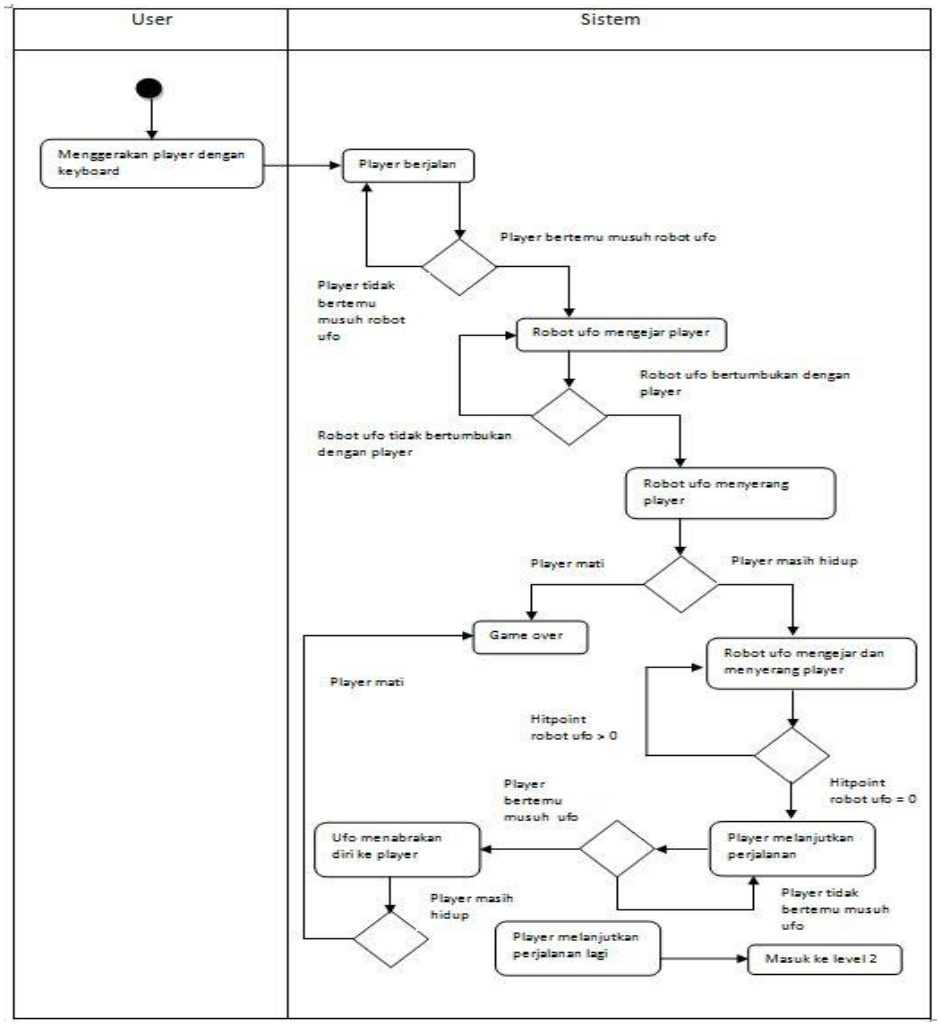
Indoneisa merencanakan untuk melakukan penyerangan diam-diam ke markas besar para mahluk asing untuk mengahancurkan kapal induknya. Karena hanya melakukan penyerangan sendirian, maka GARUDA F01 harus benar-benar cerdik dalam bertindak agar tidak jatuh tertembak oleh musuh.

3.1.3.2 Analisis Tingkat Kesulitan

Level *game* mempunyai tingkat kesulitan yang berbeda-beda, di level 1 pemain akan menghadapi musuh yang hanya bisa mengejar dan menyarang dari jarak dekat serta musuh terbang yang hanya bisa bergerak secara horizontal. Berbeda dengan level 2 musuh di level ini selain bisa mengejar juga punya kemampuan untuk menembakkan meriam dari jarak yang cukup jauh sehingga sedikit menyulitkan jika pemain sedang dalam keadaan terdesak, serta musuh terbang yang hanya mampu bergerak secara horizontal. Dan di level 3 musuh yang ada merupakan kombinasi dari musuh-musuh dari level 1 dan 2 di tambah bos besar yang mempunyai jurus kombinasi dari seluruh musuh yang ada dalam *game,* jadi bisa di simpulkan level ini lah yang tersulit.

3.1.3.3 *Gameplay*

Pemain dalam *game* ini di haruskan melewati seluruh rintangan yang ada dan termasuk mengalahkan semua musuh yang ada. Di level 1 pemain harus menghadapi serangan mahluk asing yang mencoba menyerang jakarta, di sini pemain harus bisa mengalahkan semua musuh yang ada sehingga pemain bisa melanjutkn ke level 2. Gambar activity diagram level 1 dapat di lihat pada gambar 3.3.



Gambar 3.3 Activity diagram level 1 Destroyer Garuda

Di level 2 pemain harus melakukan pembersihan basecamp mahluk asing yang berada di daerah bandung, di sini nanti pemain pun harus mengalahkan semua musuh yang ada sehingga pemain bisa melanjutkan ke level 3 . Dan di

level 3 pemain harus melakukan penyerangan ke markas utama musuh yang berada di samudra hindia, di sini nanti pemain harus menghadapi bos utama dari para mahluk asing. Perbedaan tiap levelnya adalah di tingkat kesulitannya, semakin tinggi levelnya maka musuh yang di hadapi akan semakin sulit.

3.1.3.4 Analisis *Scoring*

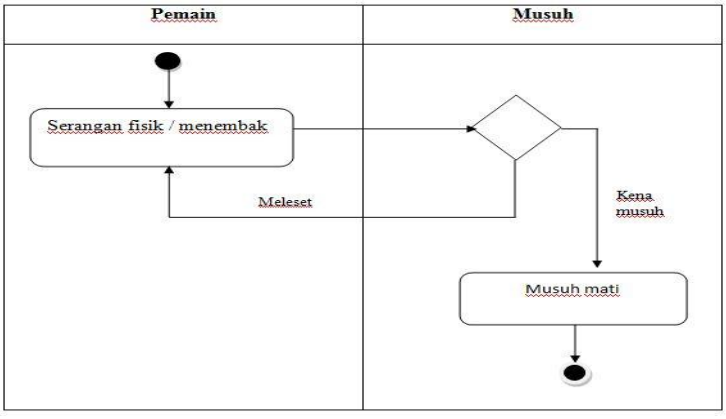
Berikut cara memperoleh nilai atau *score* pada *game* destroyer garuda yaitu :

1. Garuda mode pesawat harus menembak ufo yang terbang di atas kota. Setiap menghancurkan ufo akan mendapatkan score sebanyak 30 point.
2. Garuda mode robot harus melawan robot ufo, shooter ufo, bos hingga hancur maka score akan bertambah sebanyak 30 point setiap menghancurkan musuh.

3.1.3 Analisis Metode *Collision Detection*

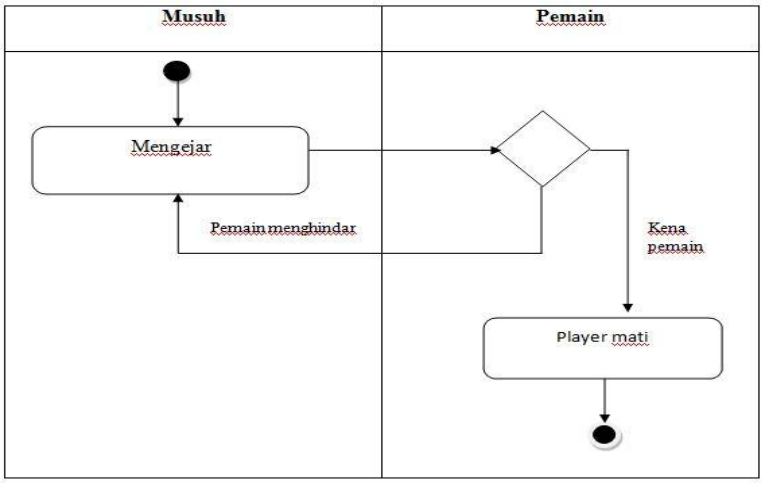
Aplikasi *game* destroyer garuda dapat menjadi lebih menarik, apabila diterapkan metode *collision detection* yang berupa metode dimana yang bersinggungan merupakan objek secara langsung, baik itu *sprite* dengan *sprite* maupun *sprite* dengan objek lain.

Metode yang akan pada aplikasi *game* destroyer garuda diterapkan pada serangan fisik dan peluru yang ditembakan oleh pemain untuk membunuh setiap musuh yang ada adalah metode *collision* antara objek dengan objek yang terjadi apabila objek tersebut saling bersentuhan pada titik yang sama. Diagram Activity tembakan pemain menyerang musuh ditunjukkan pada gambar 3.4.



Gambar 3.4 Diagram Activity Tembakan Pemain Menyerang Musuh

Metode yang akan diterapkan pada musuh yang menyerang pemain adalah metode *collision* antara objek dengan objek yang terjadi apabila objek tersebut saling bersentuhan pada titik yang sama. Diagram Activity metode *collision* musuh menyerang pemain ditunjukkan pada gambar 3.5.

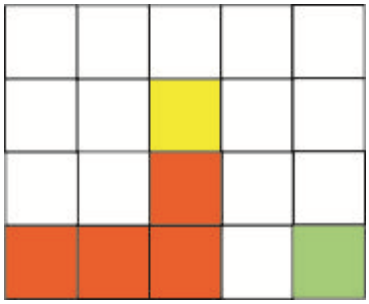


Gambar 3.5 Diagram Activity Metode Collision Musuh Menyerang Pemain

3.1.4 Analisis Algoritma Djikstra

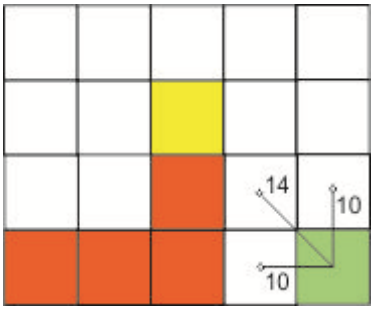
Game tentu akan lebih menarik jika di dalamnya ada sebuah tantangan yang cukup sulit dengan memberikan sebuah kecerdasan buatan pada musuh yang berada dalam *game*. AI adalah sebuah kecerdasan buatan yang di tanamkan pada mesin atau komputer, agar seolah-olah menjadi lebih pintar dan bisa berfikir sendiri layaknya manusia. AI yang akan di gunakan dalam *game* destroyer garuda ini adalah algoritma Dijkstra, yaitu algoritma pencarian buta untuk mencari jalur terpendek yang akan diterapkan pada musuh dalam melakukan perburuan terhadap pemain

Algoritma Dijkstra melakukan sejumlah langkah dengan konsep greedy, konsep greedy ini dalam algoritma Djikstra menyatakan bahwa setiap langkah yang mempunyai sisi bobot minimum akan dimasukkan ke dalam himpunan solusi. Algoritma ini melakukan pencarian secara buta untuk mencari solusi terpendek. Untuk lebih jelasnya bisa di lihat pada contoh gambar 3.6.



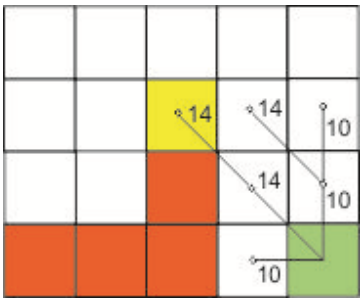
Gambar 3.6 Algoritma djikstra 1

Pada gambar di 3.5 , node berwarna orange merupakan tembok dalam *game,* dan *node* berwarna kuning adalah *player* utama sedangkan *node* berwarna hijau merupakan musuh seperti di gambarkan pada gambar 3.7.



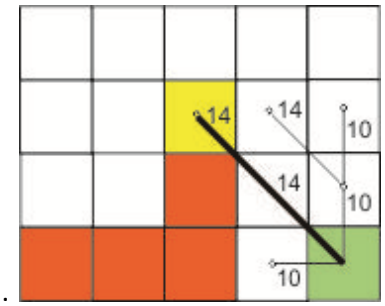
Gambar 3.7 Algoritma djikstra 2

Node hijau akan melakukan pencarian secara horizontal, vertikal dan diagonal. Pertama pencarian dilakukan ke arah kiri secara horizontal dengan nilai *cost* 10, lalu pencarian dilanjutkan ke arah atas secara vertikal dengan nilai *cost* 10, dan terakhir pencarian dilakukan ke arah kiri atas secara diagonal dengan nilai *cost* 14 seperti di gambarkan pada gambar 3.8.



Gambar 3.8 Algoritma djikstra 3

Node vertikal atas dengan *cost* 10 pencarian dilanjutkan ke arah atas secara vertikal dengan *cost* 10 sehingga sekarang total *cost* menjadi 20, lalu pencarian juga dilakukan secara diagonal sehingga total *cost* menjadi 24. Node diagonal kiri atas dengan *cost* 14 pencarian dilakukan ke arah kiri atas secara diagonal dengan *cost* 14 sehingga sekarang total *cost* menjadi 28 seperti digambarkan pada gambar 3.9.



Gambar 3.9 Algoritma djikstra 4

Node berwarna hijau menuju node berwarna kuning didapatkan jalur terpendek menggunakan algoritma djikstra, yaitu sebagai berikut:

1. *Node* berwarna hijau bergerak ke kiri atas secara diagonal dengan *cost* 14.
2. *Node* kiri atas dengan nilai *cost* 14, bergerak lagi ke kiri atas secara diagonal dengan *cost* 14 maka sampai lah ke *node* kuning dengan total *cost* 28.

3.1.5 Analisis Kebutuhan *Non*-Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

Analisis kebutuhan sistem non fungsional ini menjelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras dan analisis pengguna.

3.1.6.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras atau *hardware* merupakan salah satu hal yang penting karena tanpa *hardware* yang memenuhi syarat, program yang akan dibuat tidak akan dapat berjalan. Berikut spesifikasi standar perangkat keras yang dapat dipergunakan untuk membangun aplikasi *game* ini dan spesifikasi perangkat keras yang dapat dipergunakan untuk menjalankan alplikasi *game* ini yaitu :

A. Spesifikasi standar perangkat keras untuk pembangunan aplikasi *game* tersebut adalah :
   1. Processor 2.6 Ghz
   2. RAM 4 GB
   3. VGA 512 MB
   4. *Harddisk* 500 GB
   5. Monitor
   6. *Mouse, keyboard dan speaker.*

B. Spesifikasi standar perangkat keras untuk dapat menjalankan aplikasi *game* tersebut adalah :
   1. Processor 1.8 Ghz
   2. RAM 1 GB
   3. VGA 512 MB
   4. *Harddisk* 40 GB
   5. Monitor
   6. *Mouse, keyboard dan speaker.*

Dari hasil pengamatan perangkat lunak, terdapat beberapa hal penting yang harus diperhatikan agar sistem yang dirancang dapat berjalan dengan baik dan sesuai dengan yang diharapkan. Pengguna diharapkan dapat memenuhi kebutuhan perangkat lunak yang telah direkomendasikan agar aplikasi dapat berjalan dengan baik.

3.1.6.2 *Analisis Kebutuhan Perangkat lunak*

Perangkat lunak atau *software* merupakan hal yang terpenting dalam mendukung kinerja sebuah sistem. Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar dapat salung berinteraksi diantara keduanya.

Perangkat lunak yang dibutuhkan untuk membangun aplikasi *game* destroyer garuda ini adalah sebagai berikut :

1. Sistem Operasi Windows XP, Windows 7 dan sistem operasi lainnya.
2. Greenfoot 2.0 ke atas
3. Java JDK 1.6 ke atas

Sedangkan perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *game* destroyer garuda ini adalah sebagai berikut :

1. Sistem Operasi Windows XP, Windows 7 dan sistem operasi lainnya.
2. Greenfoot 2.0 ke atas
3. Java Jdk 1.6 ke atas

3.1.6.3 Analisis *User*

Pada analisis user (pemakai) ini akan mencakup analisis beberapa parameter terhadap calon user dari aplikasi.

3.1.6.4 *User Knowledge and Experience*

Game Destroyer Garuda ini bisa digunakan oleh kalangan apapun dan tentunya mampu menggunakan komputer serta pengetahuan dan pengalaman akan memudahkan user dalam penggunaanya. Terutama pengetahuan dan pengalaman dalam memainkan *game* action atau shooter.

3.1.6.5 *User Physical Characteristic*

Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga terhadap user dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, Color Blind, Handednes, dan Gender seperti terlihat pada tabel 3.1.

Tabel 3.1 Analisis *Users Physical Characteristic*

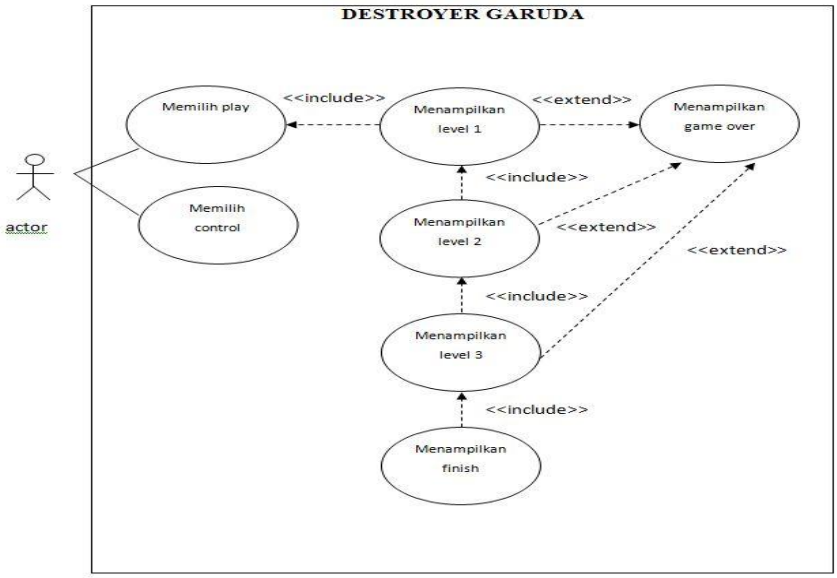
Age	6 tahun keatas
Gender	Pria dan Wanita
Handedness	Kanan dan Kiri
Color Blind	User yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu menggunakan aplikasi ini, karena tidak ada indicator warna-warna khusus yang membedakan antara fungsional yang satu dengan fungsional yang lainnya. Akan tetapi penggunaannya tidak akan optimal karena dalam *game* ini terdapat banyak sekali perbedaan warna yang menunjang interaksi dan ketertarikan dalam permainan.

3.1.6 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan tindak lanjut dari analisis sistem. Tujuannya adalah untuk memberikan gambaran mengenai sistem yang akan dibuat. Tahap awal yang dilakukan dalam melakukan analisis berorientasi objek mengunakan UML (*Unified Modeling Language*). Dimana tahap-tahap perancangan yang dilakukan dalam membangun aplikasi *game hybrid shooter side-scrolling destroyer* garuda antara lain *Usecase Diagram, Class Diagram, Sequence Diagram,* dan *Activity Diagram*.

3.1.7 *Usecase Diagram*

Usecase atau diagram *Usecase* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat dapat dilihat pada gambar 3.10.



Gambar 3.10 *Usecase Destroyer* Garuda

3.1.8 Definisi *Actor*

Definisi Actor berfungsi untuk menjelaskan Actor yang terdapat pada *Usecase* Diagram. Definisi Actor diterangkan pada Tabel 3.2.

Tabel 3.2 Definisi *Actor*

No	*Actor*	Deskripsi
1.	Pemain	Orang yang memainkan *Game*

3.1.9 Definisi Use Case

Definisi *Usecase* berfungsi untuk menjelaskan proses yang terdapat pada setiap use case. Skenario *Usecase* diterangkan pada Tabel 3.3.

Tabel 3.3 Definisi *usecase*

No	*Use Case*	Deskripsi
1	Memilih *play*	Proses untuk memilih menu *play*

2	Memilih control	Proses untuk melihat petunjuk control permainan
3	Menampilkan level 1	Proses untuk menampilkan level 1
4	Menampilkan level 2	Proses untuk menampilkan level 2
5	Menampilkan level 3	Proses untuk menampilkan level 3
6	Menampilkan *game over*	Proses untuk menampilkan *game over*
7	Menampilkan *finish*	Proses untuk menampilkan *finish*

3.1.10.1 *Skenario Usecase*

Skenario *Usecase* menunjukan proses apa saja yang terjadi pada setiap bagian di dalam *Usecase* tersebut, di mana user memberikan perintah pada setiap bagian *Usecase* dan nanti sistem akan meresponnya. *User case* scenario yang terdapat pada pembangunan aplikasi ini antara lain sebagai berikut :

3.1.10.2 *Skenario* Usecase Play

Skenario *Usecase Play* berfungsi untuk memberikan tampilan menu pilihan masing-masing pada *Play* sebagai *play game*. Skenario *Usecase Play* diterangkan pada Tabel 3.4.

Tabel 3.4 Skenario memilih play

Identifikasi	
Nomor	1
Nama	Memilih *play*
Tujuan	Menampilkan level 1
Deskripsi	Proses untuk memainkan level 1
Aktor	*User*
Skenario Utama	
Kondisi Awal	Menu utama *game*

Aksi Aktor	Memilih menu *play*
Reaksi Sistem	Aplikasi merespon dengan cara menampilkan level 1
Kondisi Akhir	Menampilkan level 1
Skenario Alternatif	
Kondisi Awal	-
Akasi Aktor	-
Kondisi Akhir	-

3.1.10.3 *Skenario* Usecase *Control*

Skenario *Usecase* Control berfungsi untuk memberikan tampilan petujuk control permainan. Skenario *Usecase* control diterangkan pada Tabel 3.5.

Tabel 3.5 Skenario memilih play

Identifikasi	
Nomor	2
Nama	Memilih control
Tujuan	Menampilkan control permainan
Deskripsi	Proses untuk melihat control permainan dalam *game*
Aktor	*User*
Skenario Utama	
Kondisi Awal	Menu control
Aksi Aktor	Melihat control perminan
Reaksi Sistem	Aplikasi merespon dengan cara menampilkan control permainan
Kondisi Akhir	Menampilkan control permainan

Skenario Alternatif	
Kondisi Awal	-
Akasi Aktor	-
Kondisi Akhir	-

3.1.10.4 *Skenario* Usecase *Level 1*

Skenario *Usecase* Level 1 *Game* berfungsi untuk memberikan tampilan level 1 pada *game*. Skenario *Usecase* Level 1 diterangkan pada Tabel 3.6.

Tabel 3.6 Skenario menampilkan level 1

Identifikasi	
Nomor	3
Nama	Menampilkan level 1
Tujuan	Menampilkan level 2
Deskripsi	Proses untuk memainkan level 2
Aktor	*User*
Skenario Utama	
Kondisi Awal	Level 1
Aksi Aktor	Menyelesaikan level 1
Reaksi Sistem	Aplikasi merespon dengan cara menampilkan level 2
Kondisi Akhir	Menampilkan level 2
Skenario Alternatif	
Kondisi Awal	Level 1
Aksi Aktor	*Player* utama mati
Kondisi Akhir	Menampilkan *game over*

3.1.10.5 *Skenario* Usecase *Level 2*

Skenario *Usecase* Level 2 *Game* berfungsi untuk memberikan tampilan level 1 pada *game*. Skenario *Usecase* Level 2 diterangkan pada Tabel 3.7.

Tabel 3.7 Skenario menampilkan level 2

Identifikasi	
Nomor	4
Nama	Menampilkan level 2
Tujuan	Menampilkan level 3
Deskripsi	Proses untuk memainkan level 3
Aktor	*User*
Skenario Utama	
Kondisi Awal	Level 2
Aksi Aktor	Menyelesaikan level 2
Reaksi Sistem	Aplikasi merespon dengan cara menampilkan level 3
Kondisi Akhir	Menampilkan level 3
Skenario Alternatif	
Kondisi Awal	Level 2
Aksi Aktor	*Player* utama mati
Kondisi Akhir	Menampilkan *game over*

3.1.10.6 *Skenario* Usecase *Level 3*

Skenario *Usecase* Level 3 *Game* berfungsi untuk memberikan tampilan level 1 pada *game*. Skenario *Usecase* Level 3 diterangkan pada Tabel 3.8.

Tabel 3.8 Skenario menampilkan level 3

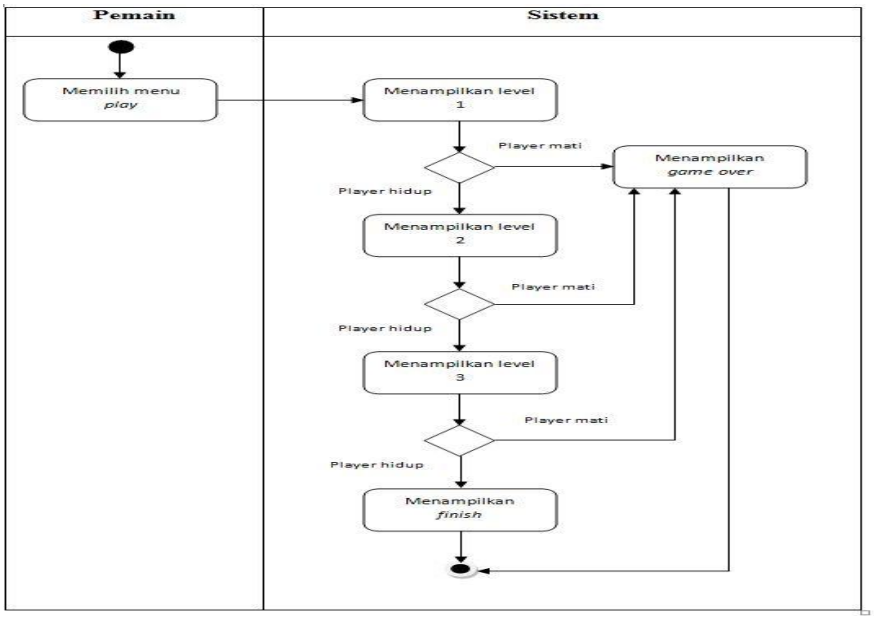
Identifikasi	
Nomor	5
Nama	Menampilkan level 3
Tujuan	Menampilkan *finish*
Deskripsi	Proses *finish*
Aktor	*User*
Skenario Utama	
Kondisi Awal	Level 3
Aksi Aktor	Menyelesaikan level 3
Reaksi Sistem	Aplikasi merespon dengan cara menampilkan *finish*
Kondisi Akhir	Menampilkan *finish*
Skenario Alternatif	
Kondisi Awal	Level 3
Aksi Aktor	*Player* utama mati
Kondisi Akhir	Menampilkan *game over*

3.1.10 Activity Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang.Berikut ini beberapa *activity* diagram yang terdapat pada *game* tersebut yaitu :

1. *Activity* Proses Menu Play

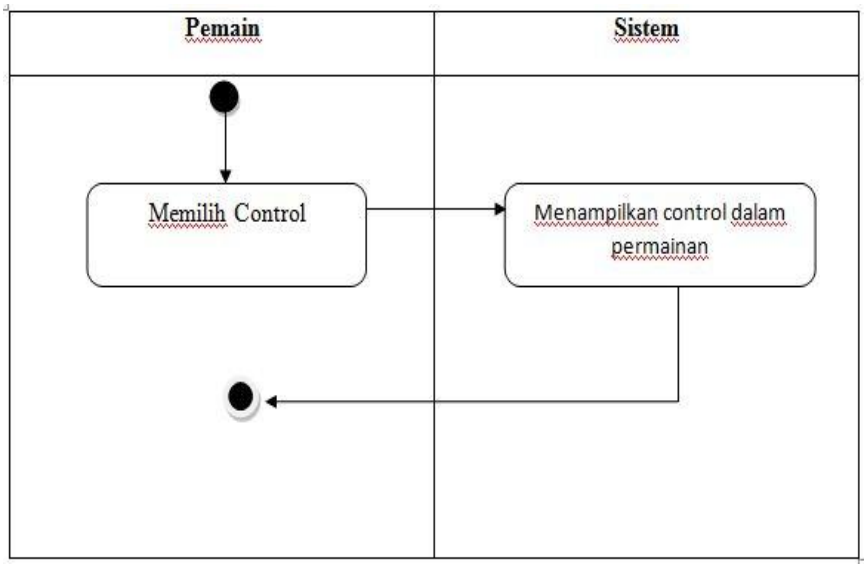
   Activity digram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang berjalan, bagaimana masing-masing alir berawal, proses yang terjadi dan bagaimana mereka berakhir, bisa di lihat pada gambar 3.11.



Gambar 3.11 ***Activity diagram menu play***

2. *Activity* Proses Control

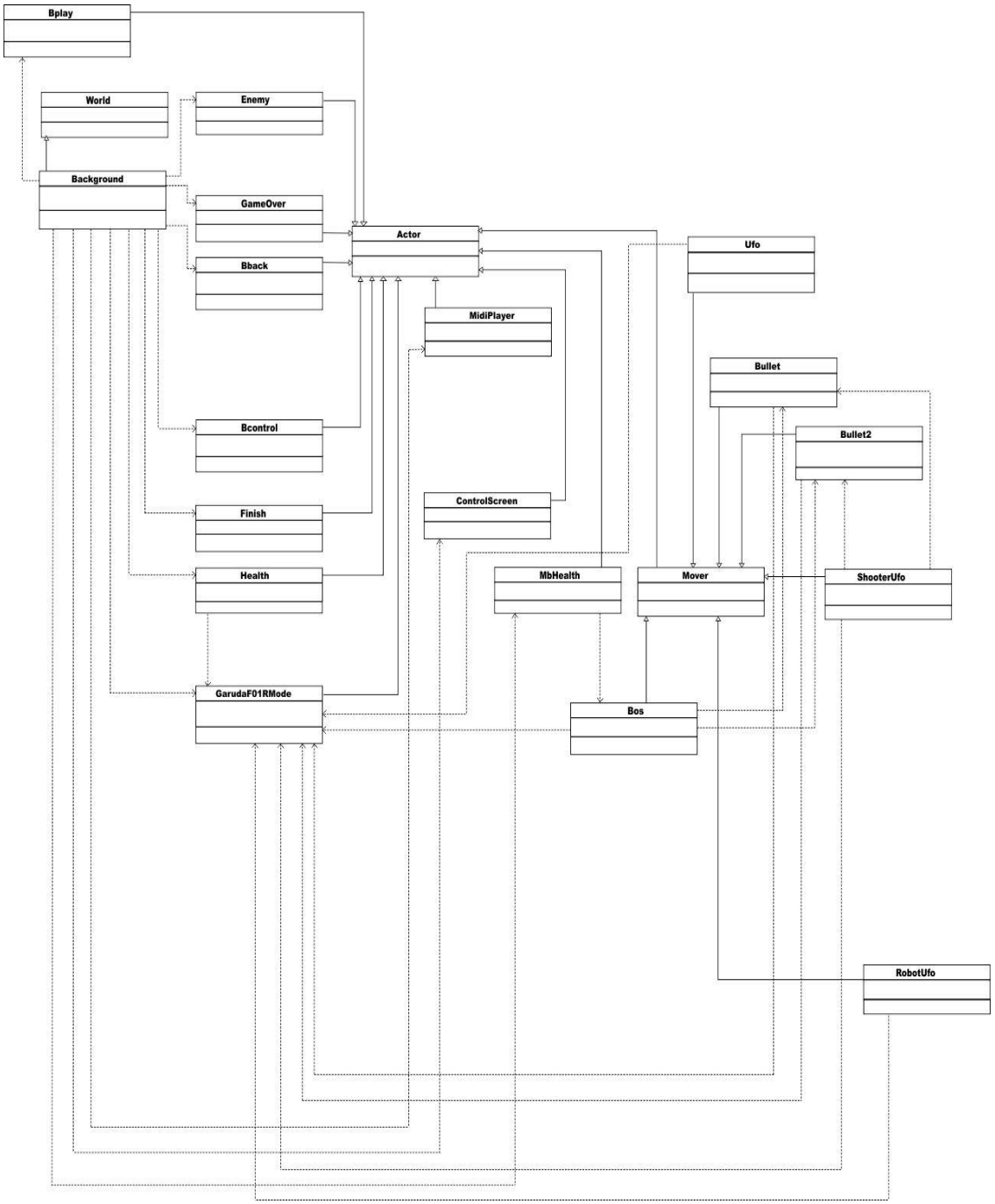
Menu control digunakan pemain untuk melihat control permainan dalam *game* destroyer garuda, proses yang terjadi pada menu control bisa di lihat pada gambar 3.12.



Gambar 3.12 ***Activity diagram menu control***

3.1.11 Class Diagram

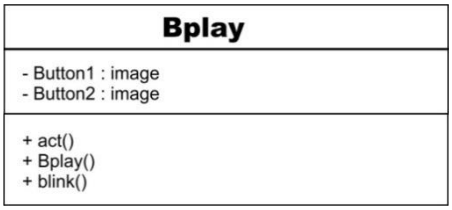
Class diagram adalah suatu diagram yang memperlihatkan suatu struktur dari sebuah sistem. Dapat dilihat pada gambar 3.13.



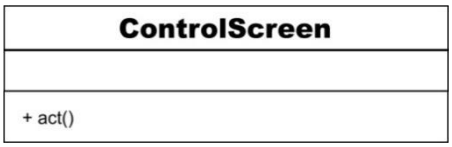
Gambar 3.13 Class Diagram *game* Destroyer Garuda

Nama nama kelas yang ada dalam *game* ini adalah sebagai berikut :

1. Kelas Bplay



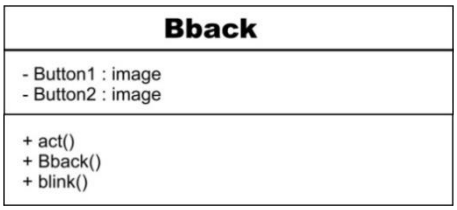
2. Kelas ControlScreen



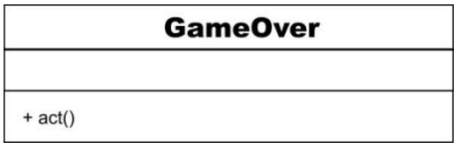
3. Kelas World



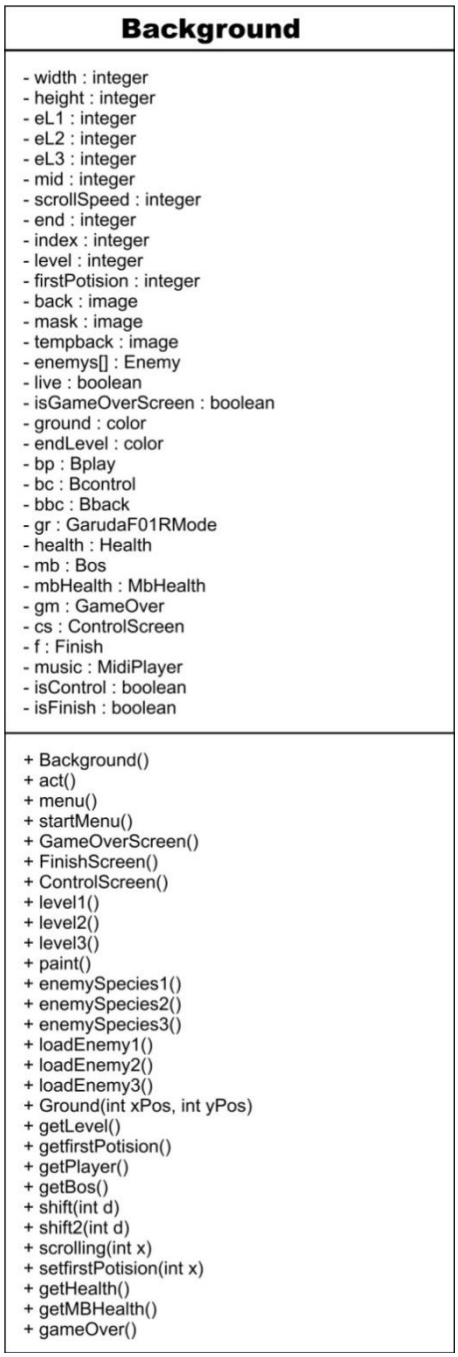
4. Kelas Bback



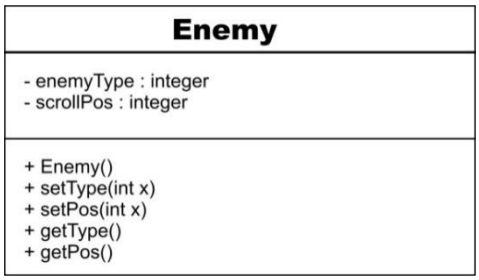
5. Kelas *Game*Over



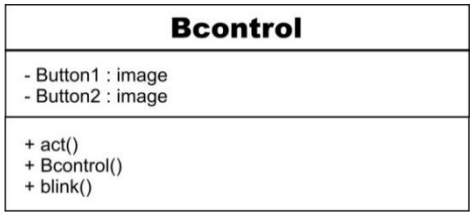
6. Kelas Background



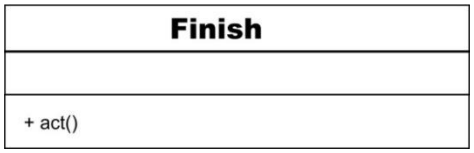
7. Kelas Enemy



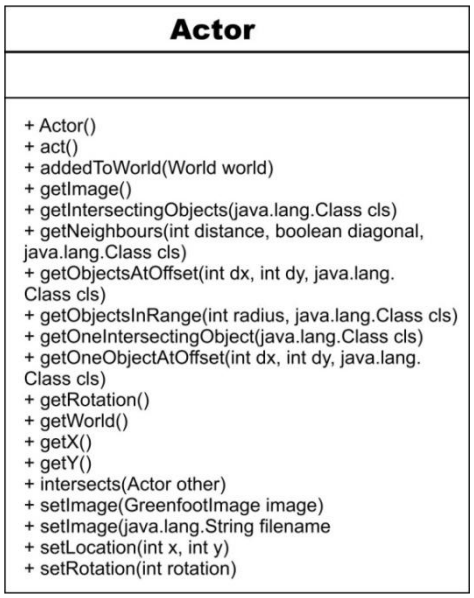
8. Kelas Bcontrol



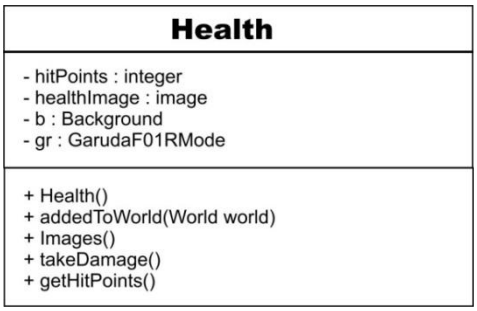
9. Kelas Finish



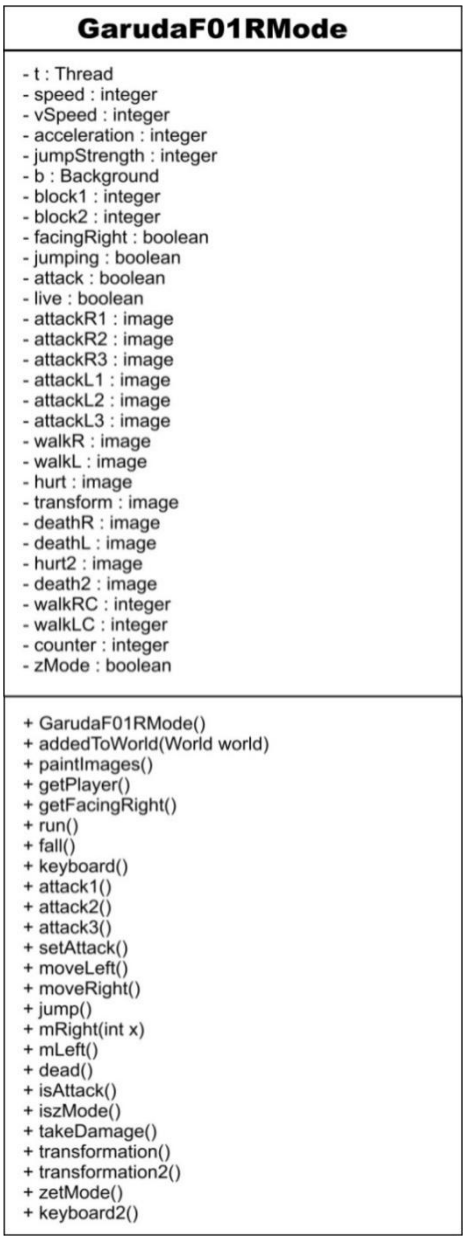
10. Kelas Actor



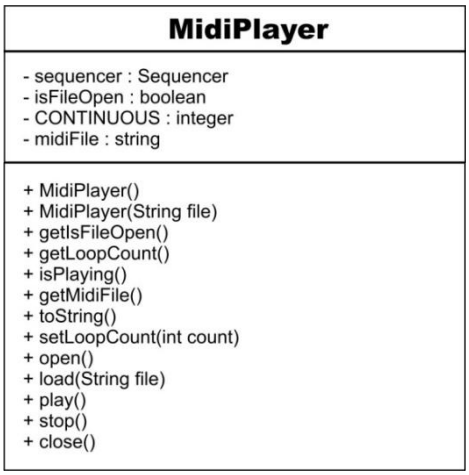
11. Kelas Health



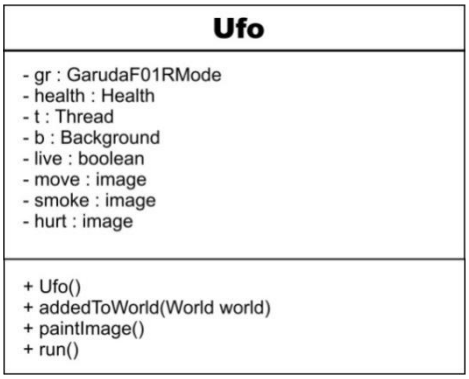
12. Kelas GarudaF01Rmode



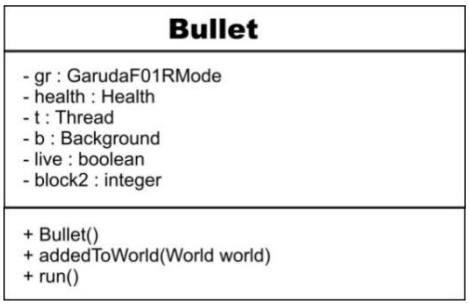
13. Kelas MidiPlayer



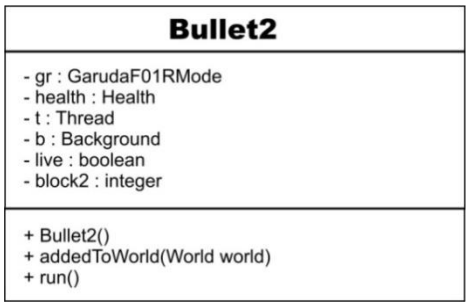
14. Kelas Ufo



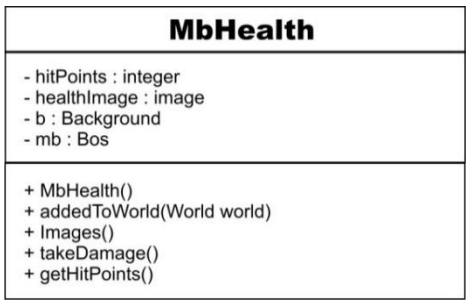
15. Kelas Bullet



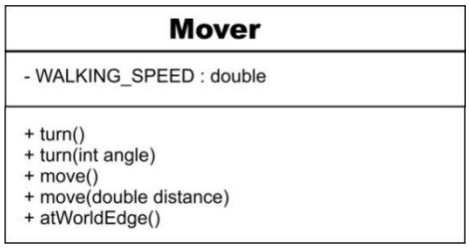
16. Kelas Bullet2



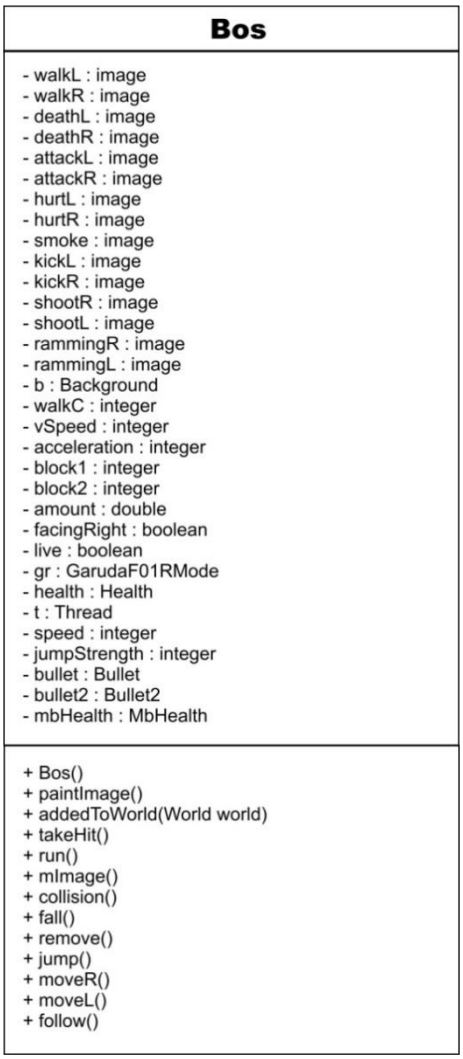
17. Kelas MbHealth



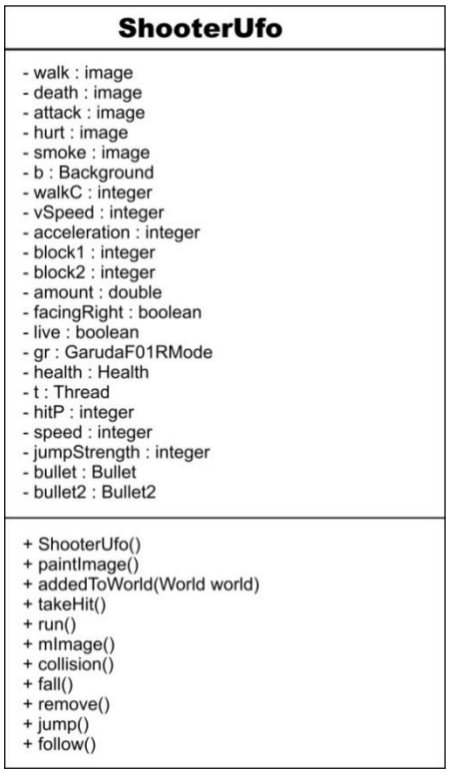
18. Kelas Mover



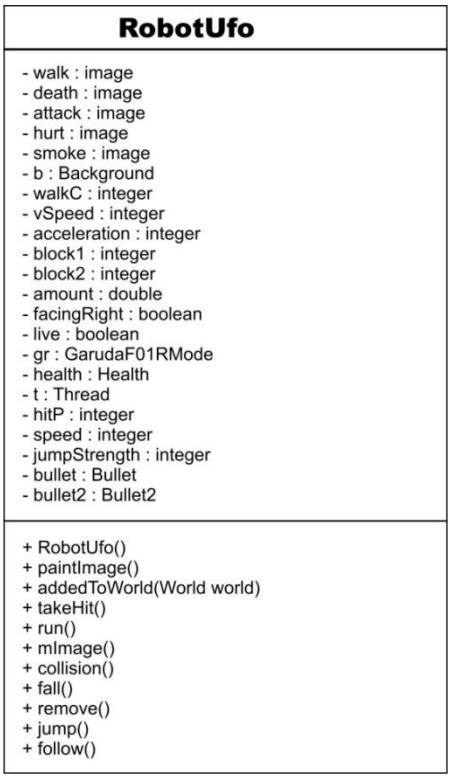
19. Kelas Bos


Bos
- walkL : image
- walkR : image
- deathL : image
- deathR : image
- attackL : image
- attackR : image
- hurtL : image
- hurtR : image
- smoke : image
- kickL : image
- kickR : image
- shootR : image
- shootL : image
- rammingR : image
- rammingL : image
- b : Background
- walkC : integer
- vSpeed : integer
- acceleration : integer
- block1 : integer
- block2 : integer
- amount : double
- facingRight : boolean
- live : boolean
- gr : GarudaF01RMode
- health : Health
- t : Thread
- speed : integer
- jumpStrength : integer
- bullet : Bullet
- bullet2 : Bullet2
- mbHealth : MbHealth
+ Bos()
+ paintImage()
+ addedToWorld(World world)
+ takeHit()
+ run()
+ mImage()
+ collision()
+ fall()
+ remove()
+ jump()
+ moveR()
+ moveL()
+ follow()

20. Kelas ShooterUfo



21. Kelas RobotUfo



3.1.12 Sequence Diagram

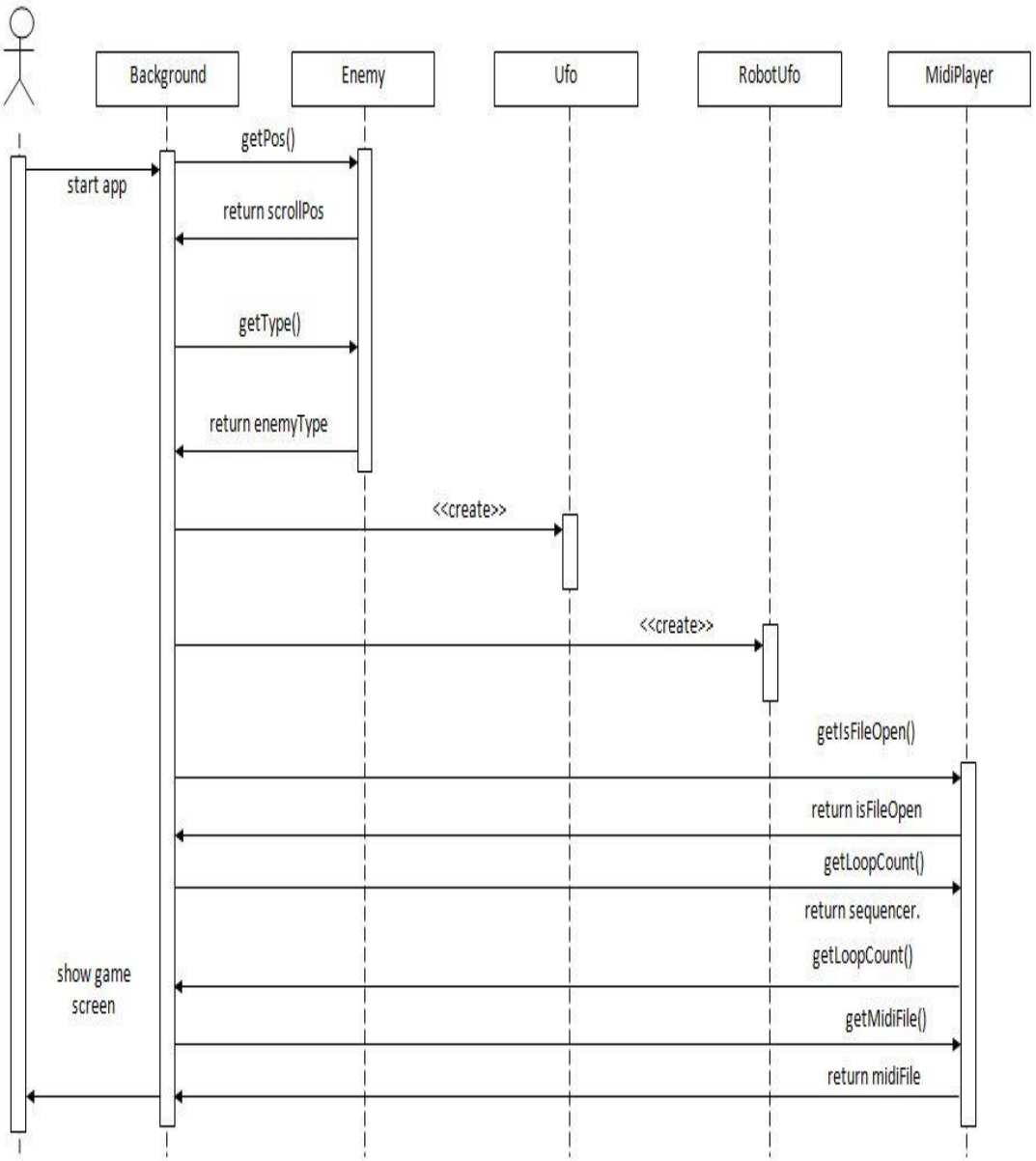
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarakan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Berikut ini beberapa *sequencial diagram* yang terdapat pada *game* Destroyer Garuda yaitu :

3.1.13.1 Sequence Diagram Level 1

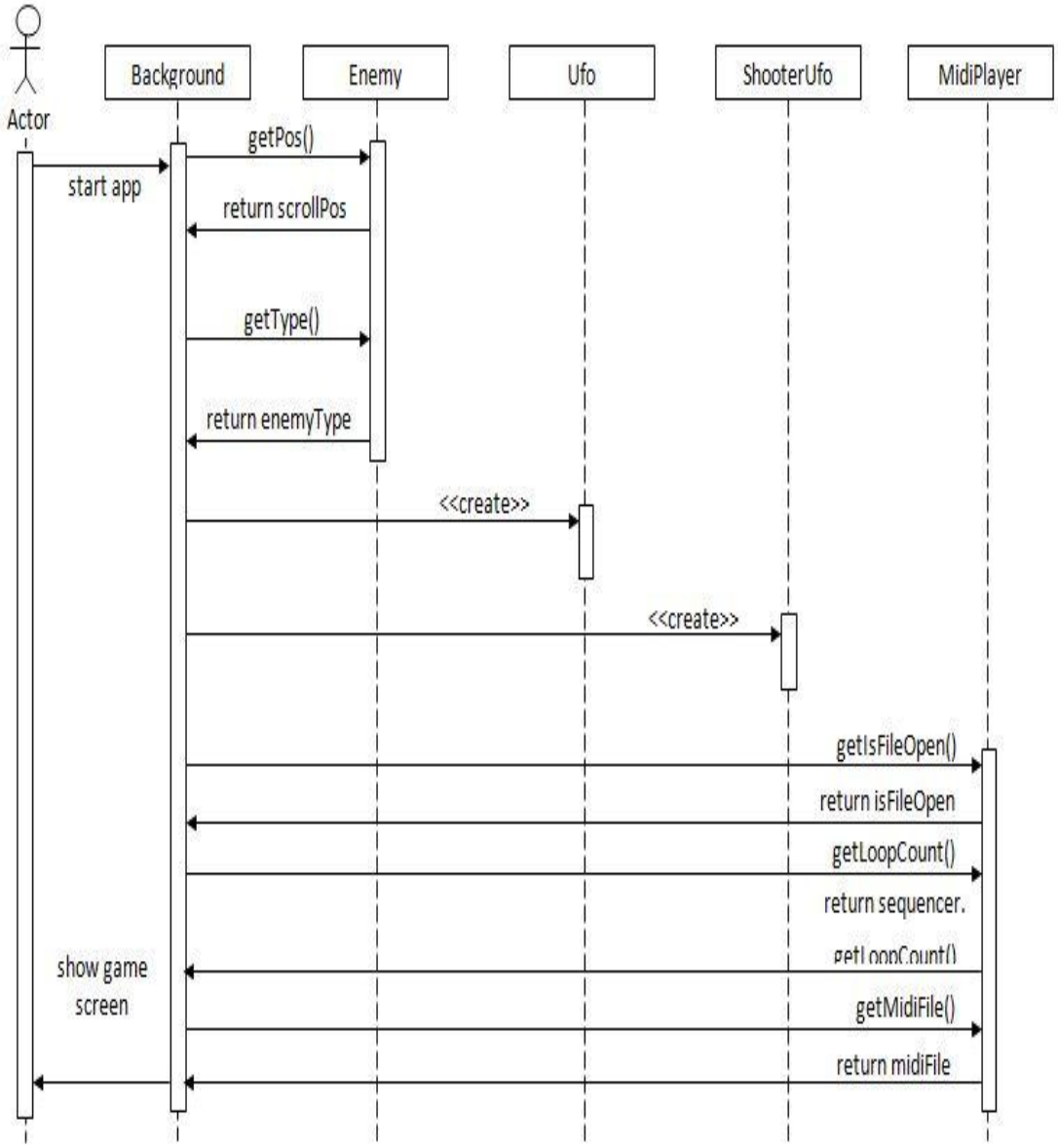
Sequence diagram untuk *Usecase* Level 1 menggambarkan interaksi antara objek dari class menu utama dan objek yang berkaitan dengan class level 1. Keseluruhan sequence level 1 dapat dilihat pada Gambar 3.14.



Gambar 3.14 Sequence Diagram level 1

3.1.13.2 Sequence Diagram Level 2

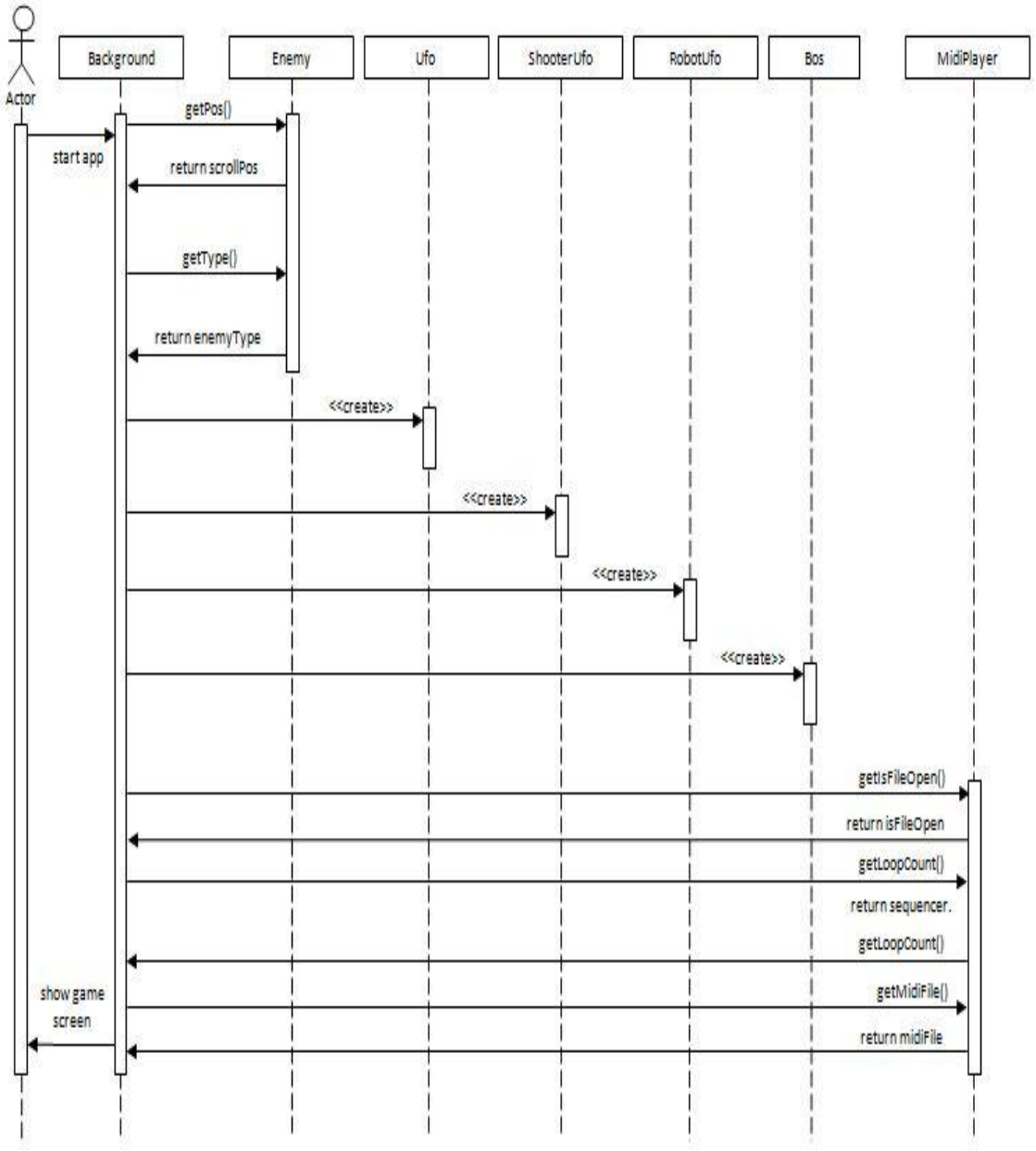
Sequence diagram untuk *Usecase* Level 2 menggambarkan interaksi antara objek dari class menu utama dan objek yang berkaitan dengan class level 2. Keseluruhan sequence level 2 dapat dilihat pada Gambar 3.15.



Gambar 3.15 Sequence diagram level 2

3.1.13.3 Sequence Diagram Level 3

Sequence diagram untuk *Usecase* Level 3 menggambarkan interaksi antara objek dari class menu utama dan objek yang berkaitan dengan class level 3. Keseluruhan sequence level 3 dapat dilihat pada Gambar 3.16.



Gambar 3.16 Sequence diagram level 3

3.1 Perancangan Sistem

Perancangan adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan setelah tahapan untuk memberikan gambaran secara terperinci. Dalam tahap ini digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean ke dalam suatu bahasa pemrograman.

Game yang akan di bangun ini berjudul *Destroyer Garuda*, *game* ini berjenis *side scrolling* karena *player* utama berjalan dari kiri ke kanan. AI yang di gunakan dalam musuh yang ada.Dalam *game* ini adalah algoritma djikstra, yang di mana algoritma ini merupakan pencarian buta untuk mencari solusi dengan jalur terpendek.

Tema yang di ambil dalam *game* ini adalah mengenai masa depan, di mana semua hal sudah di tangani dengan robot. Namun hal buruk terjadi karena serangan mahluk asing, satu demi satu negara hancur dan di kuasai oleh mereka tapi hanya satu negara yang masih mampu mempertahankan negaranya yaitu indonesia.

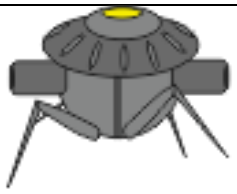
Indonesia dapat bertahan dari serangan para mahluk asing karena memiliki sebuah mesin tempur berbentuk pesawat yang mampu berubah bentuk menjadi robot yaitu Garuda F01.

3.2.1 Karakter *Game*

Karakter-karakter yang terdapat pada *game* ini akan dijelaskan pada tabel 3.9.

Tabel 3.9 Karakter *game* dan kemampuannya

Nama Karakter	Kemampuan
 Garuda F01 Robot Mode	Garuda F01 Robot Mode merupakan mode untuk pertempuran di darat. Mempunyai *hitpoint* sebanyak 10, dan mempunyai senjata berupa pedang laser. Serangan yang bisa dilakukan ada 3 macam yaitu tebasan pedang dari atas ke bawah, tebasan pedang dari bawah ke atas dan juga tusukan pedang secara horizontal.
 Garuda F01 Jet Mode	Garuda F01 Jet Mode merupakan mode untuk terbang yang digunakan untuk melewati tembok-tembok tinggi yang tidak bisa di lewati dengan loncatan ketika menggunakan mode robot, sehingga memerlukan mode

	ini untuk melewatinya dan mode ini memiliki *hitpoint* sebanyak 1 dan mode ini bisa menembak, namun karena daya hancur tembakkannya lemah jadi hanya bisa menghancurkan objek terbang saja.
 Robot Ufo	Robot ufo adalah musuh yang berada di darat, musuh tipe ini punya kemampuan mendeteksi keberadaan lawannya dalam jarak tertentu dan mengejarnya. Dan ketika masuk jarak serangnya, musuh tipe ini akan mengeluarkan jurus berupa listrik untuk mengalahkan lawannya. *Hitpoint* yang di punyai sebanyak 3, dan musuh tipe ini juga punya kemampuan untuk meloncat. Dan algoritma yang digunakan pada musuh ini adalah djikstra.
 Ufo	Ufo adalah musuh yang berada di udara, musuh tipe ini bergerak horizontal dan memiliki *hitpoint* sebanyak 1.
 Shooter Ufo	Shooter ufo adalah musuh yang berada di darat, musuh tipe ini punya kemampuan mendeteksi keberadaan lawannya dalam jarak tertentu dan mengejarnya. Dan ketika masuk jarak serangnya, musuh tipe ini akan mengeluarkan jurus tembakan berupa peluru meriam untuk mengalahkan lawannya. *Hitpoint* yang di punyai sebanyak 3, dan musuh tipe ini juga punya kemampuan untuk meloncat. Dan algoritma yang digunakan pada musuh ini adalah djikstra.

 Bos	Bos adalah musuh terakhir yang harus di hadapai oleh *player*, musuh tipe ini mempunyai *hitpoint* sebanyak 10. Mempunyai tiga jurus yaitu tendangan salto, Tandukan serta tembakan meriam.

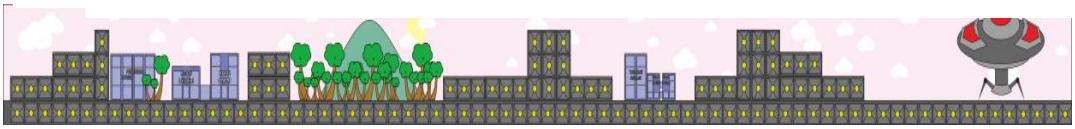
3.2.2 *Storyboard*

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan menggunakan *storyboard* dapat menyampaikan ide cerita kepada orang lain dengan lebih mudah karena dapat dengan *storyboard* dapat mengimajinasikan khayalan yang sesuai dengan gambar-gambar yang ada sehingga dapat menghasilkan persepsi yang sama pada ide cerita yang dibuat.

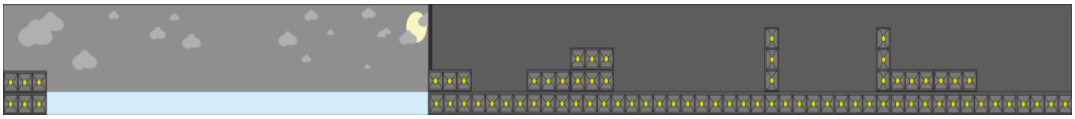
Berikut beberapa *storyboard* dari Aplikasi *Game* destroyer garuda yaitu :



Gambar 3.17 Storyboard level 1



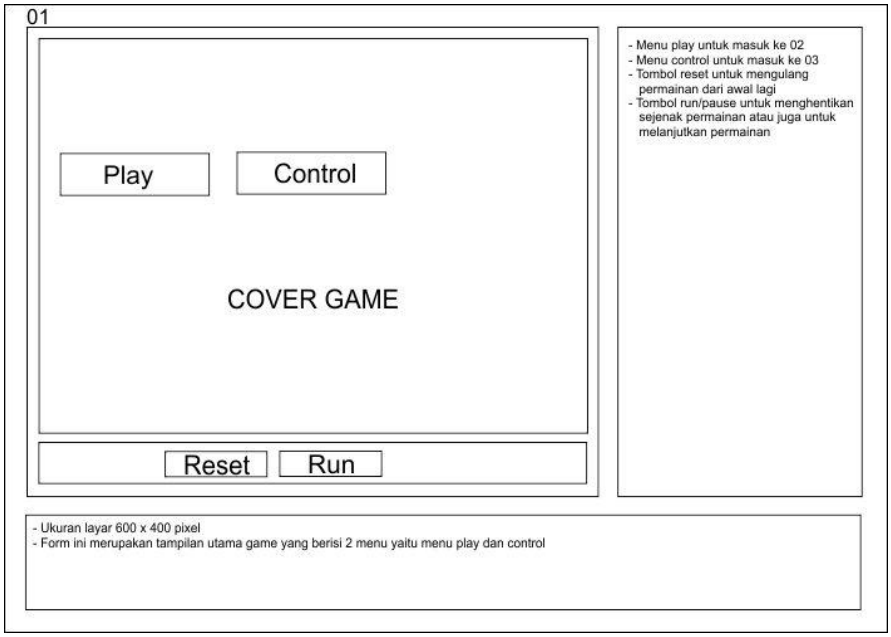
Gambar 3.18 Storyboard level 2



Gambar 3.19 Storyboard level 3

3.2.3 Perancangan Antarmuka Menu Utama

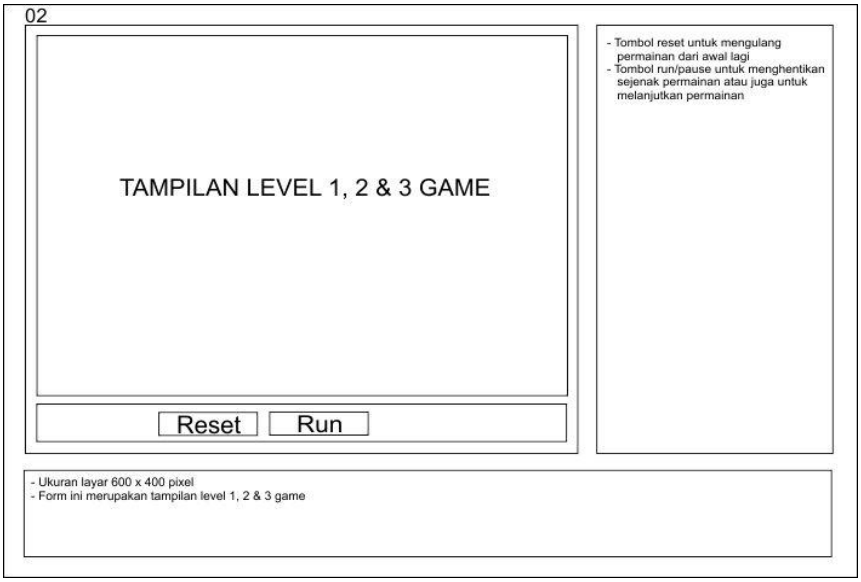
Antarmuka adalah salah satu bagian terpenting dalam perancangan suatu sistem, karena antarmuka lah yang akan berhadapan langsung dengan *user*. Antarmuka Menu Utama dapat dilihat pada gambar 3.20.



Gambar 3.20 Antarmuka menu utama *game*

3.2.4 Perancangan Antarmuka Level 1, 2, & 3 *game*

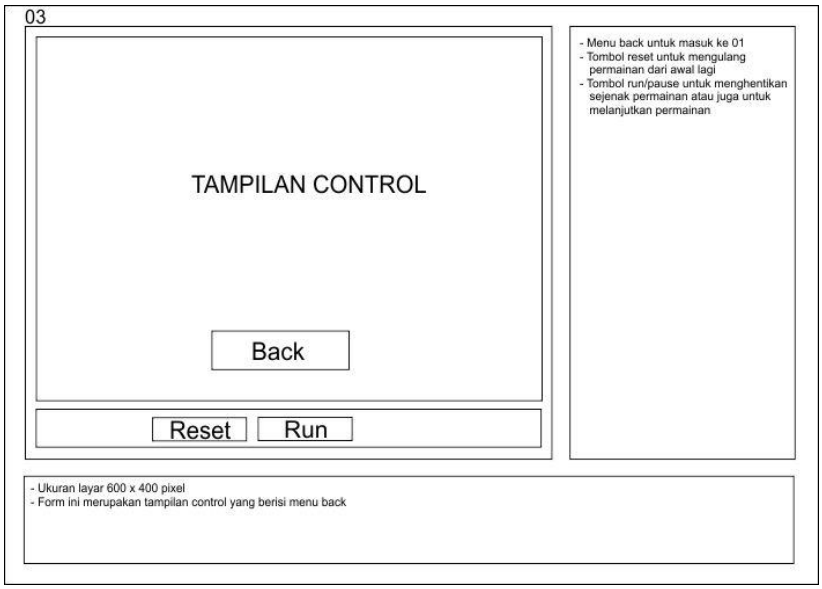
Antarmuka adalah salah satu bagian terpenting dalam perancangan suatu sistem, karena antarmuka lah yang akan berhadapan langsung dengan *user*. Antarmuka Menu Utama dapat dilihat pada gambar 3.21.



Gambar 3.21 Antarmuka tampilan Level 1, 2 & 3 *game*

3.2.5 Perancangan Antarmuka *Control*

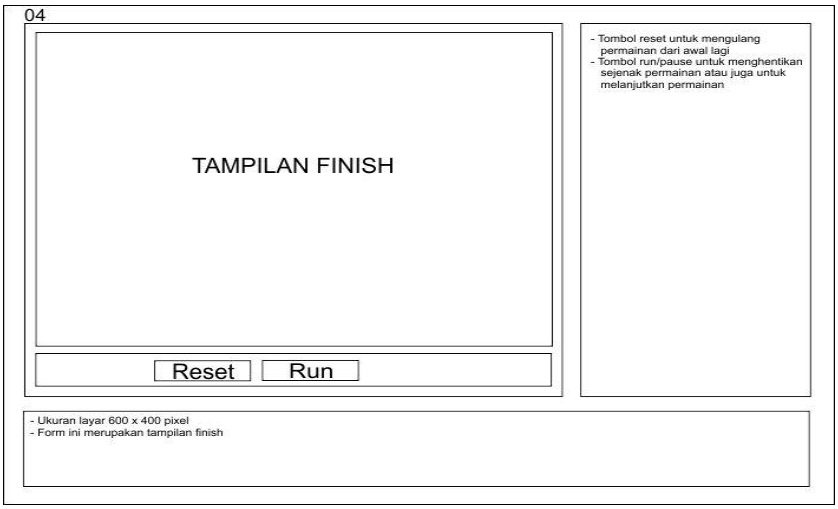
Masuk ke Menu *Control*, akan menampilkan info *control game.* Antarmuka *Control* dapat dilihat pada gambar 3.22.



Gambar 3.22 Antarmuka tampilan *control*

3.2.6 Perancangan Antarmuka *Finish*

Masuk ke akhir permainan menampilkan *form finish*. Antarmuka *Finish* dapat dilihat pada gambar 3.23.



Gambar 3.23 Antarmuka tampilan *finish*

3.2.7 Perancangan Antarmuka *Game Over*

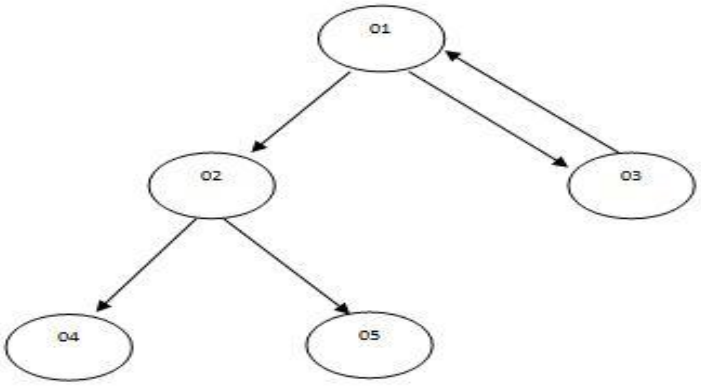
Gagal menyelesaikan permainan akan menampilkan *form game over*. Antarmuka *Game Over* dapat dilihat pada gambar 3.24.



Gambar 3.24 Antarmuka tampilan *Game Over*

3.2.8 Jaringan Semanti

Jaringan semantik menggambarkan hubungan antara tampilan satu dengan tampilan lainnya. Jaringan semantik dapat dilihat pada Gambar 3.25.



.

Gambar 3.25 Jaringan Semantik *game Destroyer* Garuda

Tabel 3.10 Pengkodean Tampilan

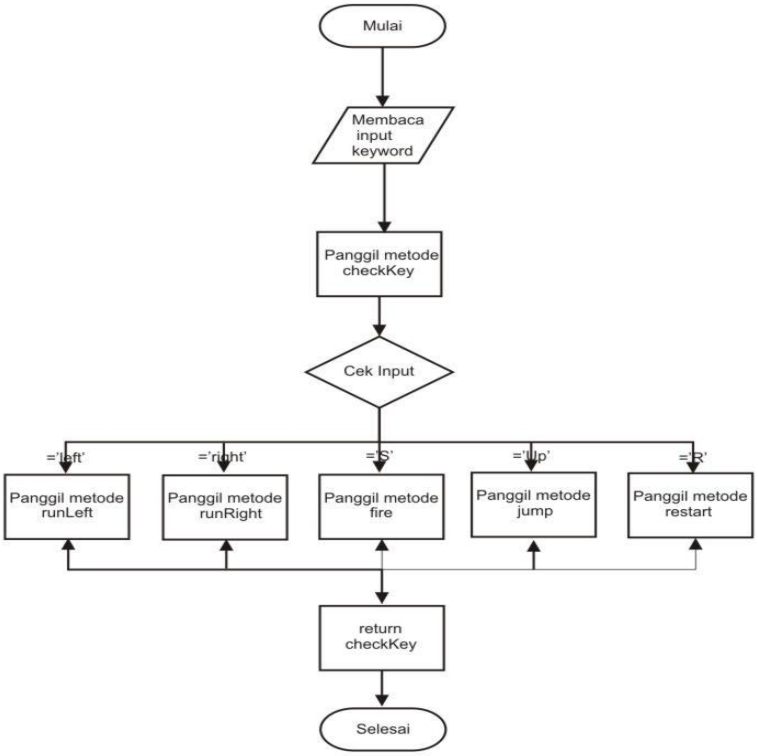
Kode Tampilan	Nama Tampilan
01	Antarmuka menu utama *game*
02	Antarmuka tampilan level 1, 2 & 3 *game*
03	Antarmuka tampilan *control*
04	Antarmuka tampilan *finish*
05	Antarmuka tampilan *game over*

3.2.9 Perancangan Method

Perancangan method merupakan perancangan yang berfungsi untuk mendeskripsikan method-method yang berada di dalam aplikasi. Method dapat dipanggil dengan menyertakan variabel, baik hanya satu variabel, banyak variabel atau bahkan tidak ada sama sekali .Adapun method-method yang terdapat dalam *game* Destroyer Garuda yang akan dibangun dijelaskan melalui flowchart sebagai berikut.

3.2.9.1 Perancangan Method *Check* Keys

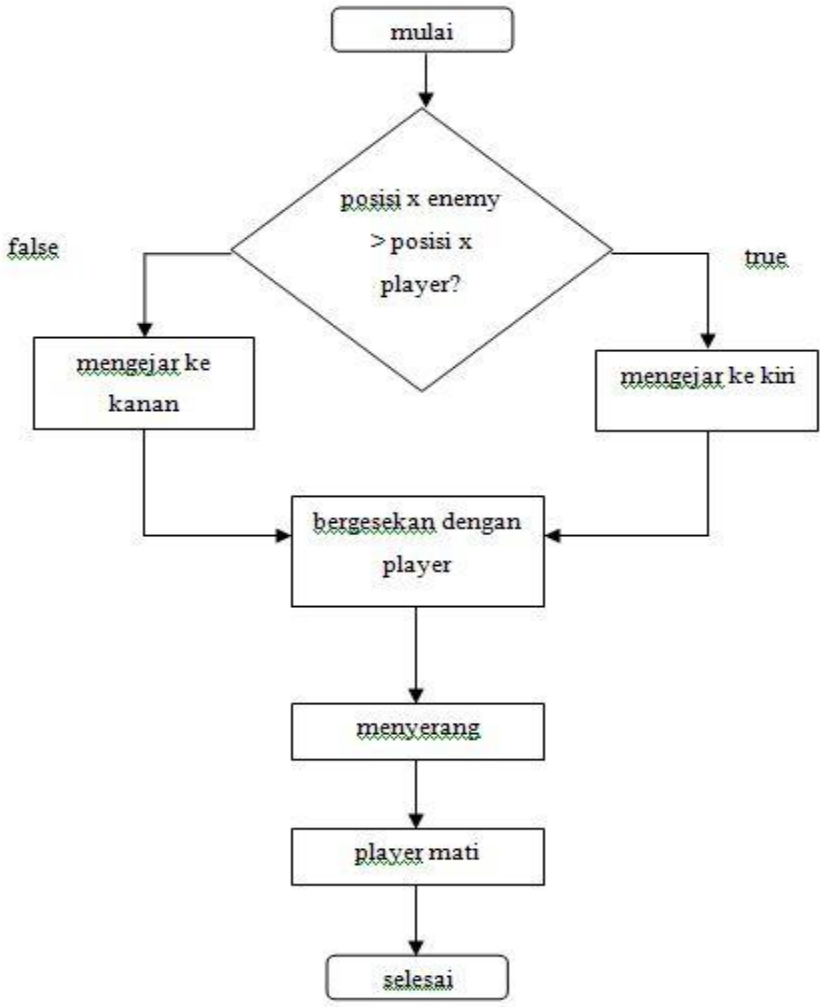
Metode ini berfungsi untuk penanganan pergerakan pada *player*. Untuk lebih jelas bisa dilihat pada gambar 3.26.



Gambar 3.26 Perancangan Method *Check* Keys

3.2.9.2 Perancangan Method follow

Method ini menjelaskan follow pada karakter Robot Ufo, ShooterUfo dan Bos, untuk lebih jelasnya dapat dilihat pada gambar 3.27.



Gambar 3.27 Perancangan Method follow Robot Ufo, ShooterUfo dan Bos

3.2.9.3 Perancangan Method Collition

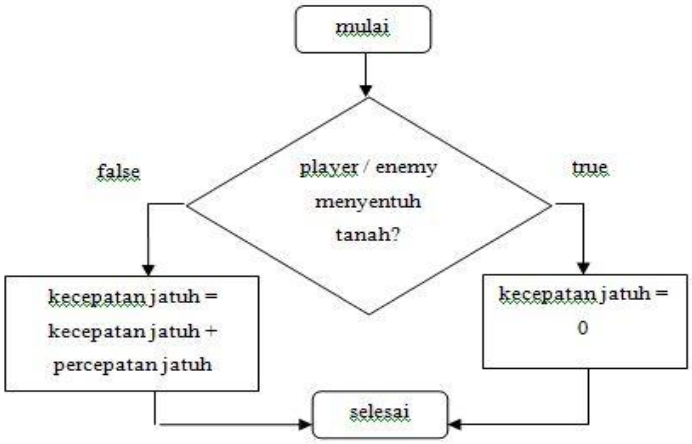
Method ini menjelaskan colliton pada karakter Robot Ufo, ShooterUfo dan Bos, untuk lebih jelasnya dapat dilihat pada gambar 3.27.



Gambar 3.28 Perancangan Method collition

3.2.9.4 Perancangan Method fall

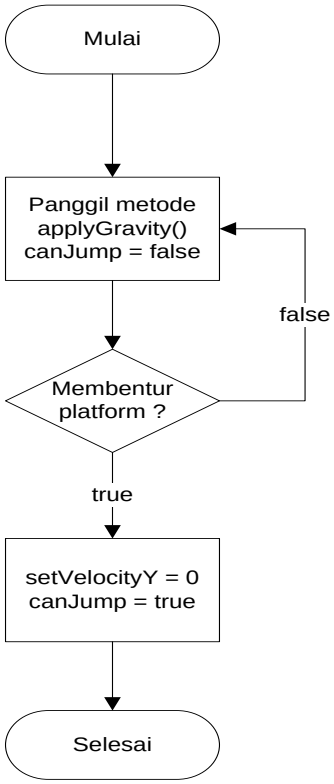
Method ini menjelaskan fall pada karakter GarudaF01Rmode, Robot Ufo, ShooterUfo dan Bos, untuk lebih jelasnya dapat dilihat pada gambar 3.29.



Gambar 3.29 Perancangan Method fall

3.2.9.5 Perancangan Method Jump

Method ini menjelaskan jump pada karakter GarudaF01 dan Robot Ufo untuk lebih jelasnya dapat dilihat pada gambar 3.30.



Gambar 3. 30 Perancangan Method Jump