BAB 3

ANALISIS DAN PERANCANGAN

3.1 Pengenalan Permainan

Permainan *arcade-shooter* yang dikembangkan merupakan pengembangan dari permainan video yang populer pada konsol Atari 2600 yaitu Asteroids. Berdasarkan studi literatur, permainan tersebut merupakan permainan yang mengawali masa keemasan permainan video yang dimulai pada dekade 70-an. Sebelumnya, perkembangan permainan video sempat mengalami kejenuhan sehingga perkembangannya melambat. Hal itu disebabkan kebanyakan permainan video yang beredar pada saat itu hanyalah *clone* 'tiruan' dari permainan *Pong*. Pada tahun 1978 muncullah jenis permainan *arcade-shooter* yang mengakhiri masa kejenuhan itu.

3.1.1 Permainan *arcade-shooter* konvensional

3.1.1.1 Elemen Permainan

Dalam permainan *arcade-shooter* konvensional elemen permainannya terdiri dari:

1. Arena Permainan

   Pada *arcade-shooter* konvensional arena permainan berbentuk kotak dengan warna gelap yang disituasikan seperti di ruang angkasa. Arena permainan ini merupakan tempat untuk bergerak dan berperang bagi entitas permainan lain seperti pesawat dan asteroid.

2. Pemain (*Human Player*)

   Pemain merupakan aktor utama yang menggunakan aplikasi permainan *arcade-shooter* konvensional. Ia dapat mengontrol pergerakan dan navigasi pesawat sehingga dapat bergerak dari satu titik ke titik lainnya pada arena permainan. Adapun tujuannya tersebut yaitu untuk menghindari asteroid ataupun untuk sekedar bergerak-gerak saja. Selain itu, pemain juga dapat mengontrol pesawat untuk menembakkan peluru yang tujuan utamanya adalah untuk menghancurkan asteroid.

3. Musuh

   Musuh pada permainan *arcade-shooter* konvensional adalah asteroid. Ia bergerak dan berputar-putar dengan pola acak.

3.1.1.2 Skenario Permainan

Pada permainan *arcade-shooter* konvensional, skenarionya disusun oleh beberapa hal sebagai berikut:

1. Alur Sistem

Tabel 3.1 Alur Sistem Permainan *Arcade-Shooter* Konvensional

Alur Sistem
1. Sistem menginisialisasi seluruh entitas permainan—arena permainan, pesawat, dan asteroid. 2. Asteroid berada pada posisi tertentu yang ditentukan oleh sistem secara acak. 3. Posisi pesawat berada ditengah arena permainan. 4. Permainan dimulai. 5. Asteroid bergerak disekitar arena permainan dan kemudian akan melakukan *warp* jika melewati batas layar. 6. Pemain mengontrol pesawat dengan memasukkan perintah tertentu melalui *keyboard* (pergerakan, navigasi dan menembak). 7. Sistem melakukan *update* terhadap posisi seluruh entitas permainan, logika sistem dan skor permainan.

2. Alur Permainan

Tabel 3.2 Alur Permainan Permainan *Arcade-Shooter* Konvensional

Alur Permainan
1. Sebuah pesawat antariksa terjebak dalam sebuah area di ruang angkasa yang penuh dengan asteroid. 2. Pesawat harus menghindar dari sejumlah asteroid yang beterbangan pada area tersebut. 3. Jika pesawat tertabrak asteroid, maka ia akan mati. 4. Pesawat dapat menggunakan senjatanya untuk menembak asteroid-asteroid tersebut. 5. Jika asteroid tertembak, maka ia akan lenyap dan skor permainan akan bertambah.

3. Misi Permainan

Tabel 3.3 Misi Permainan Permainan *Arcade-Shooter* Konvensional

Misi
1. Menghancurkan semua musuh berupa asteroid yang beterbangan pada arena permainan. 2. Menghindari setiap asteroid yang berterbangan. 3. Meraih skor tertinggi.

3.1.1.3 Entitas Permainan

Beberapa entitas permainan yang terlibat pada permainan *arcade-shooter* konvensional dapat dilihat pada tabel berikut:

Tabel 3.4 Entitas Permainan *Arcade-Shooter* Konvensional

No	Nama	Gambar	Penjelasan
1	Pesawat		Pesawat merupakan entitas utama permainan yang dikendalikan langsung oleh pemain. Ia dapat berganti posisi dari satu posisi ke posisi lain sesuai perintah dari pemain yang dimasukkan melalui *keyboard*. Selain itu ia juga dapat menembakkan peluru.

2	Asteroid		Asteroid merupakan entitas lain pada permainan dan merupakan musuh yang harus dibasmi dan dihindari oleh entitas pesawat. Ia bergerak berputar-putar di arena permainan.
3	Peluru		Peluru merupakan entitas yang muncul ketika pemain memasukkan perintah menembak terhadap pesawat melalui perantara *keyboard*. Ia merupakan senjata utama pesawat yang dapat mengahancurkan asteroid-asteroid yang beterbangan.

3.1.2 ArkAId XIV

ArkAId XIV merupakan nama dari permainan *arcade-shooter* yang dikembangkan. Adapun kata ArkAId berasal dari kata *arcade* yang merupakan *genre* dari permainan ini. Huruf AI pada pertengahan katanya sengaja dikapitalkan untuk menandai bahwa permainan ini menerapkan *game* AI yaitu FSM. Selanjutnya, huruf XIV merupakan angk romawi yang berarti angka 14 dalam huruf latin. Angka teresbut bukan menandakan bahwa permainan ini merupakan seri ke 14, melainkan angka tersebut didedikasikan kepada rekan-rekan penulis di kelas IF-14 2007 Universitas Komputer Indonesia.

3.1.2.1 Elemen Permainan

Dalam ArkAId XIV elemen permainannya masih sama seperti pada permainan *arcade-shooter* konvensional, namun ada beberapa elemen tambahan. Adapun elemen-elemen tersebut terdiri dari:

1. Arena Permainan

   Arena permainan pada ArkAId XIV masih sama dalam segi bentuk dengan yang digunakan pada *arcade-shooter* konvensional. Sebagai

pengembangannya arena permainan digantikan menjadi berbasis *image* yang mana gambarnya adalah ruang angkasa yang dipenuhi bintang dan objek angkasa lainnya. Arena permainan ini merupakan tempat untuk bergerak dan berperang bagi entitas permainan lain seperti pesawat dan asteroid.

2. Pemain (*Human Player*)

   Sama halnya dengan *arcade-shooter* konvensional pada ArkAId XIV, pemain merupakan aktor utama yang menggunakan aplikasi permainan. Ia mengontrol pergerakan dan navigasi pesawat sehingga dapat bergerak dari satu titik ke titik lainnya pada arena permainan. Adapun tujuannya tersebut yaitu untuk menghindari asteroid dan musuh lainnya, meraih *power up* ataupun untuk sekedar bergerak-gerak saja. Selain itu, pemain juga dapat mengontrol pesawat untuk menembakkan peluru yang tujuan utamanya adalah untuk menghancurkan musuh-musuh yang ada pada arena permainan.

3. Musuh (*Non-Human Players*)

   Musuh pada ArkAId XIV terdiri dari beberapa jenis, tetapi musuh pada permainan *arcade-shooter* konvensional, yaitu asteroid, tetap dimunculkan kembali. Pada permainan hasil pengembangan ini, asteroid bukan hanya dapat bergerak dan berputar-putar dengan pola acak, melainkan kini ia dapat berubah menjadi kepingan-kepingan kecil. Kepingan-kepingan tersebut juga harus dihancurkan, karena mereka juga mengancam kelangsungan hidup pesawat. Selain itu, musuh baru yang dimunculkan pada ArkAId XIV adalah beberapa UFO dan *alien* ‘makhluk luar angkasa’.

4. *Power Up*

   Elemen permainan baru yang ditambahkan adalah *power up* yang merupakan "suplemen" bagi entitas pesawat. Adapun jenisnya secaara umum terdiri dari empat macam yaitu skor, *shield, cola*, dan *gun*. Skor dibagi lagi menjadi tiga jenis berdasarkan jumlah skor yang akan ia tambahkan jika diraih oleh pesawat yaitu 250, 500, dan 1000.

3.1.2.2 Skenario Permainan

Skenario pada ArkAId XIV tidak jauh berbeda dari permainan *arcade-shooter* konvensional, namun evolusi tentunya dilakukan pada beberapa hal. Adapun skenario hasil pengembangan tersebut disusun oleh beberapa hal sebagai berikut:

1. Alur Sistem

Tabel 3.5 Alur Sistem ArkAId XIV

Alur Sistem
1. Sistem menginisialisasi seluruh entitas permainan—arena permainan, pesawat, asteroid, UFO dan *alien*. 2. Asteroid, UFO dan *alien* berada pada posisi tertentu yang ditentukan oleh sistem secara acak. 3. Posisi pesawat berada ditengah arena permainan. 4. Permainan dimulai. 5. Seluruh entitas musuh bergerak disekitar arena permainan dan kemudian akan melakukan *warp* jika melewati batas layar. 6. Pemain mengontrol pesawat dengan memasukkan perintah tertentu melalui *keyboard* (pergerakan, navigasi dan menembak). 7. Sistem melakukan *update* terhadap posisi seluruh entitas permainan, logika sistem dan skor permainan. 8. Jika asteroid berukuran terkecil hancur, maka ia akan berubah menjadi *power up*.

2. Alur Permainan

Tabel 3.6 Alur Permainan ArkAId XIV

Alur Permainan
1. Sebuah pesawat antariksa terjebak dalam sebuah area di ruang angkasa yang penuh dengan asteroid, *alien* ganas, dan UFO yang sedang berpatroli. 2. Pesawat harus menghindar dari sejumlah asteroid yang beterbangan pada area tersebut. 3. Jika pesawat tertabrak asteroid, maka ia akan mati. 4. Pesawat juga harus menghindari patrol UFO dan alien ganas yang ada, namun jika terpaksa ia harus menembak mereka. 5. Pesawat dapat menggunakan senjatanya untuk menembak asteroid, UFO dan *alien* tersebut. 6. Jika entitas-entitas musuh tertembak, maka mereka akan lenyap dan skor permainan akan bertambah. 7. Power up akan muncul pada arena permainan ketika asteroid berukuran terkecil hancur, adapun jenisnya ditentukan secara acak oleh sistem.

3. Misi Permainan

Tabel 3.7 Misi Permainan ArkAId XIV

Misi
1. Menghancurkan semua musuh yang terdiri dari asteroid, UFO dan *alien* yang berkeliaran di sekitar arena permainan. 2. Menghindari setiap asteroid yang berterbangan atau menembaknya. 3. Melawan semua musuh yang ada pada arena permainan. 4. Meraih skor tertinggi.

3.1.2.3 Entitas Permainan

Pada dasarnya entitas-entitas permainan yang terlibat pada ArkAId XIV dengan yang ada pada permainan *arcade-shooter* konvensional, tetapi mereka kini telah berbasis *sprite* dan *animated sprite*. Selain itu, beberapa entitas baru pun ditambahkan dan penjelasannya ada pada tabel berikut:

Tabel 3.8 Entitas Permainan ArkAId XIV

No	Nama	Gambar	Penjelasan
1	Pesawat		Pesawat merupakan entitas utama permainan yang dikendalikan langsung oleh pemain. Ia dapat berganti posisi dari satu posisi ke posisi lain sesuai perintah dari pemain yang dimasukkan melalui *keyboard*. Selain itu ia juga dapat menembakkan peluru plasma.
2	Asteroid		Asteroid merupakan entitas lain pada permainan dan merupakan musuh yang harus dibasmi dan dihindari oleh entitas pesawat. Ia bergerak berputar-putar di arena permainan. Selain itu, jika asteroid ini tertembak, maka ia akan berubah menjadi kepingan-kepingan yang berukuran lebih kecil. Kepingan-kepingan asteroid tersebut kemudian menjadi entitas musuh baru lagi yang tentunya harus di hindari atau dihancurkan.
3	UFO		Pada ArkAId XIV terdapat entitas musuh baru yaitu UFO (*unidentified flying object*). Ia berpatroli disekitar arena permainan dan menembaki pesawat pemain yang dianggap menganggunya.
4	Alien		*Alien* juga merupakan entitas musuh baru yang dimunculkan pada ArkAId XIV. Ia bergentayangan di arena permainan dan merupakan musuh lain yang harus dibasmi.
5	Power up 250		Fungsi dari entitas *power up* ini adalah menambah skor sebesar 250 poin pada skor total ketika ia diraih oleh pesawat.
6	Power up 500		Fungsi dari entitas *power up* ini adalah menambah skor sebesar 500 poin pada skor total ketika ia diraih oleh pesawat.

7	Power up 1000		Fungsi dari entitas *power up* ini adalah menambah skor sebesar 1000 poin pada skor total ketika ia diraih oleh pesawat.
8	Power up Shield		Fungsi dari entitas *power up* ini adalah menambah fitur *shield* ketika ia diraih oleh pesawat. Ketika *shield* digunakan oleh pesawat maka ia kebal terhadap serangan musuh.
9	Power up Cola		Fungsi dari entitas *power up* ini adalah menambah *health* 'kesehatan' pada *health bar* ketika ia diraih oleh pesawat. Adapun jika *health bar* pesawat penuh, maka *power up* ini tak akan memberi efek apapun.
10	Power up Gun		Fungsi dari entitas *power up* ini adalah memperbaharui level senjata ketika ia diraih oleh pesawat. Level senjata tertinggi adalah 5 dan yang terendah adalah 1 (*default*).
11	Peluru Plasma		Peluru plasma merupakan entitas yang muncul ketika pemain memasukkan perintah menembak terhadap pesawat melalui perantara *keyboard*. Ia merupakan senjata utama pesawat yang dapat mengahancurkan musuh-musuh yang ada pada arena permainan.

3.2 Analisis Perangkat Lunak

Analisis perangkat lunak merupakan penguraian dari suatu perangkat lunak yang utuh kedalam bagian-bagian komponennya. Hal itu dimaksudkan untuk mengidentifikasi dan mengevaluasi berbagai permasalahan, kesempatan dan hambatan yang terjadi dari kebutuhan yang diharapkan, sehingga dapat diusulkan perbaikan-perbaikan.

Analisis perangkat lunak merupakan tahap yang kritis dan sangat penting karena apabila terjadi kesalahan dalam tahap ini, maka akan mengakibatkan kesalahan pada tahap-tahap selanjutnya.

3.2.1 Deskripsi Masalah

Aplikasi permainan dibangun dengan tujuan untuk menjadikannya sebagai sarana hiburan bagi para penggunanya. Tujuan tersebut akan dicapai jika aplikasi permainan itu menarik. Adapun beberapa faktor yang dapat mewujudkan hal tersebut diantaranya terdiri dari penyajian yang menarik pada tampilan antarmuka, efek suara, dan musik latar. Semua itu berlaku pada semua jenis permainan tidak terkecuali *arcade-shooter* yang merupakan objek penelitian. Berdasarkan hasil penelitian, pada aplikasi permainan *arcade-shooter* konvensional terdapat beberapa masalah yang menyebabkan permainan tersebut menjadi kurang menarik.

Masalah pertama adalah penerapan AI yang masih lemah pada objek musuh. Hal itu tentunya berimbas pada menurunnya *gaming experience* bagi pengguna. Pada *arcade-shooter* konvensional AI yang diterapkan pada objek musuh adalah *pattern based navigation.* AI ini menjadikan musuh bergerak menurut suatu pola tertentu seperti berputar-putar, diagonal, lurus atau kombinasi dari semuanya. Perlawanan nya pun juga menerapkan *pattern based* AI. Pola pada *pattern based navigation* AI dibentuk oleh suatu algoritma tertentu yang ditanamkan pada objek musuh. Pada permainan Asteroid algoritma yang digunakannya adalah algoritma random.

Penurunan *gaming experience* terjadi karena *gameplay* ‘alur permainan’ pada *arcade-shooter* konvensional terlalu monoton. Hal itu akan terasa oleh pengguna yang telah memainkannya dengan durasi lebih dari delapan jam. Pergerakan dan perlawanan musuh yang terpola menjadikannya mudah ditebak serumit apapun pola tersebut. Oleh karenanya, seiring dengan bertambahnya durasi bermain, *gaming experience* akan terus menurun karena pengguna akan merasa tidak tertantang lagi dan bosan. Itulah alasan kenapa permainan *arcade-shooter* konvensional menjadi tidak menarik.

Penyajian tampilan antarmuka pada permainan *arcade-shooter* konvensional tidak banyak berubah sejak masa awal kejayaannya. Hal itu mungkin tidak menjadi masalah bagi para pengguna lama yang kebanyakan sudah berusia dewasa, tetapi tidak demikian dengan para pengguna baru yang berusia belia. Kenyataanya kebanyakan pengguna aplikasi permainan masih berusia belia dan merupakan generasi baru yang menginginkan segala sesuatu itu harus serba baru dan menantang. Sesuatu yang mereka anggap kuno tentunya akan mereka tinggalkan.

Untuk membuat permainan *arcade-shooter* menjadi lebih menarik maka diperlukan penyajian tampilan antarmuka yang lebih canggih dari yang lama. Dengan demikian penggunanya tidak akan merasa aneh dan tidak nyaman saat bermain. Penyajian tampilan antarmuka yang canggih dapat dilakukan dengan memperabaharui *look and feel* dari permainan tersebut. Hal itu dapat diwujudkan dengan penggunaan teknik grafis yang lebih canggih dalam pengembangannya.

Adapun beberapa objek grafis yang terlibat harus dibuat dengan semenarik mungkin sehingga dapat terlihat lebih canggih dan menjadikannya menarik.

Selain itu audio yang terdiri dari efek suara dan musik latar yang digunakan juga harus diperbaharui. Alasannya hampir sama dengan alasan diperlukannya permbaharuan pada tampilan antarmuka. Efek suara, musik latar dan tampilan grafis dapat memepengaruhi emosi dari pengguna yang memainkannya. Semua hal tersebut tentunya harus disajikan dengan sedemikian rupa. Jika pengguna dapat terhanyut dalam keasyikan bermain maka sudah dipastikan permainan tersebut merupakan permainan yang menarik. Semua itulah yang harus diterapkan saat pengembangan aplikasi permainan *arcade-shooter* ini.

3.2.2 Analisis Masalah

Berdasarkan deskripsi masalah dan didukung pula oleh data hasil penelitian melalui metode kuisioner pada saat studi pendahuluan yang dapat dilihat pada lampiran, diketahui bahwa beberapa hal yang menyebabkan permasalahan pada permainan *arcade-shooter* konvensional sehingga ia dikatakan tidak menarik diantaranya:

1. Alur permainan yang monoton dan terlalu sederhana.
2. Tampilan grafis masih dalam bentuk garis dan vektor.
3. Audio yang masih berbasis 8-bit, bahkan bisu pada versi yang lebih klasik..
4. Kurang variatifnya entitas permainan terutama musuh.
5. Kontrol navigasi objek pesawat yang buruk.

Hal-hal tersebut lah yang menyebabkan permainan *arcade-shooter* konvensional menjadi kurang menarik.

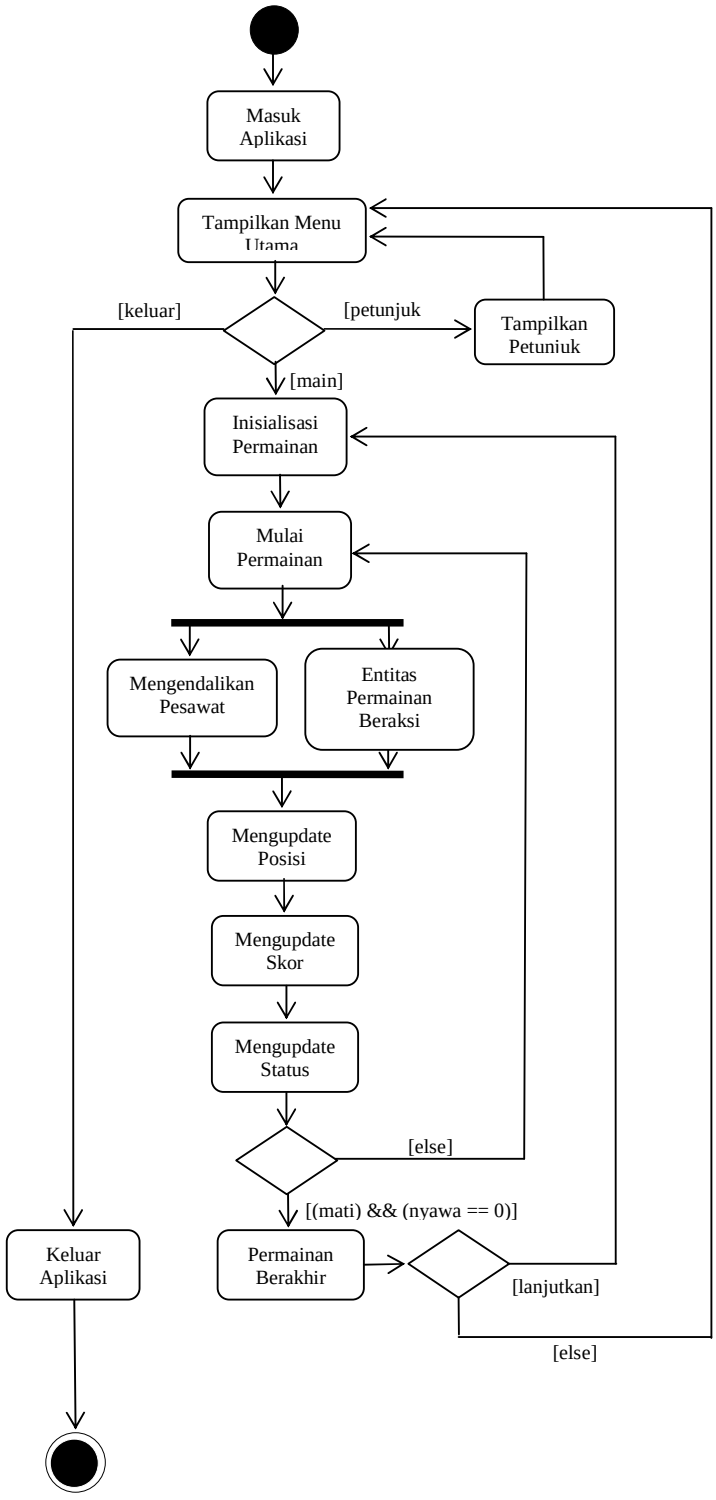
3.2.3 Analisis Metode

3.2.3.1 Metode pada Permainan *Arcade-Shooter* Konvensional

Pada permainan *arcade-shooter* konvensional AI yang diterapkan pada objek musuh adalah *pattern based.* AI tersebut menjadikan objek musuh bergerak dan menyerang dengan pola tertentu. Polanya dibentuk dengan menggunakan suatu algoritma tertentu pula. Pada permainan Asteroid algoritma yang digunakan untuk menentukan pola pergerakan pesawat adalah algoritma random. Algoritma random menjadikan objek musuh bergerak secara acak. Walaupun sederhana metode ini masih dapat dikatakan sebagai AI.

Objek musuh diberi basis pengetahuan (*knowledge base*) yang ditanamkan melalui pengkodean dalam tubuh kode programmnya. Dalam *academic* AI hal itu dikenal dengan teknik reasoning. Adapun basis pengetahuannya berisi penerapan dari algoritma random yang kemudian menentukan pola pergerakannya. Algoritma random sebenarnya berisi ungkapan-ungkapan yang menggunakan pustaka dari bahasa pemrograman yaitu *method* random. *Method* random mengacak suatu angka berdasarkan parameter tertentu yang kemudian angka tersebutlah yang menjadi posisi dan arah pergerakan objek musuh.

Adapun skenario pada aplikasi permainan *arcade-shooter* konvensional dapat dilihat pada gambar berikut:



Gambar 3.1 *Activity Diagram* Skenario Permainan *arcade-shooter*

Berikut ini penjelasan mengenai beberapa skenario pada aplikasi permainan *arcade-shooter* beserta beberapa alternatifnya:

Skenario Utama

Nama Metode : Mulai Permainan Baru

Penjelasan : Pengguna menjalankan aplikasai permainan *arcade-shooter*, memulai permainan baru dan memainkannya sampai permainan berakhir.

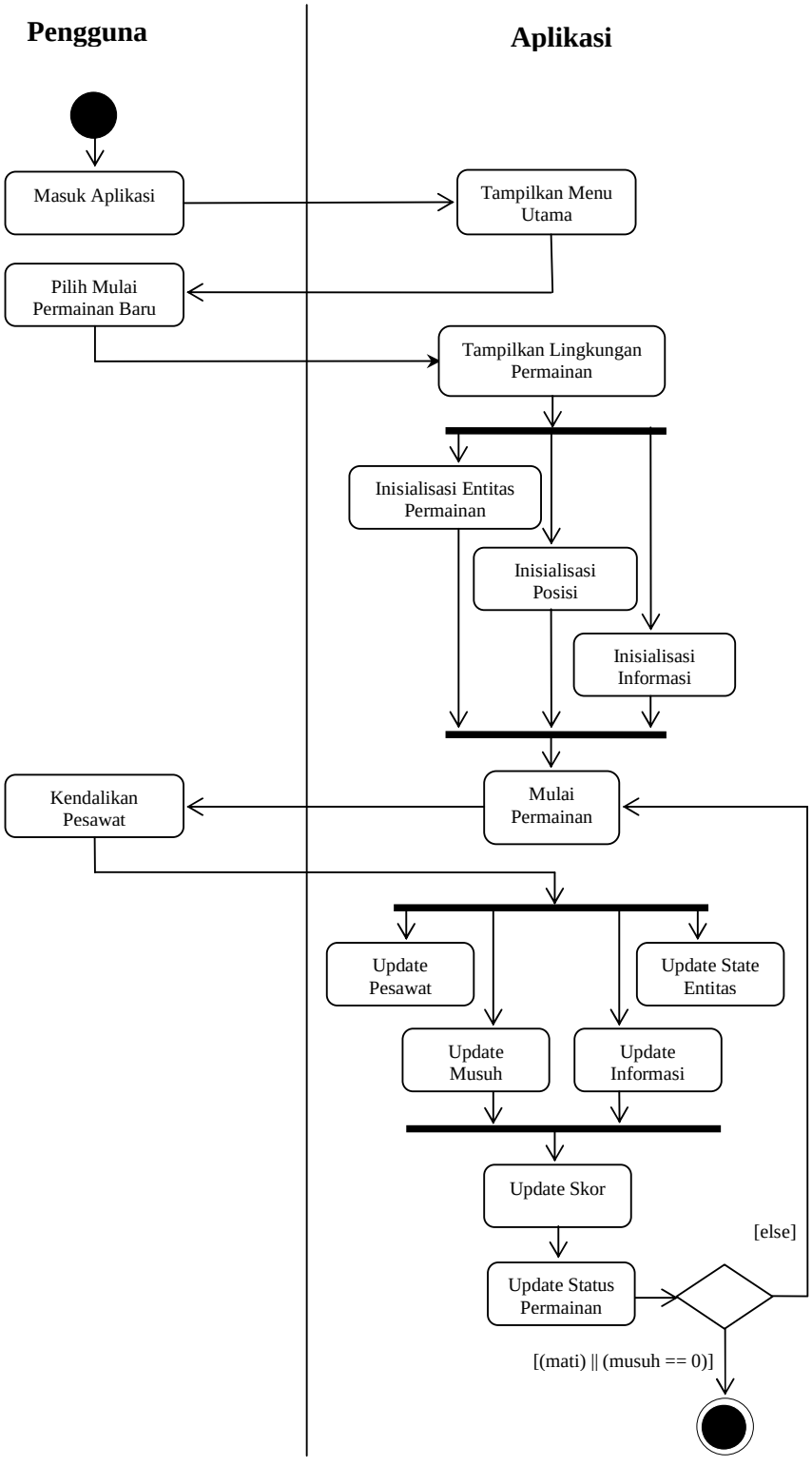
Tujuan Utama : Menghancurkan semua asteroid.

Kondisi Awal : Pengguna memilih mulai permainan baru pada menu utama.

Kondisi Akhir : Semua asteroid hancur.

Tabel 3.9 Skenario Utama Permainan *Arcade-Shooter* Konvensional

Pengguna	Aplikasi
1. Skenario ini dimulai ketika pengguna menjalankan aplikasi permainan *arcade-shooter*.	**2.** Menampilkan menu utama yang padanya terdapat tiga pilihan.
3. Memilih menu Mulai Permainan Baru (*New Game*).	**4.** Menampilkan level permainan.
	5. Menentukan posisi awal pesawat.
	6. Menentukan posisi awal asteroid.
	7. Menampilkan informasi permainan.
	8. Memulai permainan.
9. Mengendalikan pesawat.	**10.** Mengupdate entitas permainan.
	11. Mengupdate posisi musuh.
	12. Mengupdate status entitas.
	13. Mengupdate Informasi.
	14. Mengupdate skor.
	15. Mengupdate status permainan
	16. Skenario ini berakhir ketika semua pemain mati atau musuh habis.



Gambar 3.2 *Activity Diagram* Metode Mulai Permainan Baru

Alternatif Skenario #1

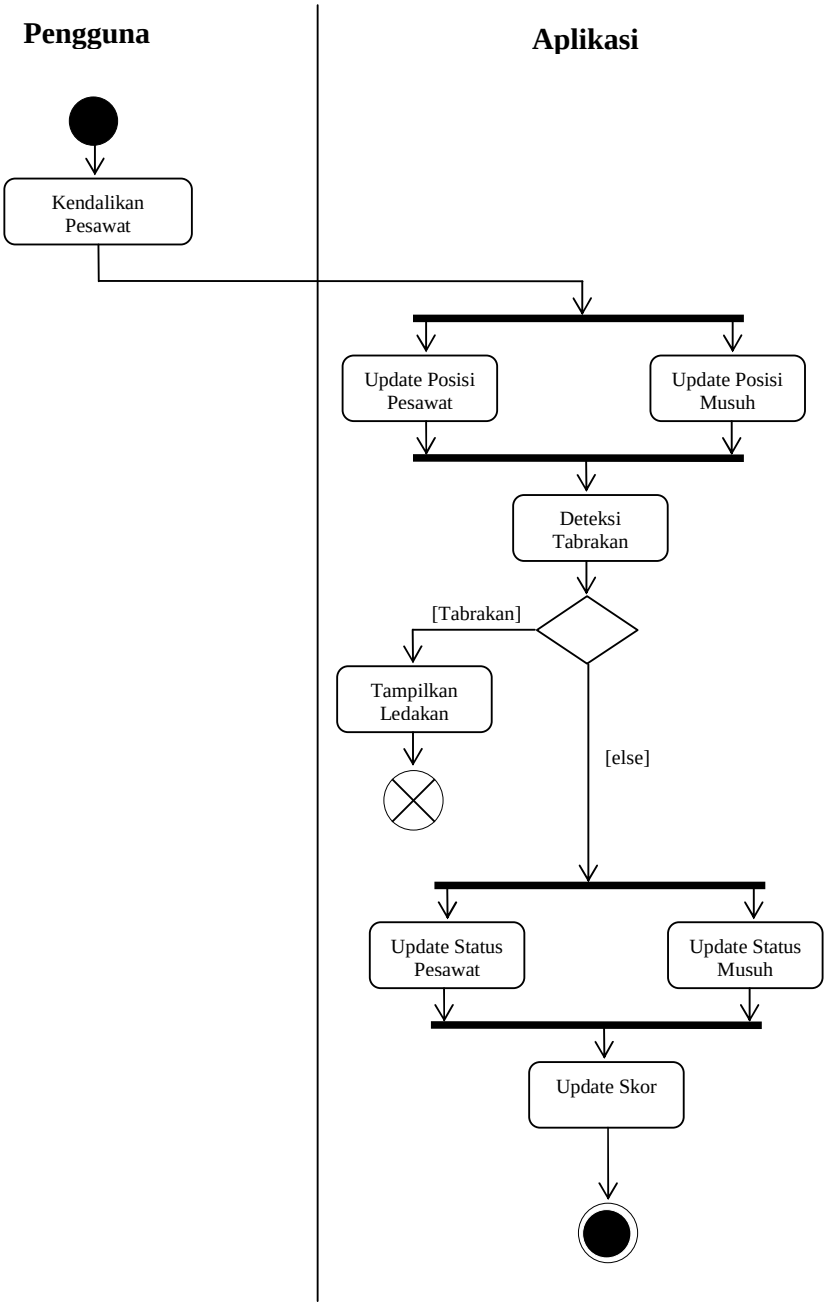
Penjelasan : Pengguna mengendalikan pesawat dan tertabrak musuh.

Kondisi Awal : Pada langkah 9, 10, atau 11 skenario utama, pengguna tidak tangkas mengendalikan pesawat sehingga musuh menabraknya.

Kondisi Akhir : Menampilkan beberapa kemungkinan pada permainan sesuai aturan yang telah ditetapkan. (lihat aturan permainan 1)

Tabel 3.10 Alternatif Skenario #1

Pengguna	Aplikasi
	1. Mengupdate posisi pesawat.
	2. Mengupdate posisi musuh.
	3. Mendeteksi tabrakan (*bounding box* pesawat **intersect** *bounding box* musuh).
	4. Menampilkan tabrakan.
	5. Mengupdate status musuh.
	6. Mengupdate status pesawat.
	7. Mengupdate Skor.
	8. Menampilkan kemungkinan pada permainan (lihat aturan permainan 1)



Gambar 3.3 *Activity Diagram* Alternatif Skenario #1

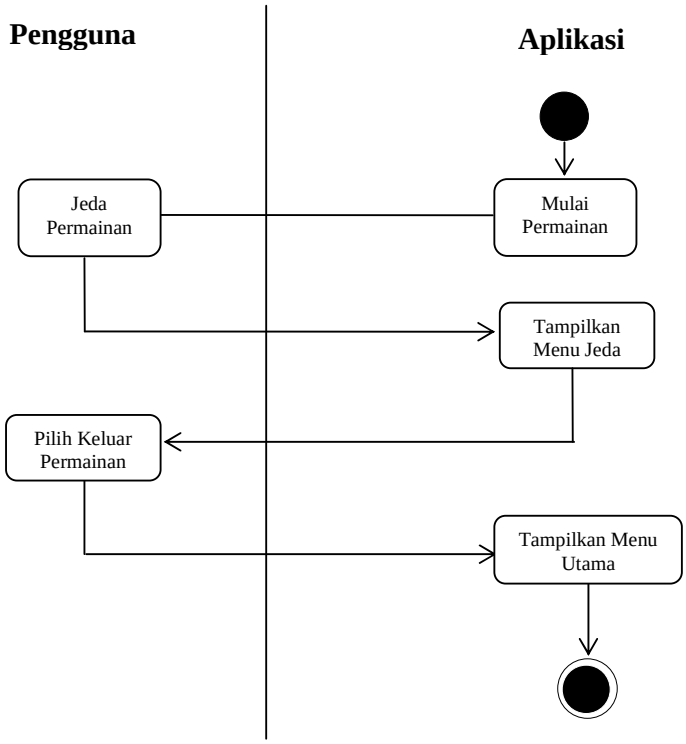
Alternatif Skenario #2

Penjelasan : Pengguna memilih meninggalkan permainan sebelum level berakhir.

Kondisi Awal : Pada langkah 8, 9, 10, 11, 12, 13 atau 14 skenario utama, pengguna menghentikan sementara permainan (*pause*).

Tabel 3.11 Alternatif Skenario #2

Pengguna	Aplikasi
	1. Menampilkan menu jeda (*pause*).
2. Memilih meninggalkan permainan (*quit*).	
	3. Menampilkan pesan peringatan untuk konfirmasi meninggalkan permainan.
4. Melakukan konfirmasi untuk meninggalkan permainan.	**5.** Kembali ke langkah 2 skenario utama.



Gambar 3.4 *Activity Diagram* Alternatif Skenario #2

Alternatif Skenario #3

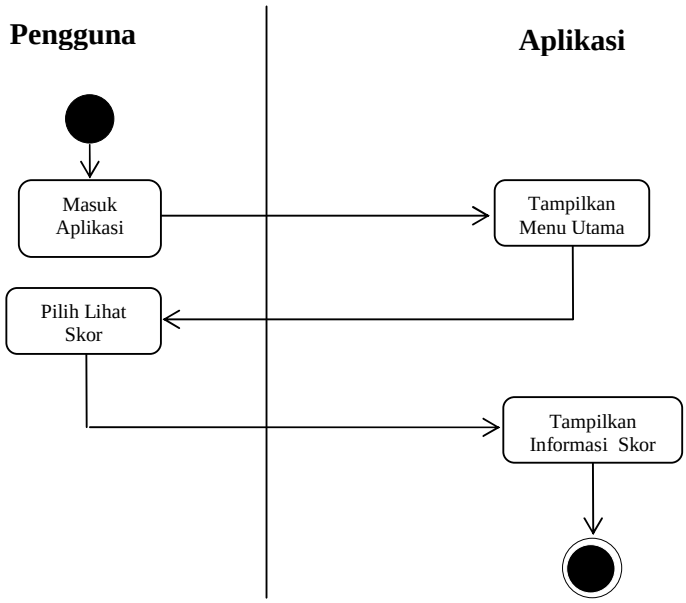
Penjelasan : Pengguna melihat informasi skor pemain.

Kondisi Awal : Pada langkah 3 skenario utama, pengguna memilih menu Lihat Skor Pemain.

Tabel 3.12 Alternatif Skenario #3

Pengguna	Aplikasi
	1. Menampilkan informasi skor pemain.
	2. Menampilkan menu kembali (*back*)
3. Memilih menu kembali (*back*).	**4.** Kembali ke langkah 2 skenario

	utama.



Gambar 3.5 *Activity Diagram* Alternatif Skenario #3

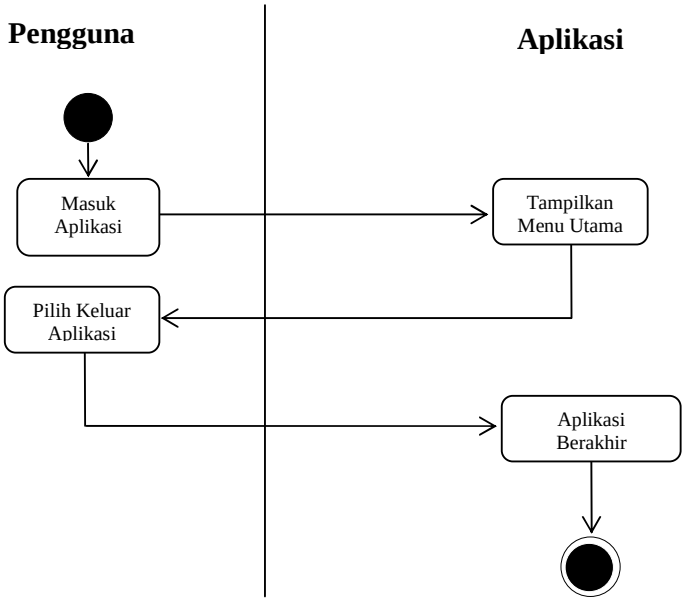
Alternatif Skenario #4

Penjelasan : Pengguna keluar dari aplikasi.

Kondisi Awal : Pada langkah 3 skenario utama, pengguna memilih menu Keluar.

Tabel 3.13 Altetnatif Skenario #4

Pengguna	Aplikasi
	1. Menampilkan pesan peringatan untuk konfirmasi keluar dari aplikasi.
2. Memilih meninggalkan aplikasi	**3.** Berakhir.



Gambar 3.6 Activity Diagram Alternatif Skenario #4

Beberapa metode yang digunakan aplikasi permainan *arcade-shooter* konvensional pada *gameplay* ‘alur permainannya’ dapat dilihat pada tabel berikut:

Metode Pengendalian Objek Pesawat

Penjelasan : Pengguna mengendalikan pesawat dengan menginputkan perintah melalui papan ketik kemudian objek pesawat merespon input perintah tersebut dengan suatu aksi.

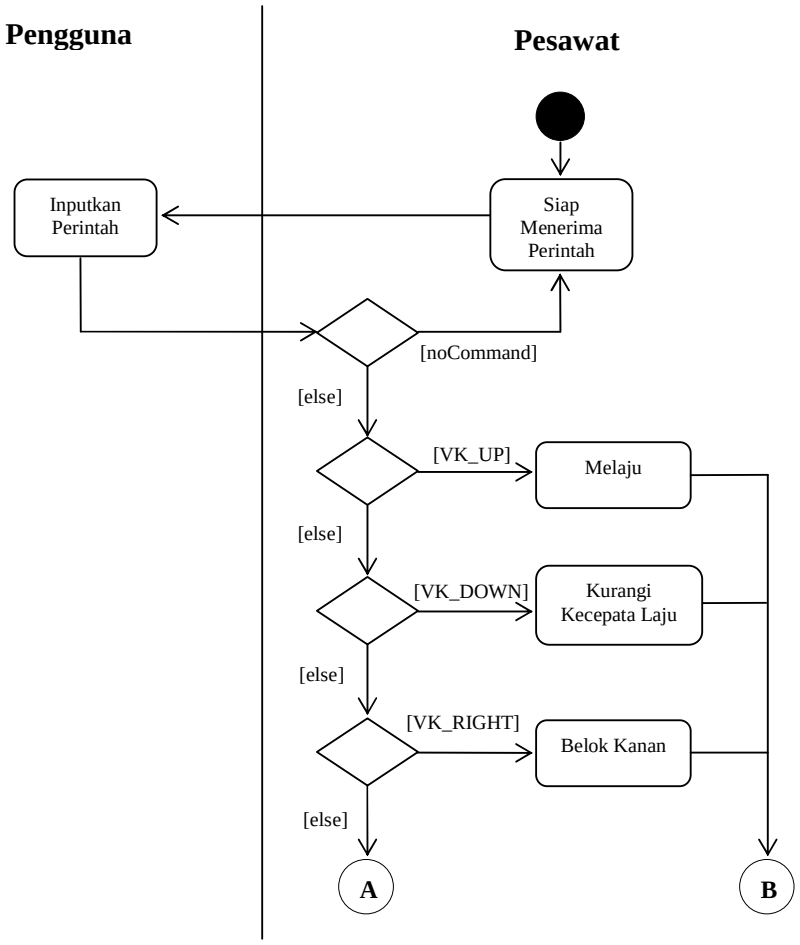
Tujuan : Objek pesawat melakukan aksi sesuai input dari pengguna.

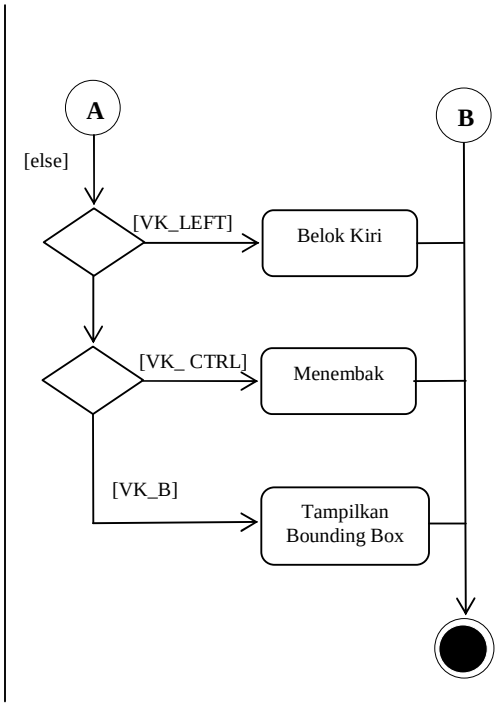
Kondisi Awal : Permainan dimulai dan pengguna memiliki kendali penuh atas objek pesawat.

Kondisi Akhir : Objek pesawat melakukan aksi.

Tabel 3.14 Metode Pengendalian Objek Pesawat

Pengguna	Pesawat
1. Bersiap menginputkan perintah	**2.** Bersiap menerima aksi dari pengguna.
3. Menekan tombol atas (VK_UP)	**4.** Melaju.
5. Menekan tombol bawah (VK_DOWN)	**6.** Mengurangi kecepatan laju.
7. Menekan tombol kanan (VK_RIGHT)	**8.** Berbelok ke kanan.
9. Menekan tombol kiri (VK_LEFT)	**10.** Berbelok ke kiri.
11. Menekan tombol ctrl (VK_CTRL)	**12.** Menembakkan satu peluru.
13. Menekan tombol ctrl (VK_B)	**14.** Menampilkan *bounding box* pesawat.





Gambar 3.7 *Activity Diagram* Metode Pengendalian Objek Pesawat

Metode Deteksi Tabrakan (*Collision Detection*)

Penjelasan : Metode yang digunakan untuk mendeteksi terjadinya tabrakan antara dua objek atau lebih pada lingkungan permainan *arcade-shooter* konvensional.

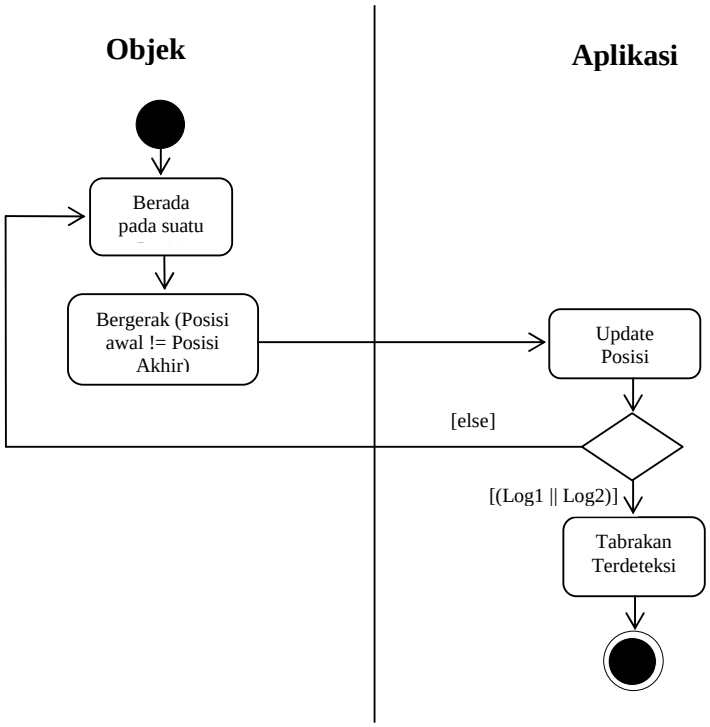
Tujuan : Mendeteksi tabrakan untuk memicu suatu kejadian pada alur permainan. Tabrakan antar dua atau lebih objek yang terjadi diantaranya adalah tabrakan antara peluru dengan musuh, pesawat dengan musuh, musuh dengan musuh.

Kondisi Awal : Setiap objek berada pada posisi tertentu dalam lingkungan permainan.

Kondisi Akhir : Tabrakan terjadi.

Tabel 3.15 Metode Deteksi Tabrakan

Objek 1	Objek 2
1. Berada pada posisi (x1, y1).	**2.** Berada pada posisi (x2, y2).
3. Berpindah posisi (bergerak).	**4.** Berpindah posisi (bergerak).
5. Menabrak objek 2 karena (koordinat y sudut atas objek 1 < koordinat y sudut bawah objek 2 && koordinat y sudut bawah objek 1 > koordinat y sudut atas objek 2) || (koordinat y sudut atas objek 1 < koordinat x sudut bawah objek 2 && koordinat y sudut bawah objek 1 > koordinat x sudut atas objek 2) // logika ini dialiaskan dengan [Log1]	**6.** Menabrak objek 1 karena (koordinat y sudut atas objek 1 > koordinat y sudut bawah objek 2 && koordinat y sudut bawah objek 1 < koordinat y sudut atas objek 2) || (koordinat y sudut atas objek 1 > koordinat x sudut bawah objek 2 && koordinat y sudut bawah objek 1 < koordinat x sudut atas objek 2) // Logika ini dialiaskan dengan [Log2]
7. Tabrakan terjadi	**8.** Tabrakan terjadi



Gambar 3.8 *Activity Diagram* Metode Deteksi Tabrakan

Metode Pergerakan Objek Musuh (*Pattern Based Random Algorithm*)

Penjelasan : Metode yang menerapkan *pattern based* movement pada musuh dengan menggunakan Algoritma *random*.

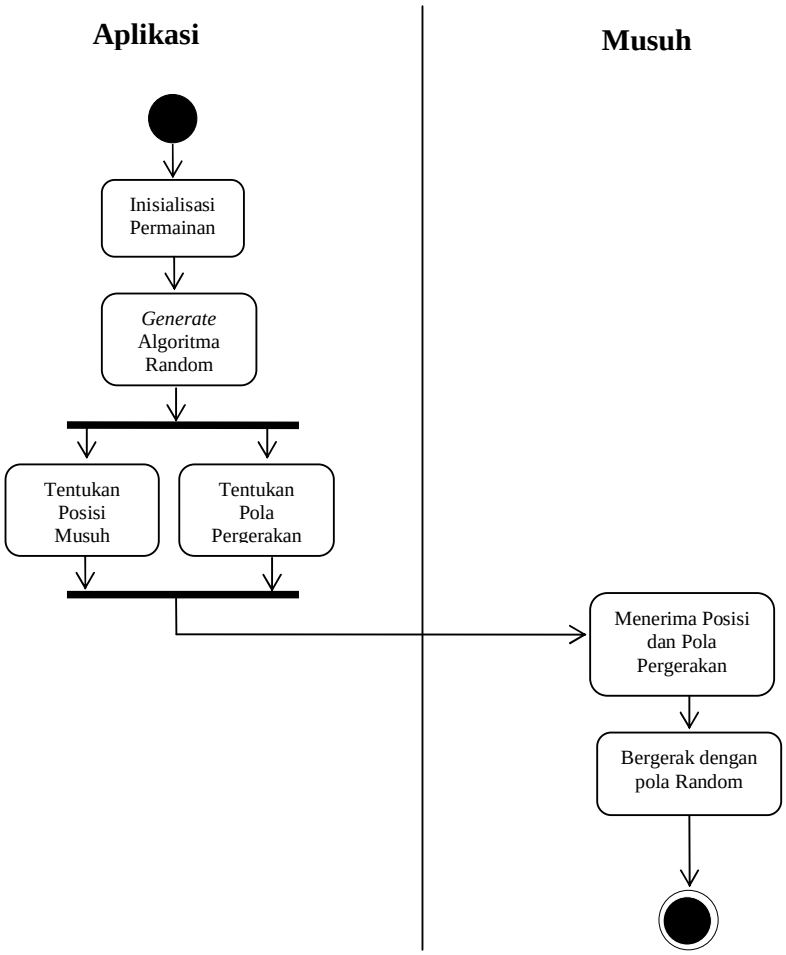
Tujuan : Menanamkan algoritma *random* pada objek musuh sehingga ia dapat bergerak dengan pola acak

Kondisi Awal : Permainan berada pada tahap inisialisasi.

Kondisi Akhir : Musuh bergerak dengan pola acak.

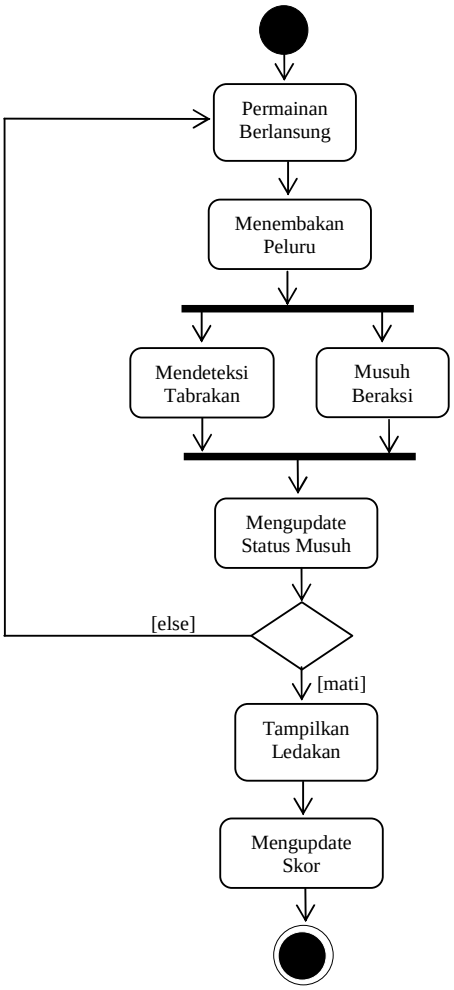
Tabel 3.16 Metode Pergerakan Objek Musuh

Aplikasi	Objek Musuh
1. Memulai tahap inisialisasi permainan.	
2. Menerapkan algoritma random dalam basis pengetahuan musuh dengan menggunakan pustaka random pada bahasa pemrograman dan mengacak nilai untuk : **3.1** Menentukan posisi koordinat objek musuh pada lingkungan permainan. **3.2** Menentukan pola pergerakan objek musuh.	
	3. Menerima posisi dan pola pergerakan.
	4. Bergerak dengan pola acak.



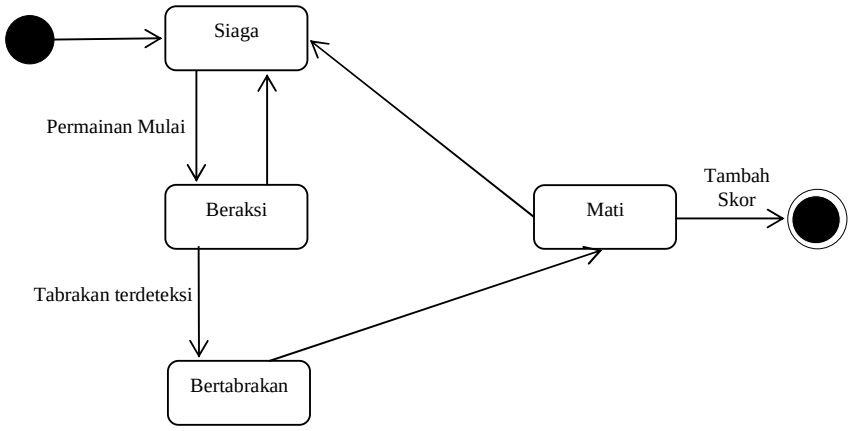
Gambar 3.9 *Activity Diagram* Metode Pergerakan Objek Musuh

Gambar berikut ini merupakan *activity diagram* pada saat pemain menembak musuh :



Gambar 3.10 *Activity Diagram* Metode Menembak Musuh

Adapun *state diagram* yang berlaku pada sebagian entitas dalam permainan arcade-shooter konvensional dapat dilihat pada gambar di bawah ini:



Gambar 3.11 State Diagram Entitas Permainan *Arcade-Shooter* Konvensional

Berdasarkan analisis metode pada permainan *arcade-shooter* konvensional didapatkan beberapa metode yang masih memiliki kelemahan. Hal tersebut dijelaskan pada tabel dibawah ini:

Tabel 3.17 Evaluasi dari Hasil Analisis Metode

Metode	**Kelemahan**
Mulai Permainan Baru	Masih menggunakan tahapan-tahapan seperti dalam aplikasi lainnya. Seharusnya ada modifikasi supaya tidak terkesan monoton dan untuk meningkatkan performa. Penyebab turunnya performa dikarenakan proses inisialisasi dan update pada permainan berada dalam *thread* terpisah.
Alternatif Skenario #1 (Pesawat tertabrak Musuh)	Setelah tabrakan terjadi hanya ditampilkan animasi ledakan saja sehingga objek bersangkutan langsung lenyap. Tentunya hal ini dianggap biasa oleh pengguna yang akibatnya tidak menaikan *game experience* sama sekali.
Pengendalian Objek Pesawat	*Event Handler* hanya dapat menerima satu perintah dalam satu waktu. Akibatnya, pergerakan pesawat terasa kaku dan kasar karena event handler

	tidak mendukung fasilitas penekanan tombol secara bersamaan (*combine key*). Sehingga reflek dari pengguna akan sia-sia.
Deteksi Tabrakan	Setelah terjadi tabrakan hanya menampilkan animasi ledakan saja. Hal itu tidak meningkatkan *gaming experience* bagi pengguna lama.
Pergerakan Objek Musuh	Masih menggunakan pergelakan terpola (*pattern based*) dan algoritma random, sehingga pergerakan musuh terkesan monoton dan mudah diperdiksi. Hal tersebut membuat pengguna akan merasa bosan seriring bertambahnya waktu bermain.

3.2.3.2 Metode yang Dikembangkan

Untuk mengembangkan permainan *arcade-shooter* menjadi lebih menarik dapat dilakukan dengan meningkatkan *gaming experience* bagi pengguna. Game AI merupakan faktor signifikan yang dapat meningkatkan *gaming experience* pada jenis permainan apapun tak terkecuali *arcade-shooter*. Oleh karena itu *game* AI pada permainan tersebut harus diperbaharui untuk mengatasi kelemahan dari permainan *arcade-shooter* konvensional.

Pengembangan metode yang dilakukan akan masih menggunakan teknik *reasoning*, namun basis pengetahuan pada objek musuh akan diperbaharui dengan diterapkannya *game* AI *engine* seperti *finite state machine*, *messaging system*, dan *scripting*. Sehingga objek musuh kini akan cukup cerdas dan mempunyai semangat perlawan yang lebih tinggi. Objek musuh yang dulunya tidak cerdas pun akan terlihat sedikit lebih cerdas jika *game* AI nya diperbaharui. Dengan bertambah gesitnya perlawanan dari musuh, pengguna pun akan merasa lebih tertantang karenanya.

FSM merupakan *game* AI yang dapat diterapkan disemua jenis permainan. FSM cocok digunakan pada permainan *arcade-shooter* karena dapat mengubah *mood* dari objek musuh ketika ia mengetahui bahwa dirinya dalam bahaya. Hal itu menjadikan objek musuh memiliki naluri bertahan hidup seperti manusia atau hewan. Ia akan berubah menjadi ganas jika dalam keadaan kritis atau terancam. Adapun tindakan yang dilakukan dalam bertahan hidup berbeda antara satu jenis musuh dengan yang lainnya. Intinya adalah bahwa penerapan FSM pada objek musuh dapat membuatnya seperti terlihat cerdas.

Selain pada objek musuh, FSM juga dapat diterapkan pada objek lainnya seperti pesawat dan peluru. Adapun pemicu dari perubahan *state*-nya bisa berupa kejadian seperti *collision detection*. Ketika pesawat bertabrakan dengan objek seperti powerup, maka ia akan mendapatkan kemampuan baru atau bonus skor tergantung dari jenis powerup-nya. Pemicu lainnya adalah interaksi pengguna yang mana ketika pengguna menekan tombol shift, maka pesawat akan berada pada *state* shielded. Selain itu pesawat juga memiliki *state* kebal, *state* tersebut terjadi beberapa saat ketika pesawat tertabrak musuh dan meledak.

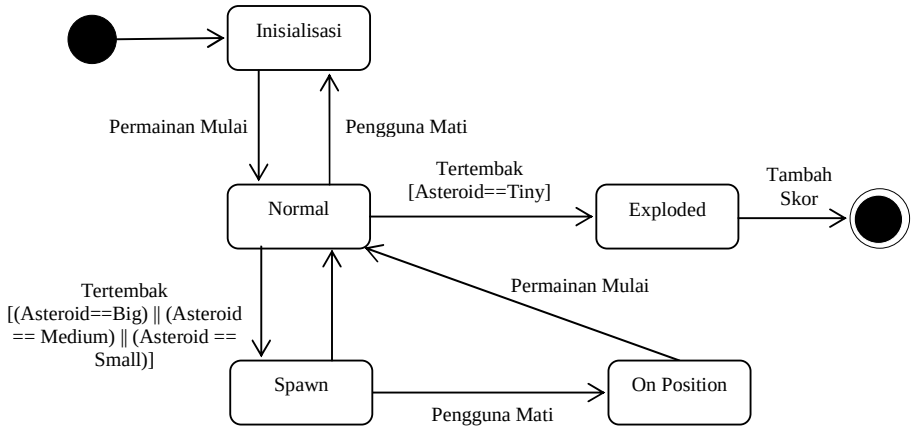
Objek peluru akan berubah *state*-nya ketika pesawat meraih *powerup* berupa amunisi. Adapun perubahan *state*-nya ditandai dengan menigkatnya kapasitas peluru dalam satu waktu sesuai tingkat *upgrade* nya. Pada level 2 pesawat dapat menembakkan dua deret peluru dalam satu waktu, begitu seterusnya hingga level 5 yang mana pesawat dapat menembakkan 6 deret peluru dalam satu waktu. Pada level 5 tersebut ada tambahan satu deret peluru. *State*-ini berakhir ketika pesawat tertabrak oleh musuh walaupun hanya sekali. Akibatnya tingkat *upgrade* peluru

pun akan terus menurun sampai level terbawah yaitu level 1 yang hanya menembakkan sederet peluru dalam satu waktu.

Berikut ini adalah gambaran entitas musuh dan perubahan *state*-nya yang digambarkan pada diagram *state machine*:

a. Spawning Enemy

Musuh jenis ini memiliki kecerdasan yang mampu membuat dirinya berkembang biak setelah tertembak. Maksud berkembang biak disini adalah ketika objek musuh yaitu asteroid tertembak, maka ia akan terpecah menjadi bagian-bagian lebih kecil lagi.



Gambar 3.12 *State Diagram* FSM Spawning Enemy

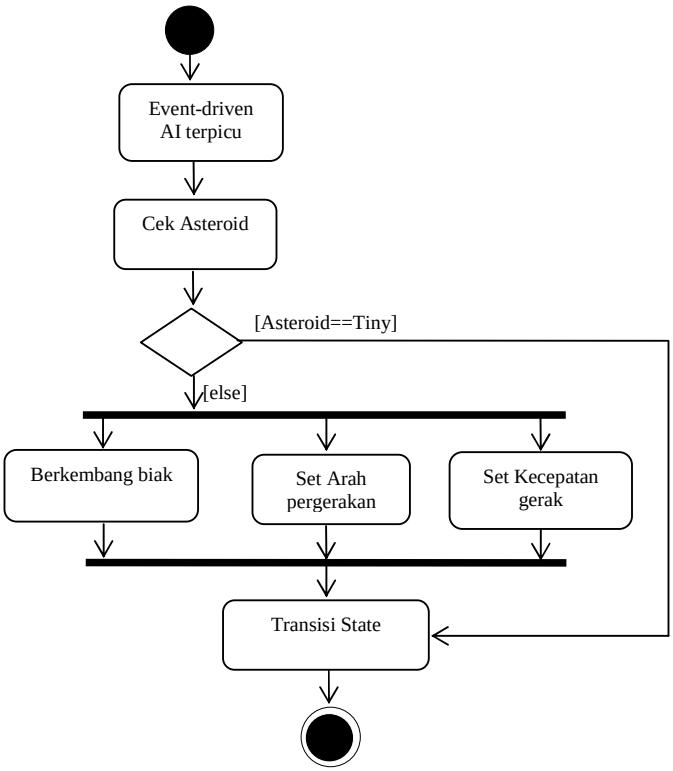
State Spawn yang digambarkan dalam *state diagram* di atas dapat dilihat penjelasan berupa algoritmanya pada tabel di bawah ini:

Tabel 3.18 Algoritma *Spawn State*

Nama Entitas	**Algoritma**
Asteroid	**{**I.S : event penggerak AI engine terpicu}

	{T.S : entitas Asteroid menjalankan basis pengetahuan (kecerdasan yang tertanam padanya) }
	{F.S : On Current State (musuh mati) ‖ Destroy (permainan usai}
	If (Collided(bullets)==true) then {pemicu AIEngine adalah ketika ia tertembak}
	Switch (isAsteroid)
	Case Big:
	Spawn(bigAsteroid)
	STATE_SPAWN
	Break
	Case Medium:
	Spawn(medAsteroid)
	STATE_SPAWN
	Break
	Case Small:
	Spawn(smallAsteroid)
	STATE_SPAWN
	Break
	Case Tiny:
	STATE_EXPLODE
	playExplosion()
	Dead==True
	Break
	if (Dead== true)
	Break
	Endif
	EndIF
	STATE_NORMAL
	}

Adapun *acivity diagram* dari algoritma yang dijelaskan pada tabel di atas dapat dilihat pada gambar di bawah ini:



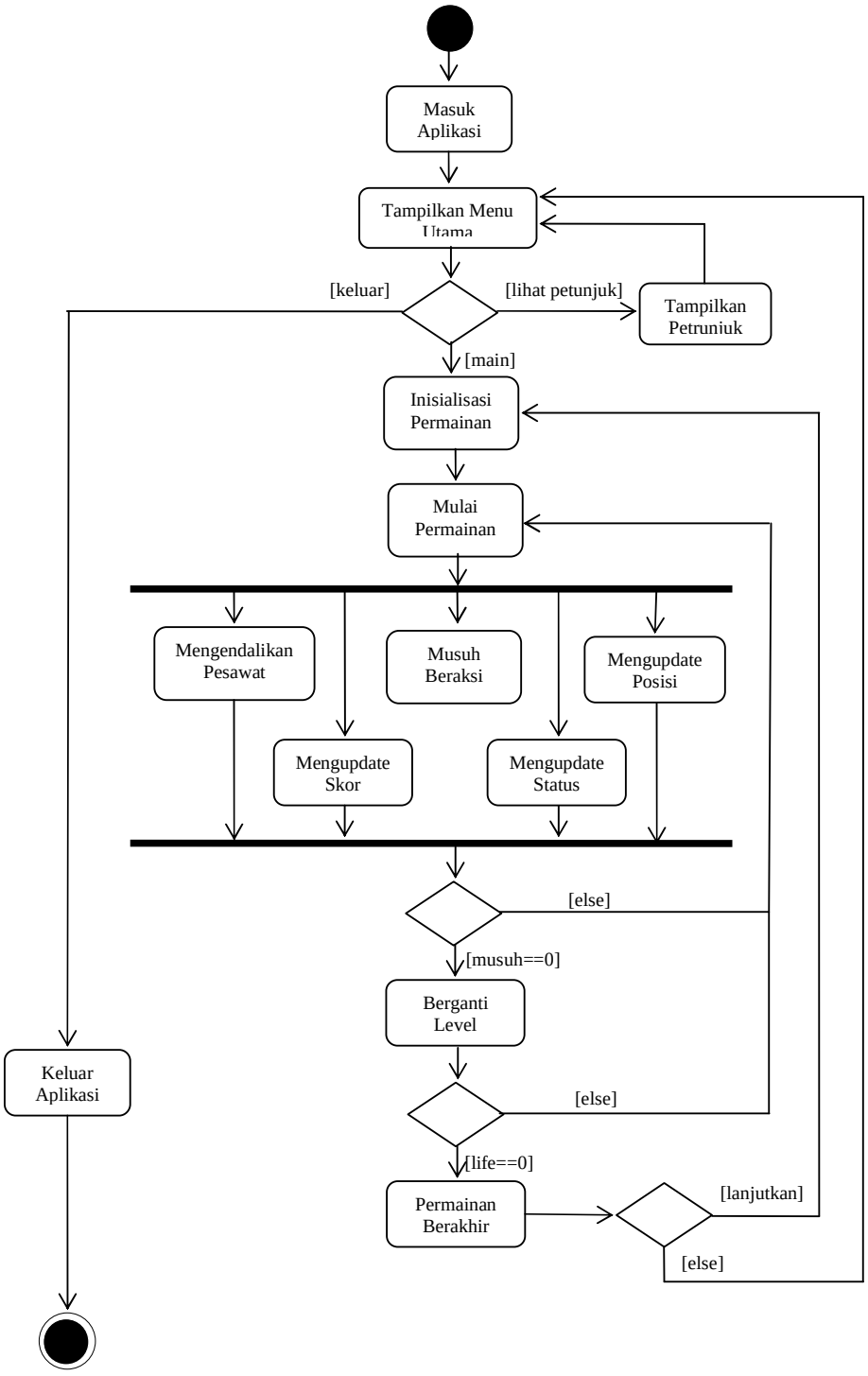
Gambar 3.13 *Activity Diagram* algoritma pada State Spawn Entitas Musuh tipe Asteroid

Scripting sebenarnya adalah sinonim dari teknik *reasoning* pada bidang *academical* AI. Pada aplikasi permainan *arcade-shooter* yang dikembangkan, metode inilah yang menjadi basis pengetahuan dan penghubung antara semua metode AI yang diterapkan. Hal tersebut dikarenakan keseluruhan *game* AI akan disatukan kedalam sebuah *script* yang kemudian akan ditanamkan pada tubuh kode program objek musuh. Penggunaan metode ini dapat menghemat waktu terutama pada tahap pengkodean.

Dengan diperbaharuinya AI pada permainan *arcade-shooter* menggunakan *geme* AI yang telah dibahas, diharapkan dapat meningkatkan *game experience* bagi penggunanya sehingga permianan ini akan menjadi lebih menarik. Adapun

tingkat kecerdasan yang ditanamkan pada objek musuh akan diatur pada tingkatan yang wajar. Karena jika objek musuh terlalu “cerdas” hal tersebut malah akan menjadi boomerang yang akan membuat permainan malah ditinggalkan penggunanya. Itu terjadi karena musuh yang terlalu hebat malah akan membuat penggunanya menjadi frustasi.

Berdasarkan penjelasan diatas, beberapa metode yang ada pada aplikasi permainan *arcade-shooter* konvensional harus dimodifikasi untuk memperbaiki kelemahan-kelemahannya. Gambar-gambar berikut ini merupakan *activity diagram* pada metode yang akan dikembangkan:



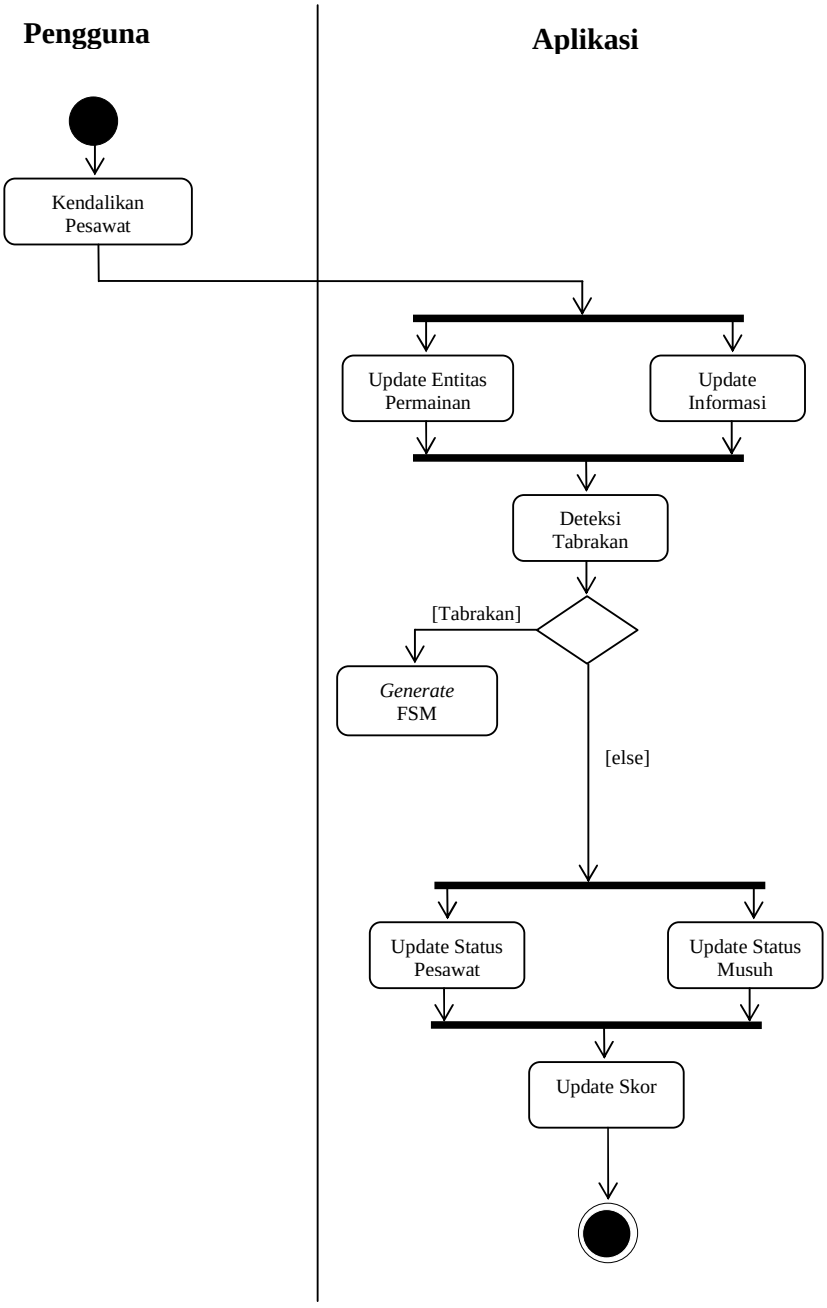
Gambar 3.14 *Activity Diagram* Skenario ArkAId XIV

Pada gambar diatas terlihat bahwa permainan pada aplikasi yang dikembangkan berjalan pada *thread* terpisah. Tentu saja itu berbeda dengan aplikasi lama, tetapi hal itu kini dimungkingkan karena semakin majunya perkembangan perangkat keras komputer. Saat ini, bahkan komputer rumahan pun menggunakan prosesor yang memiliki inti lebih dari satu. Maka dari itu, *multithreading* dapat diterapkan.

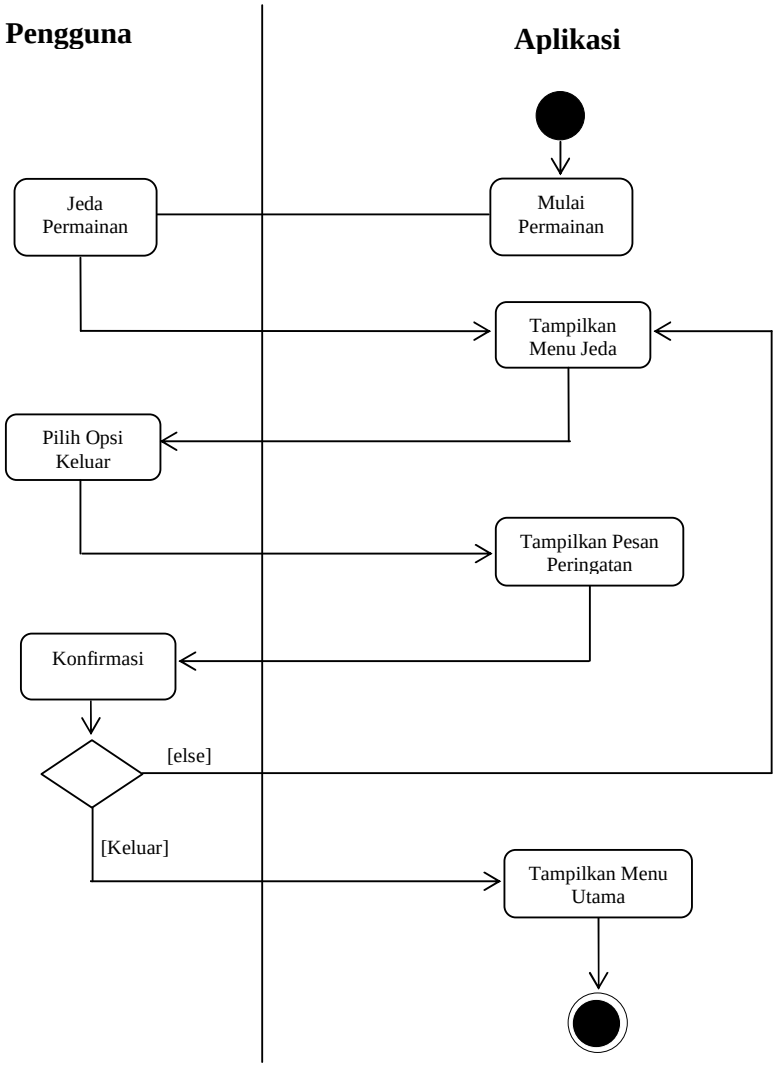
Pengembangan metode pada alternatif skenario #1 dilakukan dengan menambahkan proses aktivasi AI *engine*. Hal itu terjadi ketika tabrakan terdeteksi, yang kemudian kejadian ini menjadi pemicu aktifnya AI *engine* pada objek musuh. Dengan pengembangan ini alur permainan akan berubah dari setelah musuh tertembak ia bukan langsung meledak, melainkan ia akan memicu AI *engine* yang tertanam padanya.

Pengembangan metode pada alterfnatif skenario #2 dan #4 sebenarnya cukup mirip. Pada alur prosesnya ditambahkan proses konfirmasi yang akan memunculkan pesan peringatan kepada pengguna. Sehingga, dengan adanya pesan ini diharapkan peluang terjadinya kejadian aplikasi keluar atau permainan berakhir secara tidak sengaja dapat ditekan.

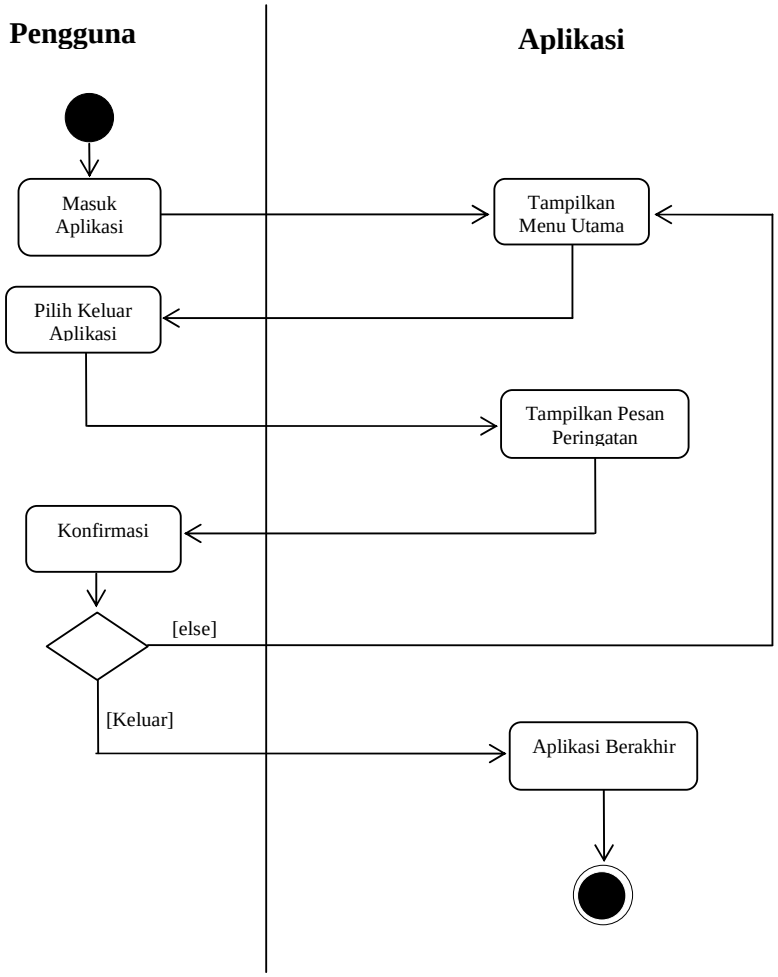
Pengembangna pada metode kontrol permainan adalah dengan menambahkan *template* untuk menampung sekumpulan perintah yang dimasukkan oleh pengguna. Dengan maksud itu, kini aplikasi dapat mendukung kombinasi dari penekanan perintah inputan pengguna.



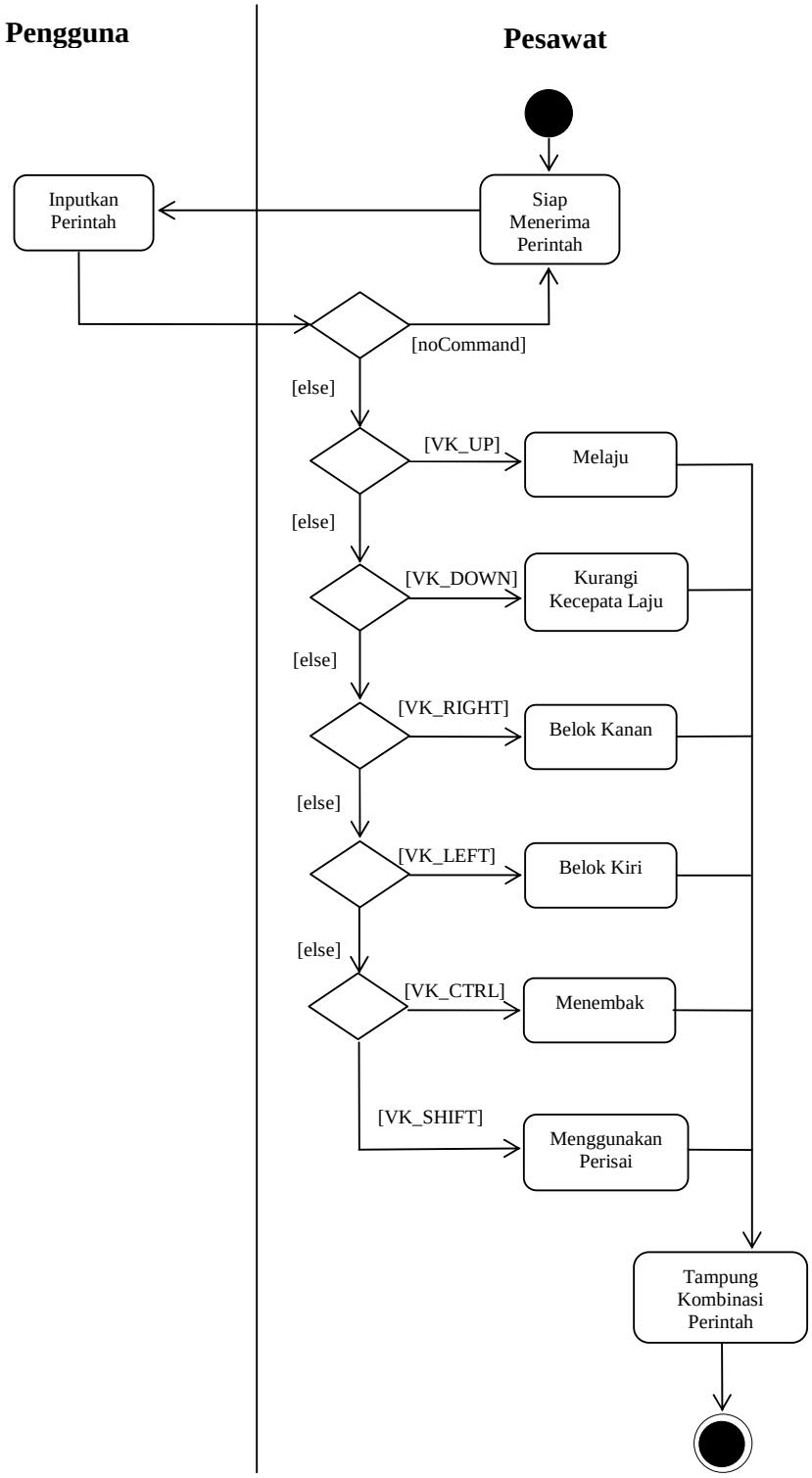
Gambar 3.15 *Activity Diagram* Alternatif Skenario #1 ArkAIdXIV



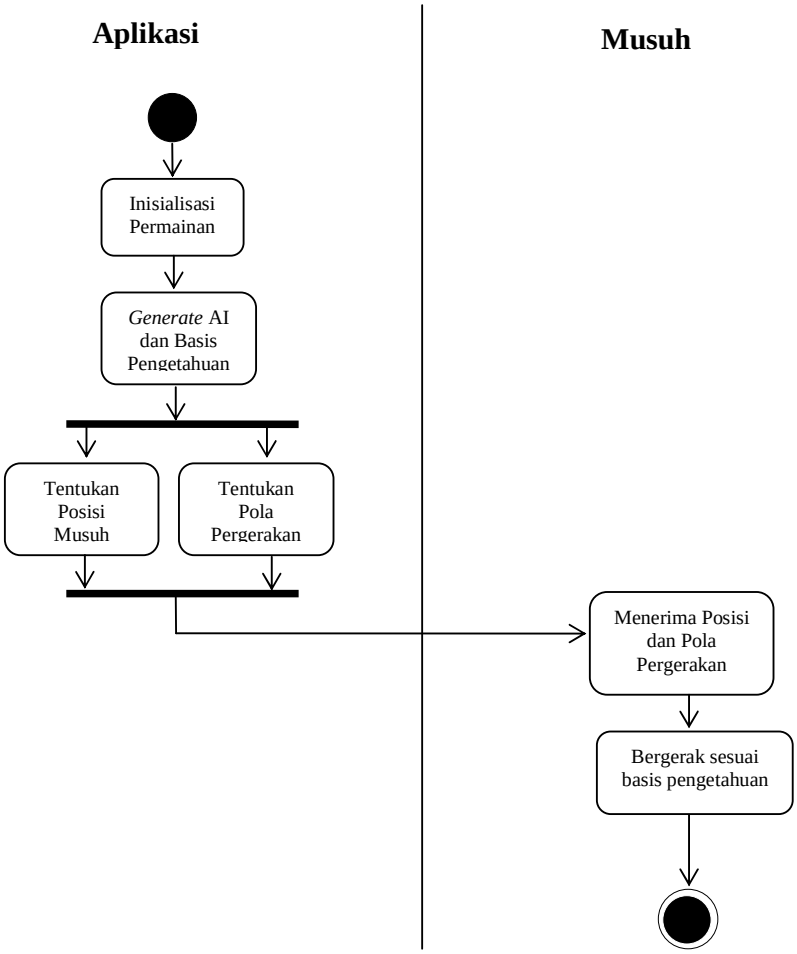
Gambar 3.16 *Activity Diagram* Alternatif Skenario #2 ArkAId XIV



Gambar 3.17 *Activity Diagram* Alternatif Skenario #3 ArkAId XIV

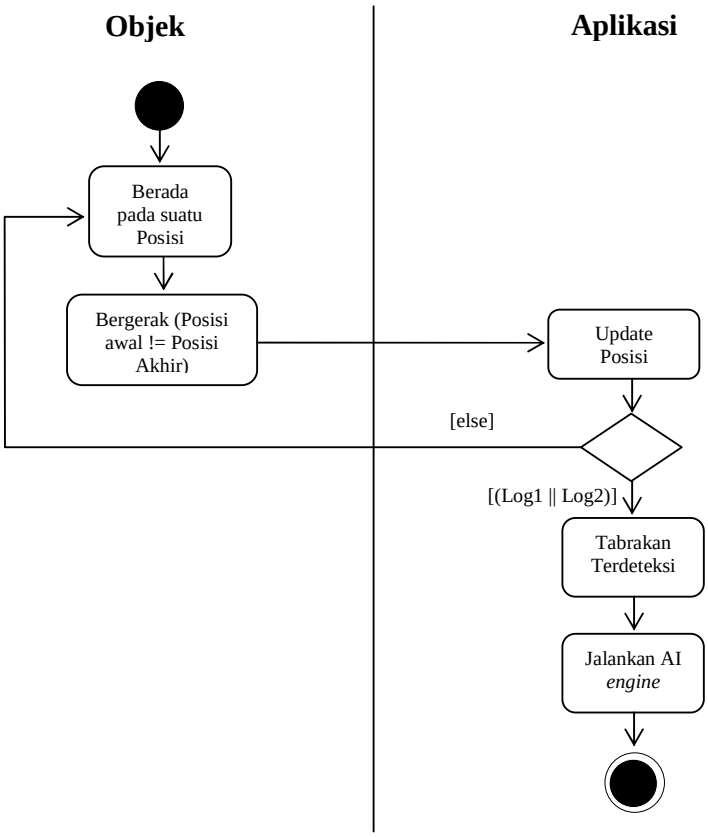


Gambar 3.18 *Activity Diagram* Metode Pengendalian Objek Pesawat



Gambar 3.19 *Activity Diagram* Metode Pergerakan Musuh ArkAId XIV

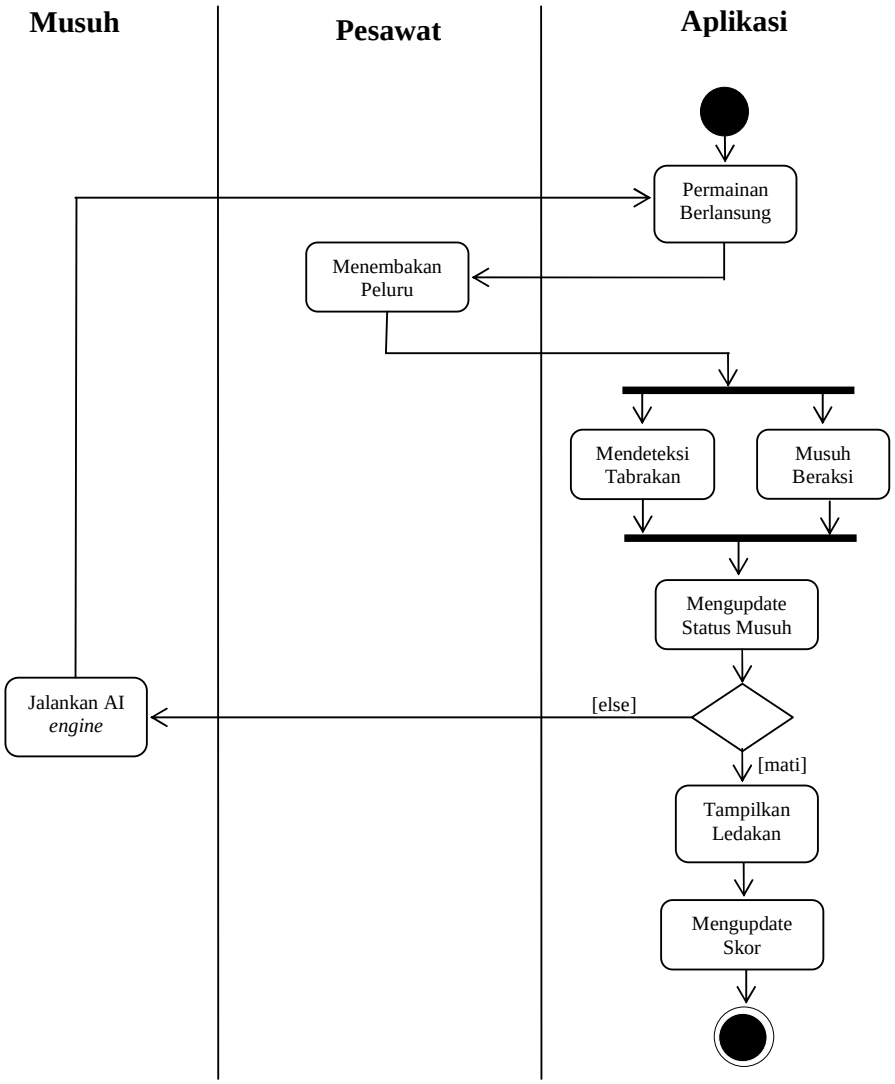
Pada gambar diatas dapat kita lihat bahwa aplikasi yang dikembangkan kini telah mendukung penerapan dari *game* AI. Metode pergerakan musuh dengan algoritma random pada aplikasi lama kini digantikan dengan penanaman *knowledge base* pada objek musuh. Sehingga musuk akan terasa memiliki kecerdasan tersendiri dalam pergerakannya.



Gambar 3.20 *Activity Diagram* Metode *Colliison Detection* ArkAId XIV

metode deteksi tabrakan kini tidak bekerja seperti biasanya. Metode tersebut kini menjadi pemicu AI *engine* yang ada pada objek musuh.

Metode menembak musuh pada aplikasi baru pun mengalami perubahan pada aplikasi yang dikembangkan. Perubahannya hampir sama dengan metode-metode lainnya yaitu dengan melibatkan proses yang akan memicu AI *engine*. Penjelasan akan hal tersebut dapat dilihat pada gambar berikut ini:

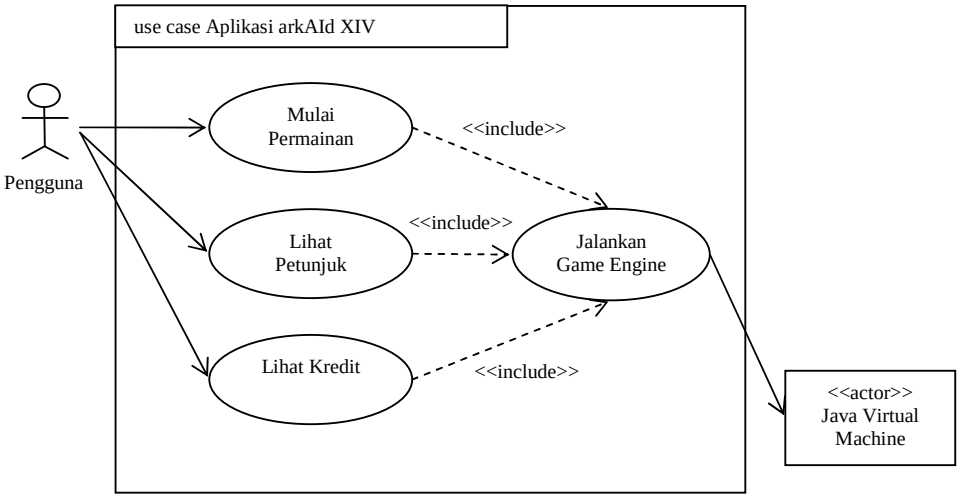


Gambar 3.21 *Activity Diagram* Metode Menembak Musuh ArkAId XIV

3.2.4 Analisis Kebutuhan Fungsional

3.2.4.1 Diagram *Use Case*

Secara umum, interaksi antara pengguna yaitu pemain dengan aplikasi permainan arkAId XIV digambarkan pada diagram *use case* berikut ini:



Gambar 3.22 Diagram *Use Case* ArkAId XIV

3.2.4.2 Skenario *Use Case*

Skenario pada diagram *use case* berisi mengenaik alur kejadian '*flow of event*' *use case* utama. Di dalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut dari awal sampai akhir. Adapun skenario *use case* yang terdapat pada aplikasi arkAId XIV ini adalah sebagai berikut:

1. **Skenario *Use Case* Mulai Permainan**

*S*kenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* mulai permainan yang dijelaskan pada tabel berikut ini:

Tabel 3.19 Skenario *Use Case* Mulai Permainan

Identifikasi		
Nama	Mulai Permainan	
Tujuan	Masuk ke arena permainan kemudian mulai bermain	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Klik Mulai Permainan		
		b. Menampilkan Arena Permainan
Kondisi akhir	Permainan dimulai	

2. Skenario *Use Case* Lihat Petunjuk

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* lihat petunjuk yang dijelaskan pada tabel berikut ini:

Tabel 3.20 Skenario *Use Case* Lihat Petunjuk

Identifikasi	
Nama	Lihat Petunjuk
Tujuan	Masuk ke submenu yang berisi mengenai petunjuk permainan
Aktor	Pemain
Skenario Utama	

Kondisi awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Klik Lihat Petunjuk		
		b. Masuk ke submenu lihat petunjuk
Kondisi akhir	Menampilkan informasi mengenai petunjuk bermain	

3. Skenario *Use Case* Lihat Kredit

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan seluruh use case lihat kredit yang dijelaskan pada tabel berikut ini:

Tabel 3.21 Skenario *Use Case* Lihat Kredit

Identifikasi		
Nama	Lihat Kredit	
Tujuan	Masuk ke submenu yang berisi informasi tentang aplikasi	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Klik Lihat Kredit		
		b. Masuk ke submenu kredit
Kondisi akhir	Menampilkan informasi tentang aplikasi ArkAId XIV	

4. Skenario *Use Case* Jalankan Game Engine

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu Java Virtual Machine, dengan seluruh *use case* yang meng-*include*-nya dan dijelaskan pada tabel berikut ini:

Tabel 3.22 Skenario *Use Case* Jalankan Game Engine

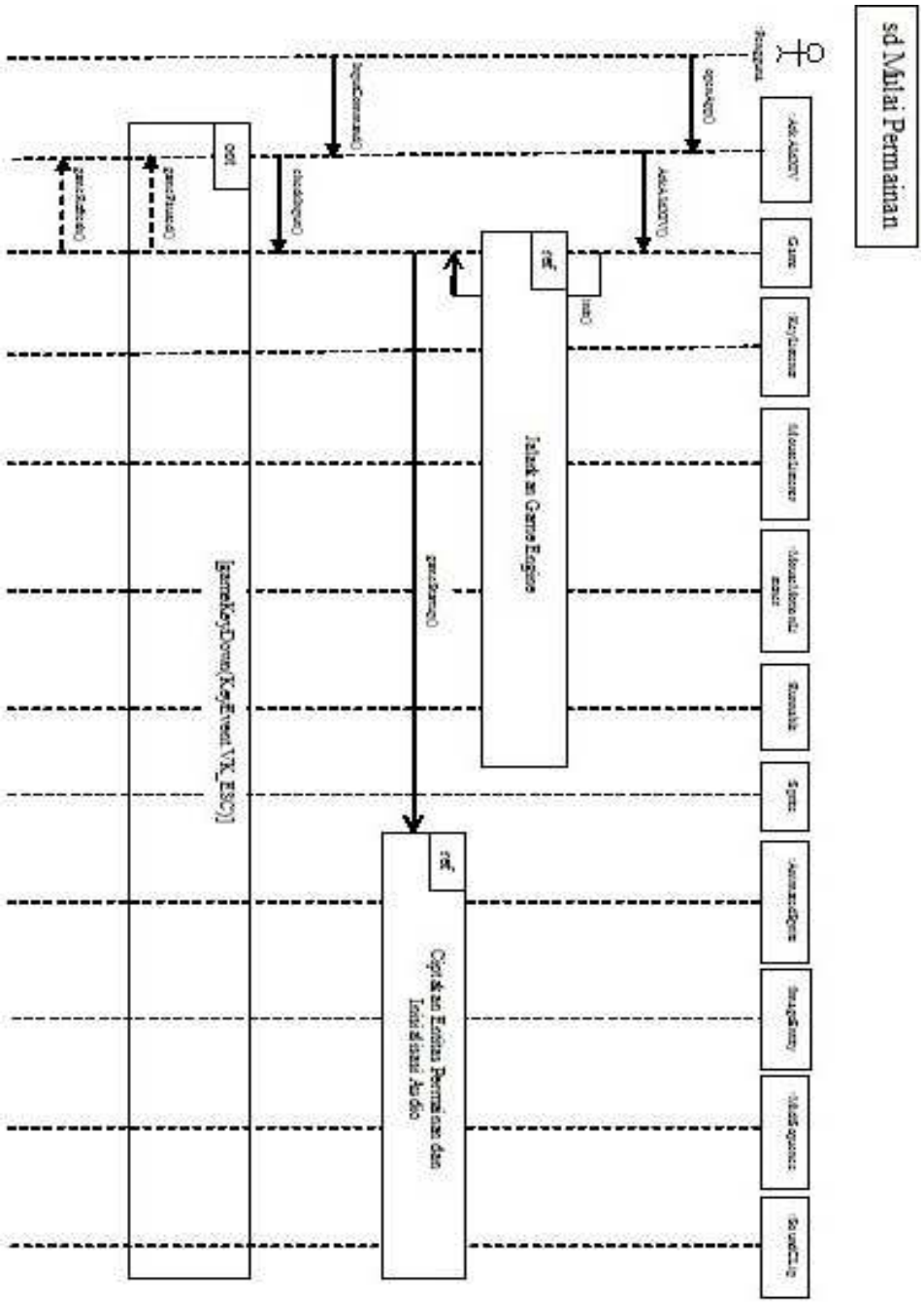
Identifikasi		
Nama	Jalankan Game Engine	
Tujuan	Menjalankan Game Engine	
Aktor	Java Virtual Machine	
Skenario Utama		
Kondisi awal	Java Virtual Machine dalam Kondisi Siaga	
Aksi Aktor		**Reaksi Sistem**
a. Aktivasi Java Runtime Environment		
		b. Game Engine Berjalan
Kondisi akhir	ArkAId XIV Berjalan	

3.2.4.3 Diagram *Sequence*

Pada bagian ini berisi penjelasan dari masing-masing *use case* yang ada pada ArkAId XIV dengan diagram *sequence*. Melalui diagram ini dapat diketahui bagaimana setiap kelas objek pada aplikasi permainan bekerja bersama-sama dengan berinteraksi melalui pengiriman pesan.

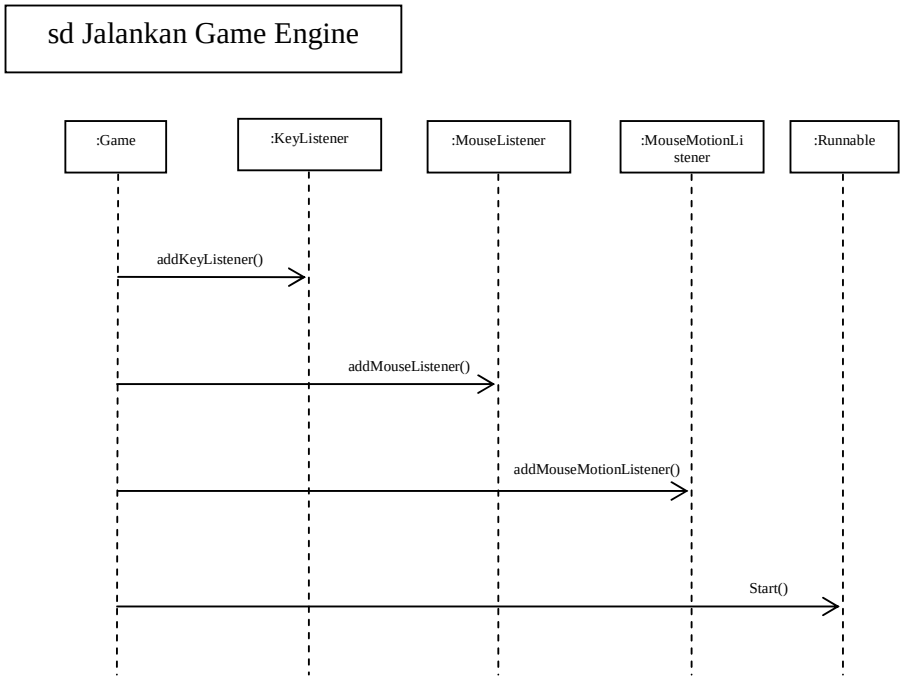
3.2.4.3.1 Diagram *Sequence* Mulai Permainan

Berikut ini merupakan diagram *sequence* dari *use case* Mulai Permainan:



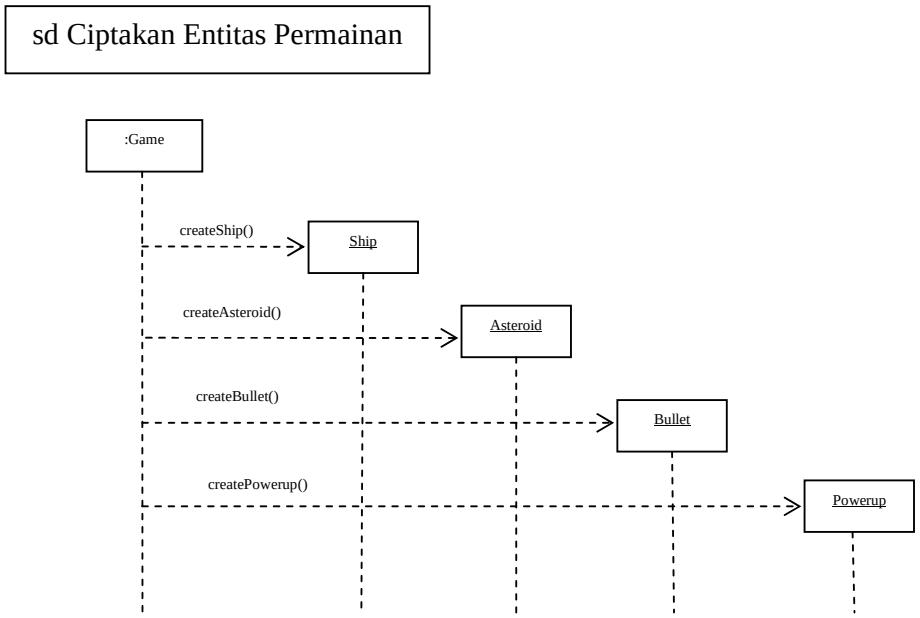
Gambar 3.23 Diagram *Sequence* Mulai Permainan Aplikasi ArkAId XIV

3.2.4.3.2 Referensi Diagram *Sequence* Jalankan Game Engine



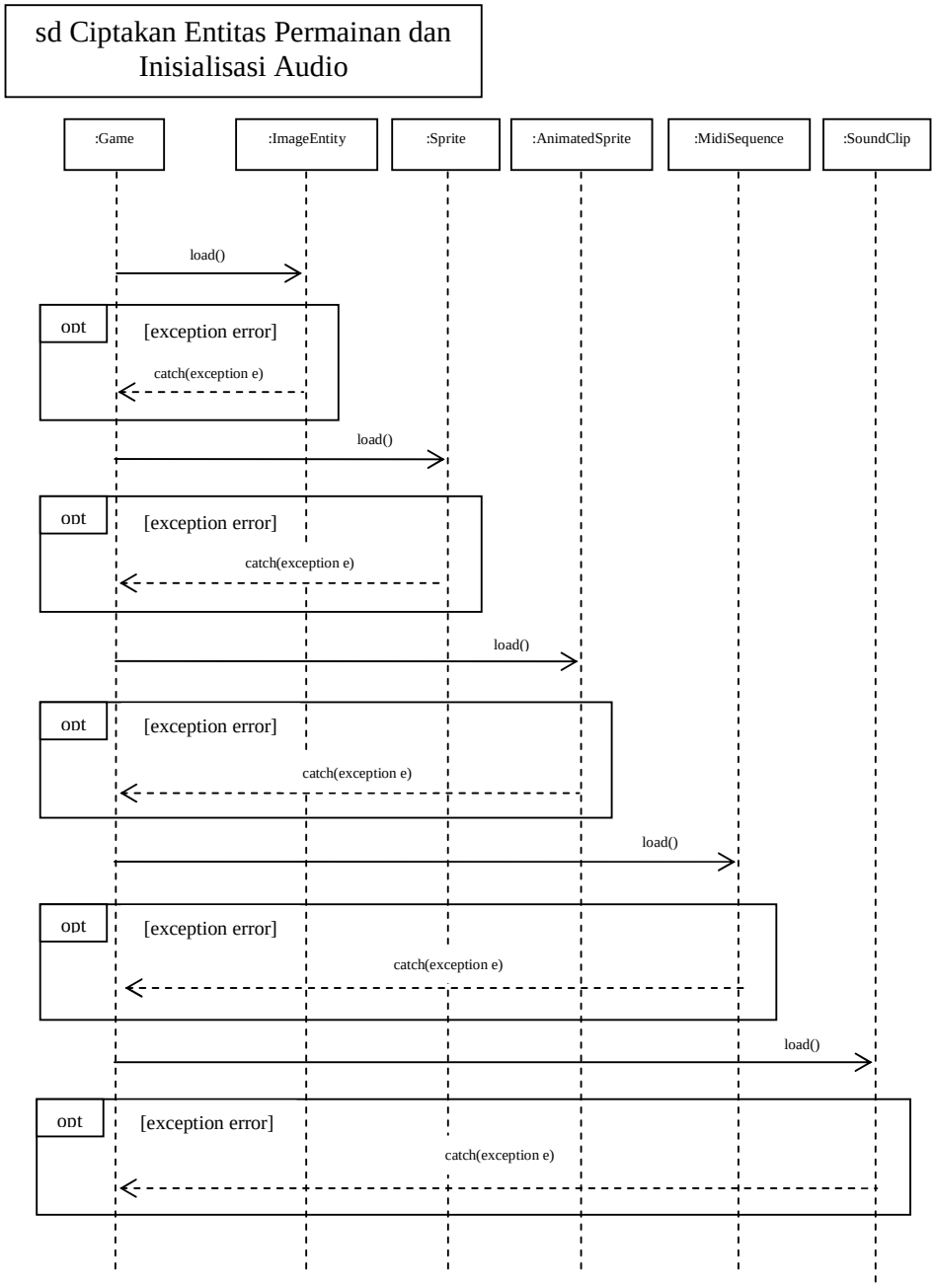
Gambar 3.24 Referensi Diagram *Sequence* Jalankan Game Engine

3.2.4.3.3 Referensi Diagram *Sequence* Ciptakan Entitas Permainan



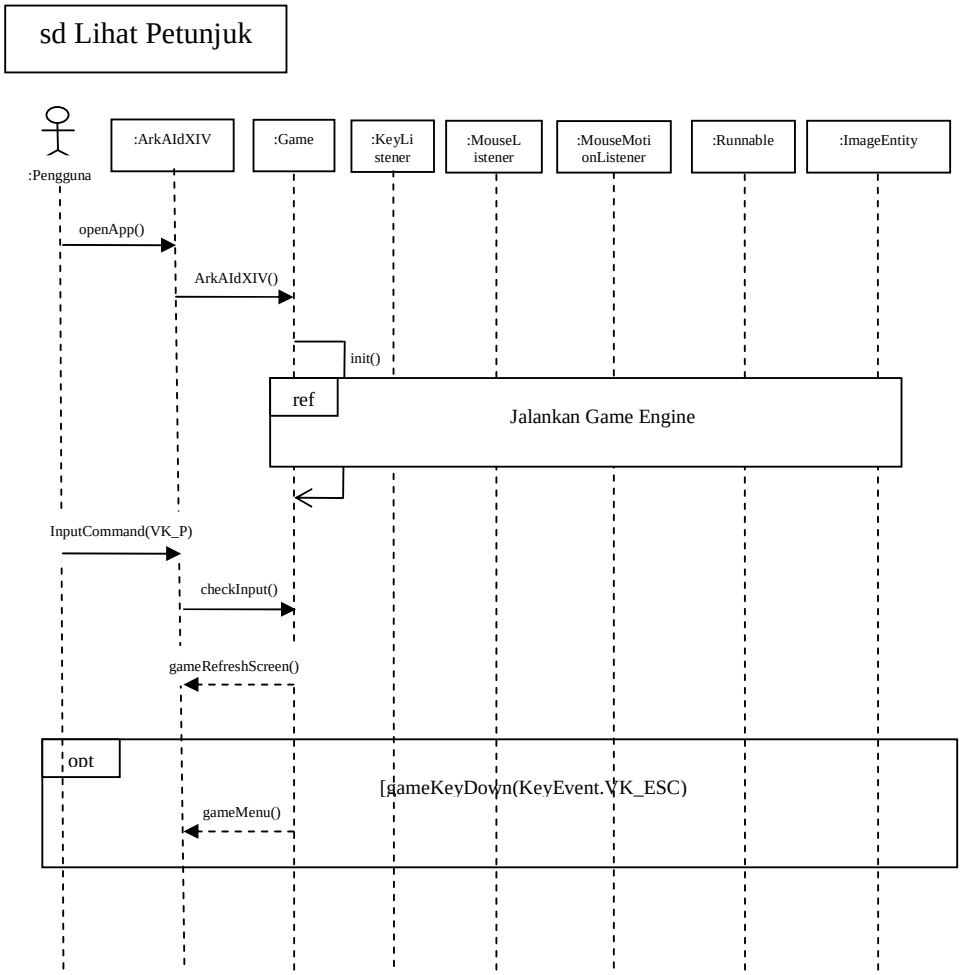
Gambar 3.25 Referensi Diagram *Sequence* Ciptakan Entitas Permainan

3.2.4.3.4 Referensi Diagram *Sequence* Ciptakan Entitas Permainan dan Inisialisasi Audio



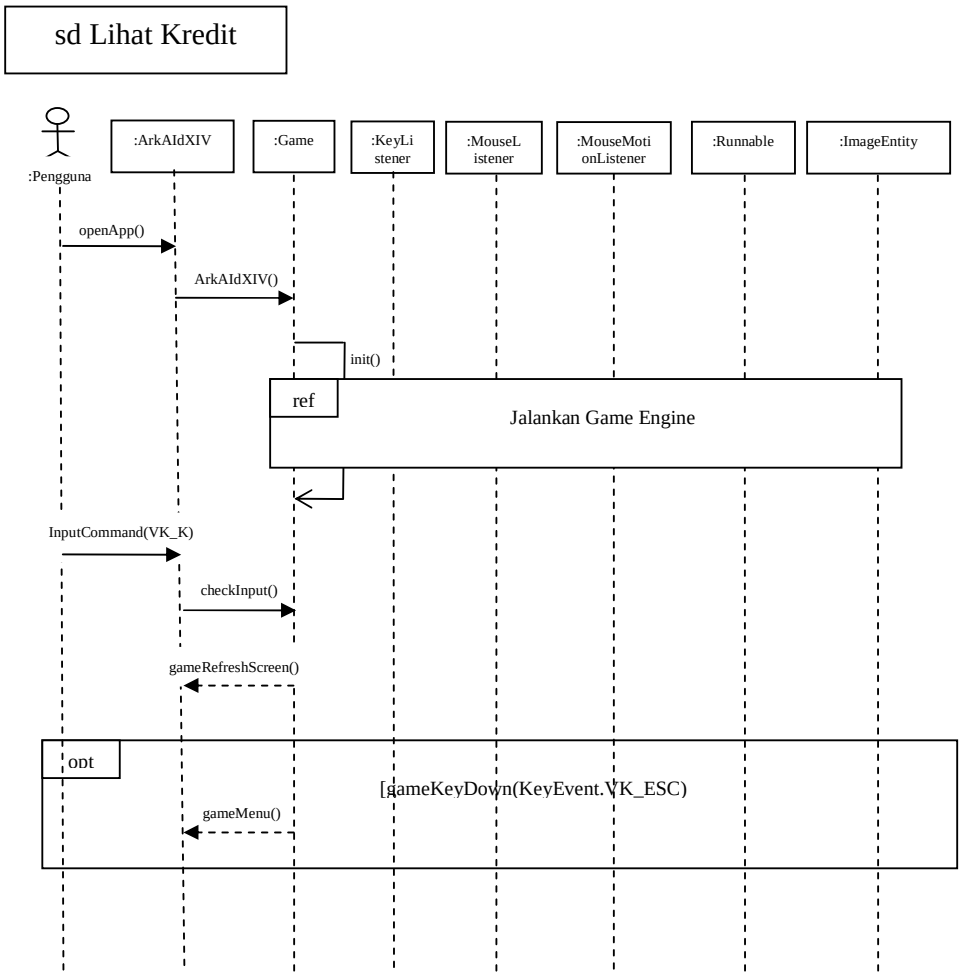
Gambar 3.26 Referensi Diagram Sequence Ciptakan Entitas Permainan dan Inisialisasi Audio

3.2.4.3.5 Diagram *Sequence* Lihat Petunjuk



Gambar 3.27 Diagram *Sequence* Lihat Petunjuk

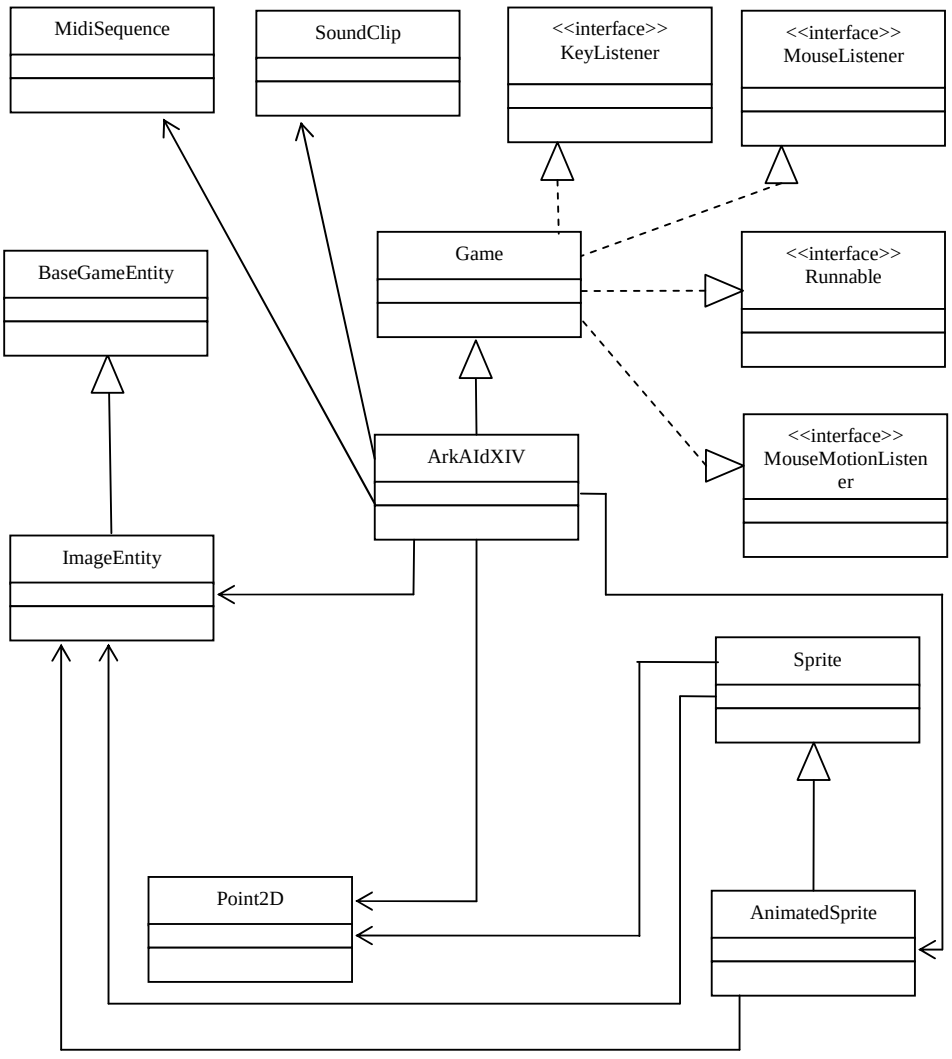
3.2.4.3.6 Diagram *Sequence* Lihat Kredit



Gambar 3.28 Diagram *Sequence* Lihat Kredit

3.2.4.4 Diagram *Class*

Diagram *Class* pada aplikasi ArkAId XIV dapat dilihat pada gambar berikut ini :



Gambar 3.29 Diagram *Class* AkAId XIV

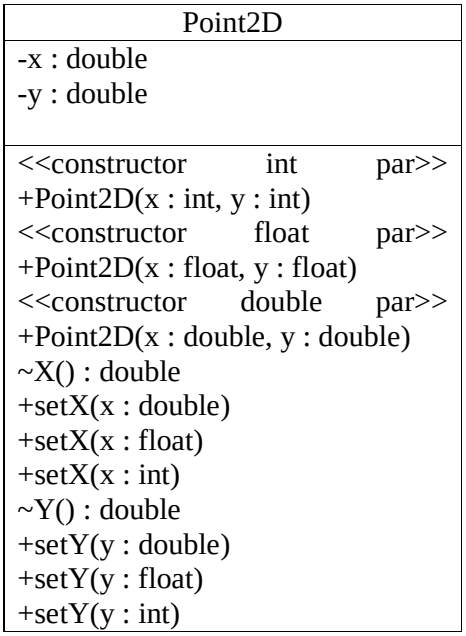
Adapun penjelasan mengenai anggota-anggota kelas yang terlibat ada pada serangkaian gambar berikut:

SoundClip
-sample : AudioInputStream -clip : Clip +looping : Boolean +repeat : int -filename : String +isLoaded() : boolean
+setLooping(_looping : boolean) +getLooping() : Boolean +setRepeat(_repeat : int) +getRepeat() : int +setFilename(_filename : String) +getFilename() : String <<constructor>> +SoundClip() <<overloaded constructor>> +SoundClip(filename : String) -getURL(filename : String) : URL +load(audiofile : String) : boolean +play() : void +stop() : void

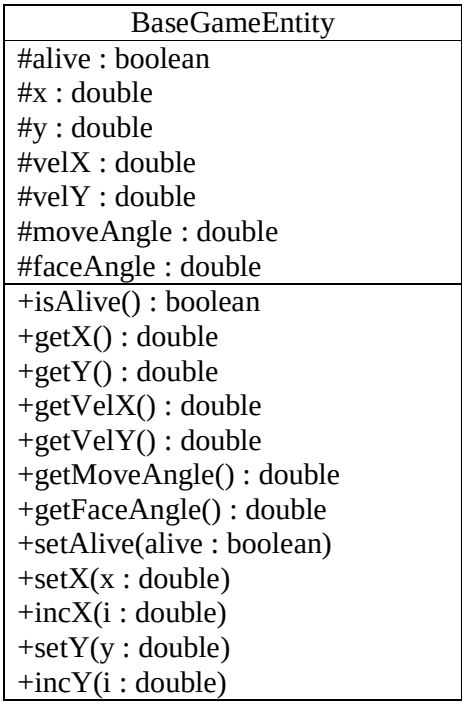
Gambar 3.30 SoundClip *Class* pada *Class Diagram* ArkAId XIV

MidiSequence
-sequencer : Sequencer -song : Sequence -filename : String -looping : Boolean -repeat : int
+getSong() : Sequence +getFilename() : String +getLooping() : boolean +setLooping(_looping : boolean) +setRepeat(int _repeat) +getRepeat() : int +isLoaded() : boolean <<constructor>> +MidiSequence() <<constructor overloaded>> +MidiSequence(filename : String) +getURL() : URL +load(midifile : string) : boolean +play() : void +stop() : void

Gambar 3.31 MidiSequence *Class* pada *Class Diagram* ArkAId XIV



Gambar 3.32 Point2D *Class* pada *Class Diagram* ArkAId XIV



+setVelX(i : double) +incVelX(i : double) +setvelY(i : double) +incVelY(i : double) +setFaceAnle(angle : double) +incFaceAngle(i : double) +setMoveAngle(angle : double) +incMoveAngle(i : double) <<constructor>> +BaseGameEntity()

Gambar 3.33 BaseGameEntity *Class* pada *Class Diagram* ArkAId XIV

ImageEntity
#image : Image #applet : Applet #at : AffineTransform #g2D : Graphics2D
<<constructor>> +ImageEntity(a : Applet) +getImage() : Image +setImage(image : Image) +width() : int +height() : int +getCenterX : double +getCenterY : double +setGraphics(Graphics2D g) -getURL(filename : String) : URL +load(filename : String) +transform() : void +draw() : void +getBound() : Rectangle

Gambar 3.34 ImageEntity *Class* pada *Class Diagram* ArkAId XIV

Sprite
-entity : ImageEntity #pos : Point2D #vel : Point2D #rotRate : double #currentState : int #sprType : int #_collided : boolean #_lifespan : boolean

#_lifeage : boolean
<<constructor>> +Sprite(a : Applet, g2d : Graphics2D g2d) +load() : void +transform() : void +draw() : void +drawBounds(c : Color) +updatePosition() : void +rotationRate() : double +setRotationRate(rate : double) +updateRotation() : void +state() : int +setState(state : int) +getBounds() : Rectangle +position() : Point2D +setPosition(pos : Point2D) +velocity() : Point2D +setVelocity(vel : Point2D) +center : Point2D +alive() : boolean +setAlive(alive : boolean) +faceAngle() : double +setFaceAngle(angle : double) +setFaceAngle(angle : float) +setFaceAngle(angle : int) +moveAngle() : double +setMoveAngle(angle : double) +setMoveAngle(angle : float) +setMoveAngle(angle : int) +imageWidth() : int +imageHeight() : int +collidesWith(rect : Rectangle) : boolean +collidesWith(sprite : Sprite) : boolean +collidesWith(point : Point2D) : boolean +applet() : Applet +graphics() : Graphics2D +image() : Image +setImage(image : Image) +spriteType() : int +setSpriteType(type int) +collided() : boolean +setCollided(collide : boolean) +lifespan() : int

+setLifespan(life : int) +lifeage() : int +setlifeAge(age : int) +updateLifetime()

Gambar 3.35 Sprite *Class* pada *Class Diagram* ArkAId XIV

AnimatedSprite
-animImage : ImageEntity ~tempImage : BufferedImage ~tempSurface : Graphics2D -currFrame : int -totFrames : int -animDir : int -frCount : int -frDelay : int -frWidth : int -frHeight : int -cols : int
<<constructor>> +AnimatedSprite(applet : Applet, g2d : Graphics2D) +load(filename : String, columns : int, rows : int, width : int, height : int) +currentFrame() : int +setCurrentFrame(frame : int) +frameWidth() : int +setFrameWidth(width : int) +frameHeight() : int +setFrameHeight(height : int) +totalFrames() : int +setTotalFrames(total : int) +animationDirection() : int +setAnimationDirection(dir : int) +frameDelay() : int +setFrameDelay(delay : int) +columns() : int +setColumns(num : int) +getAnimImage() : Image +setAnimImage(image : Image) +updateAnimation() : void +updateFrame() : void

Gambar 3.36 AnimatedSprite *Class* pada *Class Diagram* ArkAId XIV

Game
-gameloop : Thread -_sprites : LinkedList -backbuffer : BufferedImage -g2d : Graphics2D -screenWidth : int -screenHeight : int -mousePos : Point2D -mouseButtons[] : boolean -_frameCount = 0 : int -_frameRate = 0: int -desiredRate : int -startTime : long
+sprites() : LinkedList +applet() : Applet -_gamePaused = false : boolean +gamaPaused() : boolean +pauseGame() : void +resumeGame() : void *gameStartup()* : void *gameTimedUpadate()* : void *gameRefreshScreen()* : void *gameShutdown()* : void *gameKeyDown(keyCode :* int*)* : void *gameKeyUp(keyCode :* int*)* : void *gameMouseDown()* : void *gameMouseUp()* : void *gameMouseMove()* : void *spriteUpdate(sprite : AnimatedSprite)* *spriteDraw(sprite : AnimatedSprite)* *spriteDying(sprite : AnimatedSprite)* *spriteCollision(spr1 : AnimatedSprite, spr2 : AnimatedSprite)* <<constructor>> +Game(frameRate : int, width : int, height : int) +graphics() : Graphics2D +frameRate() : int +mouseButton(btn : int) :boolean +mousePosition() : Point2D +init() : void +update(g : Graphics) : void +paint() : void +start() : void

+run() : void +stop() : void +keyTyped(k : KeyEvent) : void +keyPressed(k : KeyEvent) : void +keyReleased(k : KeyEvent) : void -checkButtons(e : MouseEvents) : void +mousePressed(e : MouseEvent) : void +mouseRelease(e : MouseEvent) : void +mouseMoved(e : MouseEvent) : void +mouseDragged(e : MouseEvent) : void +mouseEntered(e : MouseEvent) : void +mouseExited(e : MouseEvent) : void +mouseClicked(e : MouseEvent) : void #calcAngleMoveX(angle : double) : double #calcAngleMoveY(angle : double) : double #updateSprites() : void #testCollisions() : void #drawSprites() : void #purgeSprites() : void

Gambar 3.37 Game *Class* pada *Class Diagram* ArkAId XIV

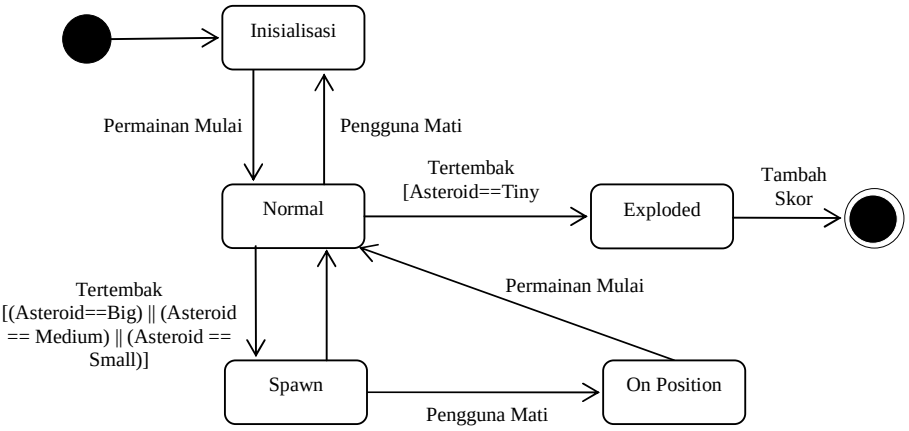
ArkAIdXIV
FRAMERATE : int SCREENWIDTH : int SCREENHEIGHT : int final u/ ASTEROIDS = 10 : int BULLET_SPEED = 4 : int ACCELERATION = 0.05 : double SHIPROTATION = 5.0 : double STATE_NORMAL = 0 : int STATE_COLLIDED = 1 : int STATE_EXPLODING = 2 : int SPRITE_SHIP = 1 : int SPRITE_ASTEROID_BIG = 10 : int SPRITE_ASTEROID_MEDIUM = 11 : int SPRITE_ASTEROID_SMALL = 12 : int SPRITE_ASTEROID_TINY = 13 : int SPRITE_BULLET = 100 : int SPRITE_EXPLOSION = 200 : int SPRITE_POWERUP_SHIELD = 300 : int SPRITE_POWERUP_HEALTH = 301 : int SPIRTE_POWERUP_250 = 302 : int SPRITE_POWERUP_500 = 303 : int

SPRITE_POWERUP_1000 = 304 : int SPRITE_POWERUP_GUN = 305 : int GAME_MENU = 0 : int GAME_RUNNING = 1 : int GAME_OVER = 2 : int showBounds = false : boolean collisionTesting = true : boolean background : ImageEntity bulletImage : ImageEntity[5] medAsteroids : ImageEntity[2] smlAsteroids : ImageEntity[3] tnyAsteroids : ImageEntity[4] explosions : ImageEntity[2] shipImage : ImageEntity[3] barImage : ImageEntity[2] barFrame : ImageEntity powerupShield : ImageEntity powerupHealth : ImageEntity powerup250 : ImageEntity powerup500 : ImageEntity powerup1000 : ImageEntity powerupGun : ImageEntity health = 20 : int shield = 20 : int score = 0 : int highscore = 0 : int firepower = 1 : int gameState = GAME_OVER : int rand : Random collisionTimer = 0 : long keyLeft : boolean keyRight : boolean keyUp : boolean keyFire : boolean keyB : boolean keyC : boolean keyShield : boolean music : MidiSequence shoot : SoundClip explosion : SoundClip
<<constructor>> +ArkAIdXIV() gameStarup() : void -resetGame() : void gameTimeUpdate() : void gameRefreshScreen() : void gameShutdown() : void

+spriteUpdate(sprite : AnimatedSprite) : void
+spriteDraw(sprite : AnimatedSprite) : void
+spriteDying(sprite : AnimatedSprite) : void
+spriteCollision(spr1 : AnimatedSprite, spr2 : AnimatedSprite) : void
+gameKeyDown(keyCode : int) : void
+gameKeyUp(keyCode : int) : void
+gameMouseDown() : void
+gameMouseUp() : void
+gameMouseMOve() : void
-breakAsteroid(sprite : AnimatedSprite) : void
-spawnAsteroid(sprite : AnimatedSprite) : void
-spawnPowerup(sprite : AnimatedSprite) : void
+createAsteroid() : void
+isAsteroid(spriteType : int) : boolean
+checkInput() : void
+applyThrust() : void
+ fireBullet() : void
-adjustDirection(sprite : AnimatedSprite, angle : double)
-stockBullet() : AnimatedSprite
+startBigExplosion(point : Point2D) : void
+startSmallExplosion(point : Point2D) : void
+warp(spr : AnimatedSprite) : void
+bumpScore(howmuch : int) : void+load(filename : String)
+transform() : void
+draw() : void
+getBound() : Rectangle

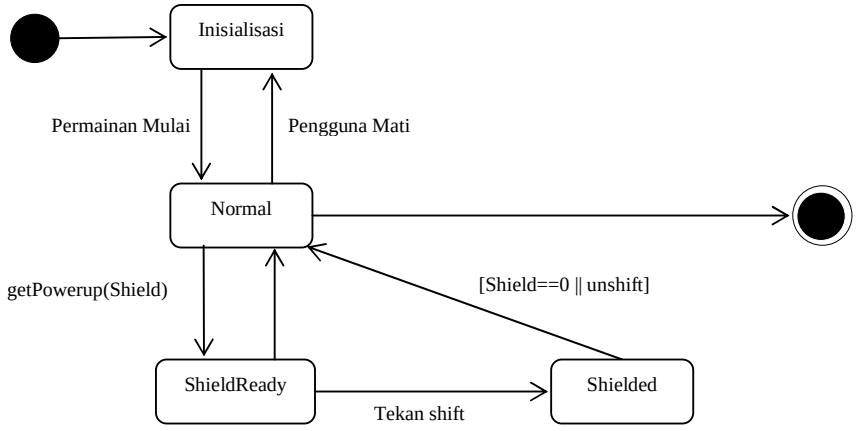
3.2.4.5 Diagram *State*

3.2.4.5.1 Diagram State Spawning Enemy



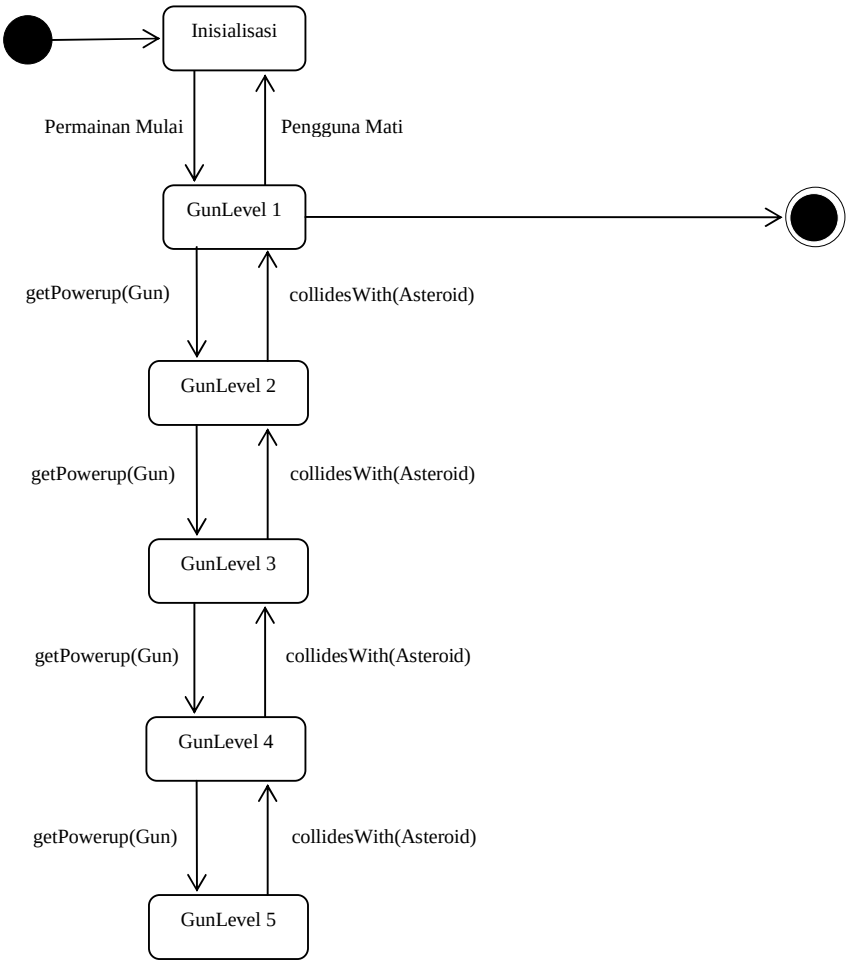
Gambar 3.38 *State Diagram* FSM *Spawning Enemy*

3.2.4.5.2 Diagram State Shielded



Gambar 3.39 *State Diagram* FSM *Ship Shielded*

3.2.4.5.3 Diagram State Bullet Upgrade



Gambar 3.40 *State Diagram* FSM *Bullet Upgrade*

3.2.5 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional dilakukan untuk mengetahui spesifikasi kebutuhan untuk sistem. Spesifikasi kebutuhan terdiri atas analisis perangkat keras dan perangkat lunak. Tahap analisis kebutuhan non-fungsional terdiri dari tiga sub tahap yaitu analisis pengguna, analisis perangkat keras, dan analisis perangkat lunak.

3.2.5.1 Analisis Pengguna

Berdasarkan hasil penelitian, spesifikasi minimal bagi pengguna yang dapat menggunakan aplikasi ini adalah sebagai berikut:

1. Peran : Pemain
2. Usia : Lebih dari 5 tahun
3. Tingkat Keterampilan : Dapat menggunakan komputer, aplikasi komputer dan perambah internet
4. Pengalaman : Pernah memainkan aplikasi permainan komputer

3.2.5.2 Analisis Perangkat Keras (Pengembang)

Perangkat keras yang dipergunakan dalam pembuatan aplikasi yang akan dibangun menggunakan spesifikasi sebagai berikut:

Tabel 3.23 Spesifikasi Perangkat Keras (Pengembang)

Nama Perangkat	Spesifikasi
Prosesor	Core 2 Duo
RAM	2 GB 800 MHz
Hard Disk	160 MB
Display	Resolusi 1366 x 768 18.5 LED Monitor
Keyboard	USB Standard
Mouse	USB Standard

3.2.5.3 Analisis Perangkat Lunak (Pengembang)

Perangkat lunak yang dipergunakan dalam pembuatan aplikasi yang akan dibangun menggunakan software dan tools sebagai berikut :

Tabel 3.24 Spesifikasi Perangkat Lunak (Pengembang)

Nama Perangkat Lunak	Jenis
Sistem Operasi	Microsoft Windows 7 Prof 64 Bit
IDE & Text Editor	Macromedia Dreamweaver 8,

	Notepad ++
Tools & SDK	JDK 6u27 x64, Ms. Command Prompt

3.2.5.4 Analisis Perangkat Keras (Pengguna)

Pada sisi pengguna, perangkat keras yang dibutuhkan untuk menjalankan aplikasi ini adalah sebagai berikut :

Tabel 3.25 Spesifikasi Perangkat Keras (Pengguna)

Nama Perangkat	**Spesifikasi**
Prosesor	Dual Core
RAM	1 GB
Hard Disk	40 GB
Display	1024 x 768 CRT Monitor
Keyboard	USB Standard
Mouse	USB Standard

3.2.5.5 Analisis Perangkat Lunak (Pengguna)

Pada sisi pengguna, perangkat lunak yang dibutuhkan untuk menjalankan aplikasi ini adalah sebagai berikut :

Tabel 3.26 Spesifikasi Perangkat Lunak (Pengguna)

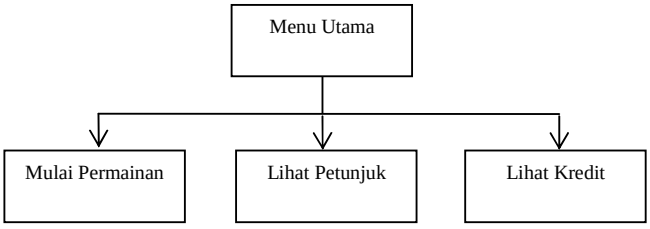
Nama Perangkat Lunak	**Jenis**
Sistem Operasi	Microsoft Windows Xp, Linux
Utilities	JRE 6
Perambah Internet	Mozzila Firefox/IE/Google Chrome

3.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak adalah kegiatan untuk merancang aplikasi yang akan kita bangun, tahapannya dimulai dari perancangan arsitektur menu, lalu berlanjut ke perancangan antarmuka, pesan dan jaringan semantik.

3.3.1 Perancangan Arsitektur Menu

Untuk mempermudah pembuatan sistem diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bagian mana yang terlebih dahulu nantinya yang akan diakses setelah program tersebut selesai. Gambar di bawah ini menunjukkan perancangan arsitektur menu dari sistem yang akan dibangun.



Gambar 3.41 Arsitektur Menu Aplikasi ArkAId XIV

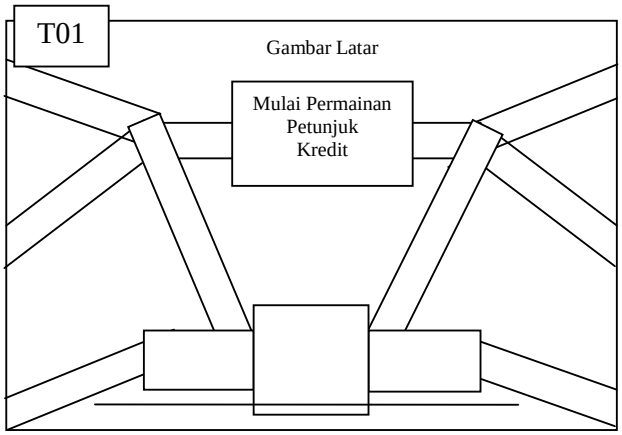
1. Mulai Permainan : submenu untuk memulai permainan arcade-shooter.
2. Lihat Petunjuk : submenu yang menampilkan informasi seputar petunjuk permainan seperti penekanan tombol tertentu pada *keyboard* beserta reaksinya.
3. Lihat Kredit : submenu yang menampilkan informasi mengenai pengembang aplikasi beserta informasi lain seputar aplikasi ArkAId XIV.

3.3.2 Perancangan Antar Muka

User interface ‘antarmuka pengguna’ adalah aspek sistem komputer atau program yang dapat dilihat, didengar atau dipersepsikan oleh pengguna dan

perintah-perintah atau mekanisme yang digunakan pengguna untuk mengendalikan operasi dalam suatu aplikasi.

3.3.2.1 Antar Muka Menu Utama

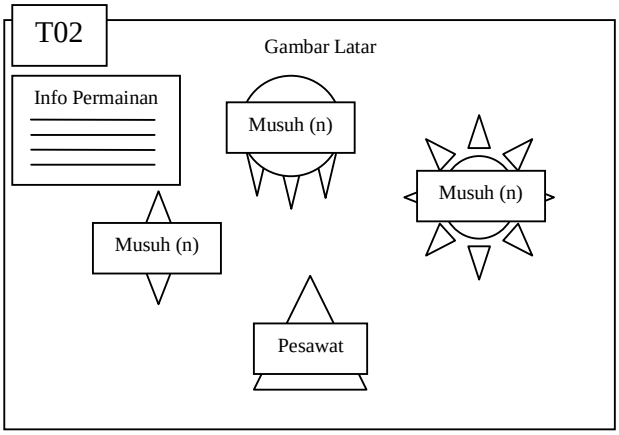


Ukuran : 800 x 600 (*Default*)
Latar : Gambar Latar
Tipe/Ukuran/Warna/Font : Custom, Courier

- Tekan Enter [Mulai Permainan] menuju T02
- Tekan P [Petunjuk] menuju T03
- Tekan K [Kredit] menuju ke T04
- Tekan ESC keluar M01

Gambar 3.42 Antar Muka Menu Utama

3.3.2.2 Antarmuka Mulai Permainan

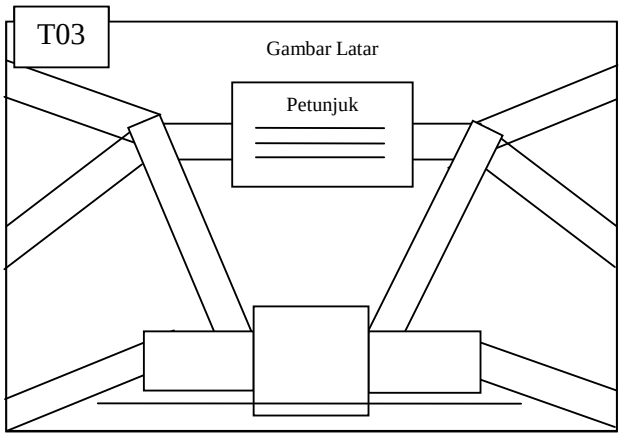


Ukuran : 800 x 600 (*Default*)
Latar : Gambar Latar
Tipe/Ukuran/Warna/Font : Custom, Courier

- Tekan ESC menuju Menu Jeda [Paused]
- Musuh (n) merupakan objek musuh (n) sejumlah n buah.
- Pesawat merupakan objek utama permainan yang dikendalikan pengguna.
- Info permainan merupakan info skor permainan.

Gambar 3.43 Antarmuka Mulai Permainan

3.3.2.3 Antarmuka Lihat Petunjuk

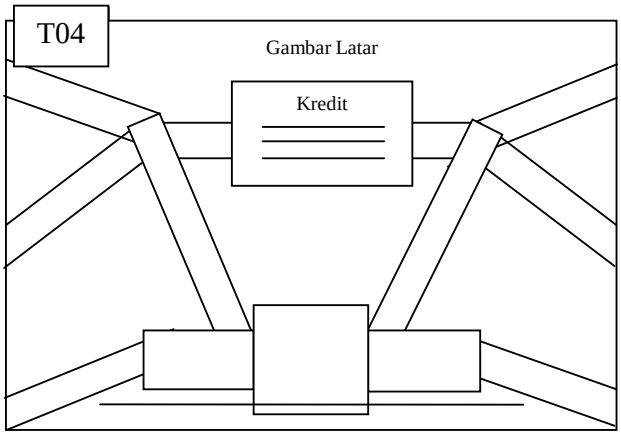


Ukuran : 800 x 600 (*Default*)
Latar : Gambar Latar
Tipe/Ukuran/Warna/Font : Custom, Courier

- Tekan Esc [Kembali] menuju T01

Gambar 3.44 Antarmuka Lihat Petunjuk

3.3.2.4 Antarmuka Lihat Kredit



Ukuran : 800 x 600 (*Default*)
Latar : Gambar Latar
Tipe/Ukuran/Warna/Font : Custom

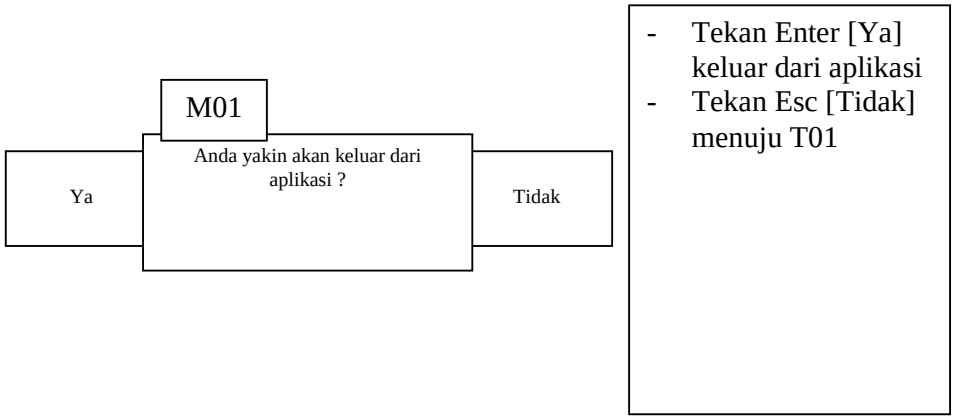
- Tekan Esc [Kembali] menuju T01

Gambar 3.45 Antarmuka Lihat Kredit

3.3.3 Perancangan Pesan

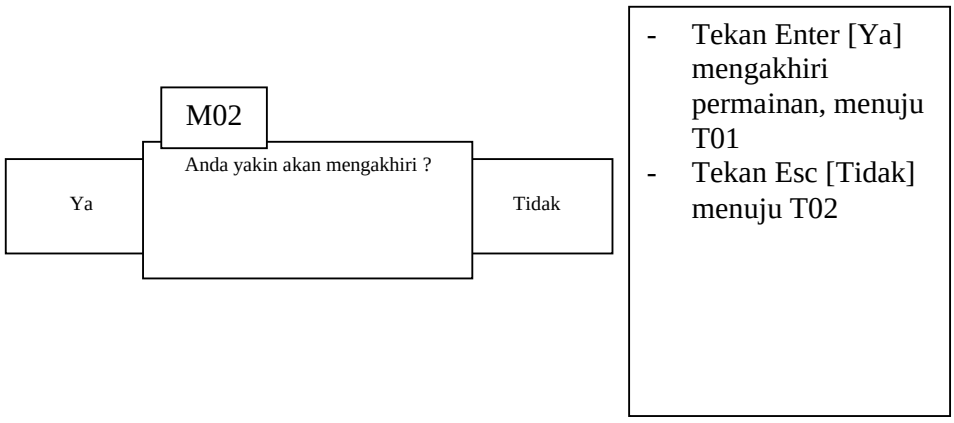
Pesan yang dirancang dalam konteks aplikasi yang dikembangkan merupakan pesan peringantan yang keluar jika pengguna melakukan aktifitas yang memicu *event* tertentu.

3.3.3.1 Pesan Peringatan Keluar Aplikasi



Gambar 3.46 Pesan Peringatan Keluar Aplikasi

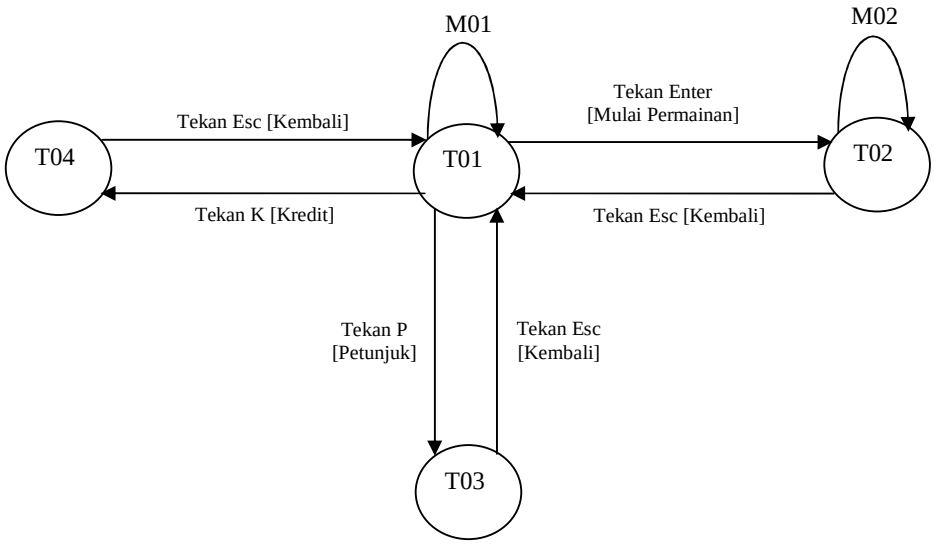
3.3.3.2 Pesan Peringatan Keluar Permainan



Gambar 3.47 Peringatan Keluar Permainan

3.3.4 Jaringan Semantik

Tahap selanjutnya adalah menggambarkan hubungan antar menu pada aplikasi ArkAId XIV. Penggambaran tersebut diwujudkan dalam sebuah jaringan semantik.



Gambar 3.48 Jaringan Semantik Aplikasi ArkAId XIV