BAB 3

ANALISIS DAN PERANCANGAN

3.1 Pengenalan Permainan

Game snake yang dikembangkan merupakan pengembangan dari *game snake* yang sangat tidak asing dan populer pada telepon seluler beberapa waktu yang lalu. Berdasarkan studi literatur *game snake* yang mengawali debutnya sebatas dengan beberapa kotak persegi yang bergerak di atas *background* berwarna hijau. *Snake* sendiri sudah eksis sejak akhir tahun 1970-an melalui *arcade*, permainan tersebut merupakan permainan yang mengawali masa keemasan permainan video yang dimulai pada dekade 70-an, Dan seiring perkembangan telah mengalami beberapa kali pembaruan. Konsep *gameplay*-nya pun juga diadopsi banyak *game* lain yang memang mengambil *setting* satu arena berbentuk kotak [14].

3.1.1 *Game Snake* Konvensional

3.1.1.1 Elemen Permainan

Dalam *game snake* kanvensional elemen permainannya terdiri dari:

1. Arena Permainan

   Pada *game snake* konvensional arena permainan sebatas dengan beberapa kotak persegi kecil yang bergerak di atas *background* berwarna hitam. Arena permainan ini merupakan tempat untuk bergerak untuk melaju mencari umpan yang sebanyak-banyaknya dalam waktu sesingkatnya.

2. Pemain (*Human Player*)

Pemain merupakan aktor utama yang menggunakan aplikasi *game snake* konvensional. Pemain dapat mengontrol pergerakan ular sehingga dapat bergerak dari satu titik ke titik lainnya dengan arah sumbuh x dan y pada arena permainan. Adapun tujuannya tersebut yaitu untuk mencari makanan yang sebanyak-banyaknya dengan waktu tercepat dan menghindari tabrakan pada tubuh ularnya sendiri yang mengakibatkan permainan *game over*.

3. Umpan

Umpan pada *game snake* konvensional terdiri dari dua macam, yaitu besar dan kecil yang nilainya masing-masing berbeda.

3.1.1.2 Skenario Permainan

Pada *game snake* konvensional, skenarionya disusun oleh beberapa hal seperti pada Tabel 3.1 berikut :

Tabel 3.1 A1ur Sistem *Game Snake* Konvensional

Alur Sistem
1. Sistem menginisialisasi seluruh entitas permainan-arena permainan, ular, dan umpan. 2. Umpan berada pada posisi tertentu yang ditentukan oleh sistem secara acak. 3. Posisi ular berada tempat yang tak menentu pada arena permainan. 4. Permainan dimulai. 5. Umpan muncul dengan posisi yang teracak dengan dua macam bentuk yang berbeda. 6. Pemain mengontrol ular dengan memasukkan perintah tertentu melalui

tombol *handphone* (panah atas, bawah, kiri dan kanan). 7. Sistem melakukan *update* terhadap posisi seluruh entitas permainan, logika sistem dan skor permainan.

2. Alur Permainan

Tabel 3.2 Alur Permainan *Game Snake* Konvensional

Alur Permainan
1. Sebuah ular berada dalam arena permainan persegi empat dimana akan terdapat umpan-umpan dengan muncul secara acak 2. Ular akan melakukan pemangsaan terhadap umpan-umpan dengan dua jenis yang berbeda secara maksimal. 3. Ular akan menjadi memanjang setelah melakukan pemangsaan umpan-umpan tersebut. 4. Jika ular menabrak badan sendiri maka akan terjadi *game over*.

3. Misi Permainan

Tabel 3.3 Misi *Game Snake* Konvensional

Misi
1. Memakan semua umpan yang ada baik yang kecil maupun yang besar. 2. Melakukan laju dengan hati-hati untuk menghindari menabrak badan ular sendiri. 3. Meraih skor tertinggi.

3.1.1.3 Entitas Permainan

Beberapa entitas permainan yang terlibat pada *game snake* konvensional dapat dilihat pada Tabel 3.4 berikut ini :

Tabel 3.4 Entitas *Game Snake* Konvensional

No	Nama	Gambar	Penjelasan
1	Ular		Ular merupakan entitas utama dalam permainan yang dikendalikan langsung oleh pemain. Ia dapat berganti posisi dari satu posisi ke posisi lain sesuai perintah dari pemain yang dimasukkan dari *keyboard.*
2	Umpan Besar		Umpan besar merupakan entitas lain dalam arena permainan. Memiliki tinggkat skor yang sangat tinggi. Kemunculan umpan ini sangat jarang dan pada saat kemunculan memiliki renggang waktu yang sangat sebentar, sehingga user harus mempunyai kegesitan untuk mendapatkan umpan besar ini.
3	Umpan Kecil		Umpan besar merupakan entitas lain dalam arena permainan. Merupakan santapan bagi ular yang dimakan dengan memiliki skor yang kecil.

3.1.2 Gas Atar

Gas Atar merupakan nama dari *game snake* yang dikembangkan. Adapun kata Gas berasal dari kata *game snake* yang merupakan genre dari permainan ini, sedangkan Atar merupakan AI A* (*star*) yang diterapkan pada *game* ini. Tapi, tidak hanya A* (star) saja yang akan diterapkan tapi juga ada beberapa AI lain diantaranya, *reaction algorithm* dan *collision detection* serta metodenya juga ada yang mengalami perubahan.

3.1.2.1 Elemen Permainan

Dalam Gas Atar elemen permainannya masih sama seperti pada permainan *game snake* konvensional, namun ada beberapa elemen tambahan. Adapun elemen-elemen tersebut terdiri dari:

1. Arena Permainan

   Arena permainan pada Gas Atar masih sama dalam segi bentuk dengan yang digunakan pada *game snake* konvensional namun, karena adanya pertandingan antara ular yang dijalankan *user* dengan ular yang diterapkan AI-nya. Sebagai pengembangannya arena permainan masih menggunakan berbasis *vektor* yang mana gambarnya ruang hitam kotak-kotak tetapi yang lebih besar. Arena permainan ini merupakan tempat untuk bergerak melaju dan muncul secara acak bagi entitas permainan lain seperti umpan. Pada arena ini akan ada batasan dinding yang mengakibatkan ular AI bereaksi apabila menabrak dinding arena.

2. Pemain (*Human Player*)

   Sama halnya dengan *game snake* konvensional pada Gas Atar, pemain merupakan aktor utama yang menggunakan aplikasi permainan. Pemain mengontrol laju ular sehingga dapat berbelok ke kiri, ke kanan, ke atas dan ke bawah pada arena permainan. Adapun tujuannya tersebut yaitu untuk mendapatkan makanan secepatnya dengan melihat jarak yang paling dekat sehingga pergerakan bisa jadi lebih optimal.

3. Musuh (*Non-Human Players*)

   Musuh pada Gas Atar terdiri dari beberapa jenis, tetapi pada permainan *game snake* konvensional tidak memiliki musuh hanya jika pada saat mencari umpan menabrak badan ular itu sendiri mengakibatkan *game over* serta adanya umpan yang muncul secara acak dan hanya beberapa detik saja. Pada *game snake* pengembangan ini akan terdapat beberapa musuh ular, yaitu ular yang diberikan AI A* (star) yang mengakibatkan terjadinya perlombaan dalam mencari umpan.

4. Umpan

   Umpan sendiri pada *game snake* konvensional memang sudah ada. Tapi di sini ada bebrapa pengembangan antara lain jenis umpan dan reaksi dari umpan. Untuk jenis akan terdapat beberapa macam jenis yang memiliki reaksi yang berbeda-beda. Reaksi umpan akan terjadi jika ular melakukan pemangsaan terhadap satu jenis umpan tersebut. Umpan-umpan tersebut

dibedakan dengan warna dan bentuk yang sangat jelas, sehingga bisa memilih-milih bagusnya yang mana akan dimangsa.

5. Batasan dinding

   Batasan dinding yang ada pada *game snake* akan ada pada area permainan ini. Batasan dinding ini nanti akan menjadi batasan maksimum kiri, kanan, atas, bawah ular pada area permaian dalam mencari makanan yang ada, jika ular AI menabrak maka akan dikendalikan dengan algoritma *collision detection*.

3.1.2.2 Skenario Permainan

Skenario pada Gas Atar tidak jauh berbeda dari permainan *game snake* konvensional, namun evolusi tentunya dilakukan pada beberapa hal. Adapun skenario hasil pengembangan tersebut disusun oleh beberapa hal seperti pada tabel 3.5 berikut:

1. Alur Sistem

Tabel 3.5 Alur Sistem Gas Atar

Alur Sistem
1. Sistem menginisialisasi seluruh entitas permainan-arena permainan, ular user, ular AI, umpan, dan batasan dinding. 2. Umpan-umpan berada pada posisi tertentu yang ditentukan oleh sistem secara acak. 3. Posisi ular user dan ular AI berada di tempat star yang sama pada arena permainan. 4. Permainan dimulai. 5. Ular AI bergerak disekitar arena permainan dan kemudian akan melakukan deteksi tabrakan jika melewati batasan dinding (batas layar).

6. Pemain mengontrol ular user dengan memasukkan perintah tertentu melalui mouse atau keyboard (ke kiri, ke kanan, ke atas dan ke bawah). 7. Sistem melakukan pembaharuan terhadap posisi seluruh entitas permainan, logika sistem dan skor permainan. 8. Jika umpan yang dimakan maka akan melakukan reaksi masing-masing sesuai dengan warna yang ada.

2. Alur Permainan

Tabel 3.6 Alur Permainan Gas Atar

Alur Permainan
1. Beberapa ular berada pada arena yang di sekitarnya terdapat beberapa umpan-umpan yang jadi mangsa ular-ular tersebut. 2. Ular user menghindari terjadinya tabrakan terhadap batasan dinding yang ada . 3. Jika ular AI menabrak batasan dinding maka ular AI akan melakukan algoritma deteksi tabrakan. 4. Sebelas jenis umpan akan muncul secara acak dalam arena permainan 5. Beberapa ular yang ada akan memangsa umpan-umpan yang ada dan menjadi skor ular tersebut masing-masing. 6. Umpan akan melakukan rekasi masing-masing apabila dimakan oleh ular-ular yang ada. 7. Ular yang skornya minimum akan diberi waktu dan jika waktunya habis maka ular tersebut akan *game over*.

3. Misi Permainan

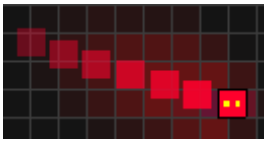
Tabel 3.7 Misi Permainan Gas Atar

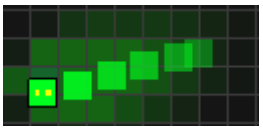
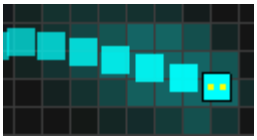
Misi
1. Memakan semua umpan-umpan yang ada dengan semaksimal mungkin dengan waktu yang ada. 2. Menghindari tabrakan terhadap batasan dinding yang ada 3. Melawan ular AI dengan melakukan pemangsaan secepatnya dan sebanyaknya. 4. Meraih skor tertinggi sehingga tidak ada batasan waktu.

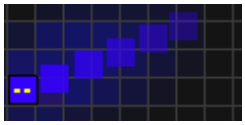
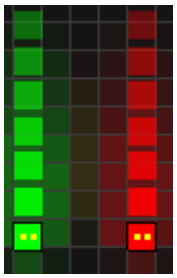
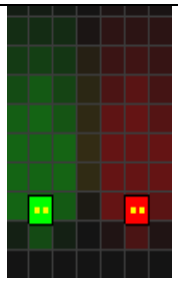
3.1.2.3 Entitas Permainan

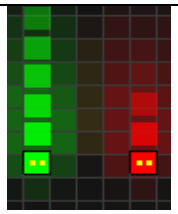
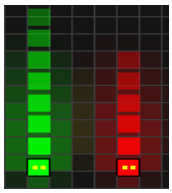
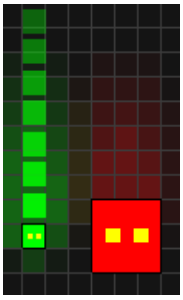
Pada dasarnya entitas-entitas permainan yang terlibat pada Gas Atar dengan yang ada pada permainan *game snake* konvensional, tetapi kini telah berbasis *animated sprite*. Selain itu, beberapa entitas baru pun ditambahkan dan penjelasannya ada pada Tabel 3.8 berikut:

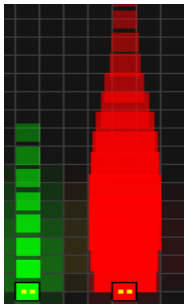
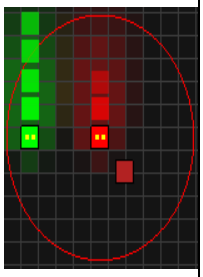
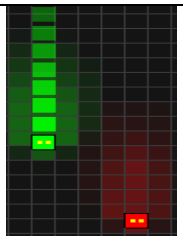
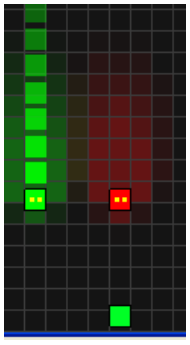
Tabel 3.8 Entitas Gas Atar

No	Nama	Gambar	Penjelasan	Gambar Kejadian
1	Ular Warna Merah		Ular yang ada nantinya akan dapat dikendalikan menjadi ular *user* dan juga bisa dijalankan AI nya sendiri, sehingga dapat menjadi musuh dan menjadi *user* itu sendiri nantinya, sesuai dengan keinginan *user* mau memilih warna yang mana.	-

2	Ular Warna Hijau		Ular yang ada nantinya akan dapat dikendalikan menjadi ular *user* dan juga bisa dijalankan AI nya sendiri, sehingga dapat menjadi musuh dan menjadi *user* itu sendiri nantinya, sesuai dengan keinginan *user* mau memilih warna yang mana.	-
3	Ular Warna Kuning		Ular yang ada nantinya akan dapat dikendalikan menjadi ular *user* dan juga bisa dijalankan AI nya sendiri, sehingga dapat menjadi musuh dan menjadi *user* itu sendiri nantinya, sesuai dengan keinginan *user* mau memilih warna yang mana.	-
4	Ular Warna Biru Muda		Ular yang ada nantinya akan dapat dikendalikan menjadi ular *user* dan juga bisa dijalankan AI nya sendiri, sehingga dapat menjadi musuh dan menjadi *user* itu sendiri nantinya, sesuai dengan keinginan *user* mau memilih warna yang mana.	-

5	Ular Warna Merah muda		Ular yang ada nantinya akan dapat dikendalikan menjadi ular *user* dan juga bisa dijalankan AI nya sendiri, sehingga dapat menjadi musuh dan menjadi *user* itu sendiri nantinya, sesuai dengan keinginan *user* mau memilih warna yang mana.	-
6	Ular Warna Biru		Ular yang ada nantinya akan dapat dikendalikan menjadi ular *user* dan juga bisa dijalankan AI nya sendiri, sehingga dapat menjadi musuh dan menjadi *user* itu sendiri nantinya, sesuai dengan keinginan *user* mau memilih warna yang mana.	-
7	Umpan Warna Biru dengan dikelilingi garis putih		Umpan ini akan menambah skor dengan memberi 6 kotak persegi tambahan pada belakang ular yang memangsa umpan warna ini.	
8	Umpan Warna Coklat		Jika dimakan umpan ini akan memberi reaksi negatif pada ular lawan yang skornya minimum selain yang memakannya dengan menghabiskan skor dan kotak	

			persegi (ekor) ular skor minimum tersebut.	
9	Umpan Merah		Umpan ini akan memberi tambahan skor pada ular dengan tambahan satu kotak persegi di belakang ular yang memakannya.	
10	Umpan Kotak Persegi putih		Umpan ini akan menambah skor dengan memberi 2 kotak persegi tambahan di belakang ular yang memangsanya.	
11	Umpan Hijau Tua		Jika dimakan umpan ini akan memberi reaksi negatif pada ular lawan yang skornya minimum selain yang memakannya dengan menghabiskan skor dan kotak persegi (ekor) ular skor minimum tersebut.	
12	Umpan Hijau Muda		Umpan ini akan menambah skor dengan memberi 5 kotak persegi tambahan dibelakang ular yang memangsanya.	
13	Umpan Warna Biru		Jika dimakan, umpan ini akan memberi reaksi pada ularnya, yaitu penambahan besar kepala ular sehingga ular dapat memangsa dengan area lebih luas.	

14	Umpan Warna Ungu		Umpan ini akan memberi reaksi saja dan tidak menambah skor. Dengan reaksi bentuk ular menjadi bergelombang dengan adanya sprite yang ada dan hanya memperindah bentuk ular yang ada.	
15	Umpan Warna Merah muda		Jika dimakan, umpan ini akan memberi reaksi dengan adanya garis melingkar disekeliling kepalanya. Dengan adanya lingkaran tersebut maka umpan yang berada dalam lingkaran tersebut otomatis akan termakan sendiri oleh ularnya dengan waktu yang telah ditentukan.	
16	Umpan Warna Kuning		Umpan ini akan memberi reaksi percepatan pada ular yang memakannya untuk mendapatkan umpan yang ada .	
17	Umpan Warna Awal Hijau danKemudian Berubah warna		Umpan ini diciriin dengan warna yang berubah dari warna awal. Dengan member reaksi, yaitu sebelum ular pemangsa akan memangsa dengan jarak 4 kotak persegi maka ular ini akan terlebih dahulu lari sampai batasan dinding.	

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, pemberian warna yang menarik, dan mudah dikendalikan. Semua itu berlaku pada semua jenis permainan tidak terkecuali *game snake* yang merupakan objek penelitian. Berdasarkan hasil penelitian, pada aplikasi permainan *game snake* 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

game snake konvensional AI yang diterapkan pada objek umpan adalah algoritma random. AI ini menjadikan umpan dapat muncul secara acak dalam waktu yang telah ditentukan. Perlawanan nya pun belum ada sama sekali, baik perlawan terhadap ular serta perlawan yang ada pada objek umpan. Pada permainan Gas Atar nanti akan ada ular yang lain dengan penerapan AI A* (star) untuk dapat bisa bergerak sendiri dalam mencari umpan dangan jarak dan waktu yang sesingkat-singkatnya. Serta umpan yang diberi *reaction algorithm* yang mengakibatkan pereaksi umpan.

Penurunan *gaming experience* terjadi karena *gameplay* alur permainan pada *game snake* konvensional terlalu monoton. Hal itu akan terasa oleh pengguna yang telah memainkannya dengan durasi lebih dari delapan jam. Pergerakan dari ular tidak leluasa dikarenakan adanya keterbatasan arena dan perpanjangan ular tidak maksimal. Oleh karenanya, seiring dengan bertambahnya durasi bermain *gaming experience* akan terus menutun karena pengguna akan merasa tidak tertantang lagi dan bosan. Itulah alasan kenapa *game snake* konvensional menjadi tidak menarik.

Penyajian tampilan antarmuka pada *game snake* konvensional tidak banyak berubah sejak masa awal kejayaannya. Hal itu mungkin tidak menjadi masalah bagipara 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 tinggalkan.

Untuk membuat *game snake* 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 memperbaharui *loak 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 mempengaruhi 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 *game snake* 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 *game snake* konvensional sehingga dikatakan tidak menarik diantaranya:

1. Alur permainan yang monoton dan terlalu sederhana.
2. Metodenya kurang, terutama belum ada perlawan.
3. Kontrol navigasi ular yang belum maksimal dalam pencarian umpan.
4. Kurang banyak dan variatifnya entitas permainan terutama entitas umpan.

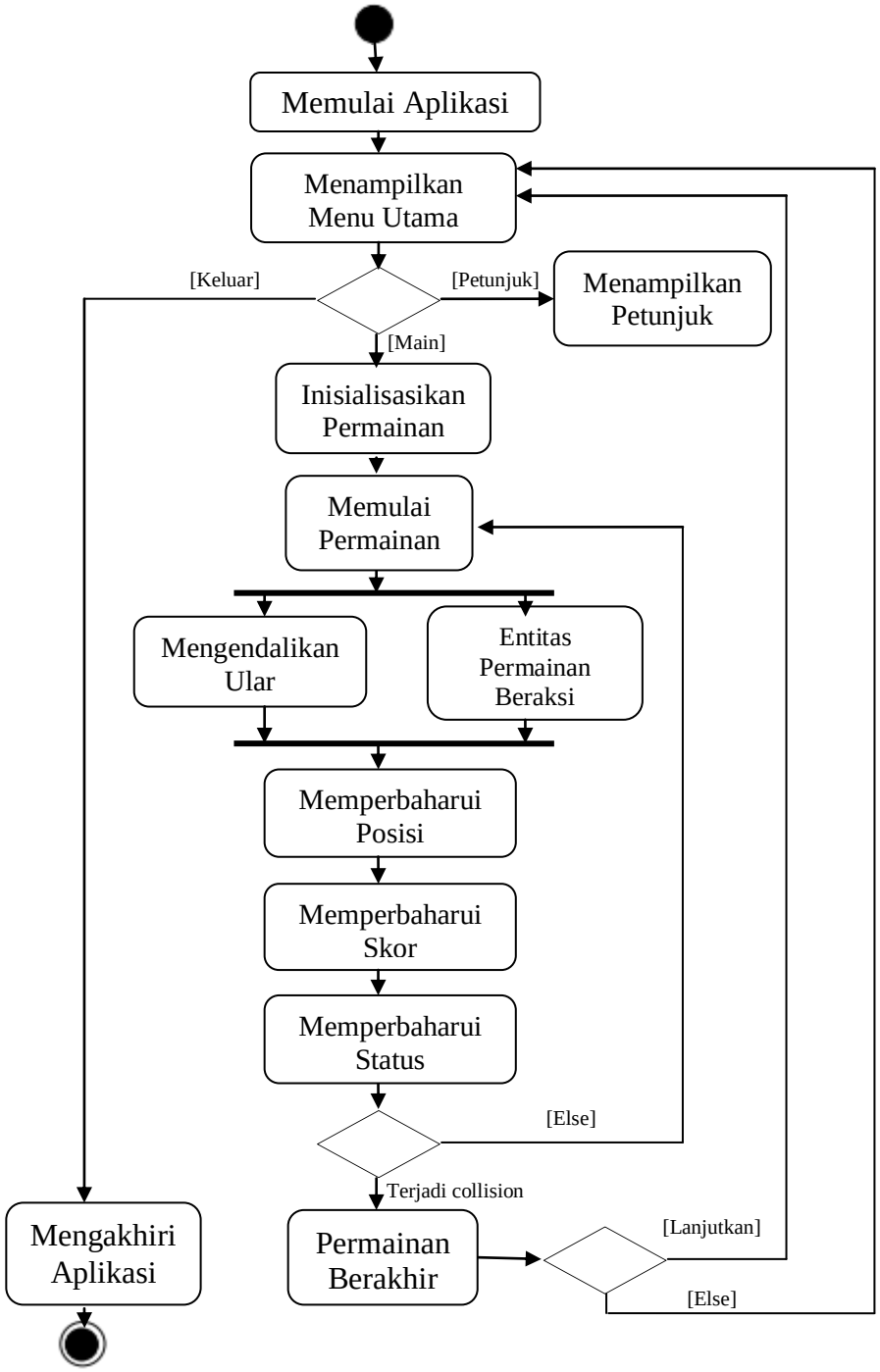
Hal tersebutlah yang menyebabkan *game snake* konvensional jadi kurang menarik.

3.2.3 Analisis Metode

3.2.3.1 Metode pada *Game Snake* Konvensional

Pada *game snake* konvensional AI yang diterapkan pada objek umpan adalah adalah algoritma random. Algoritma random menjadikan objek musuh bergerak secara acak. Adapun basis pengetahuannya berisi penerapan dari algoritma random 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 *game snake* konvensional dapat dilihat pada Gambar 3.1 berikut:



Gambar 3.1 *Activity Diagram* Skenario *Game Snake* Konvensional

Berikut ini penjelasan mengenai beberapa skenario pada aplikasi *game snake* beserta beberapa alternatifnya:

Skenario Utama

Nama Metode : Mulai Permainan Baru

Penjelasan : Pengguna menjalankan aplikasai *game snake* memulai permainan baru dan memainkannya sampai permainan berakhir.

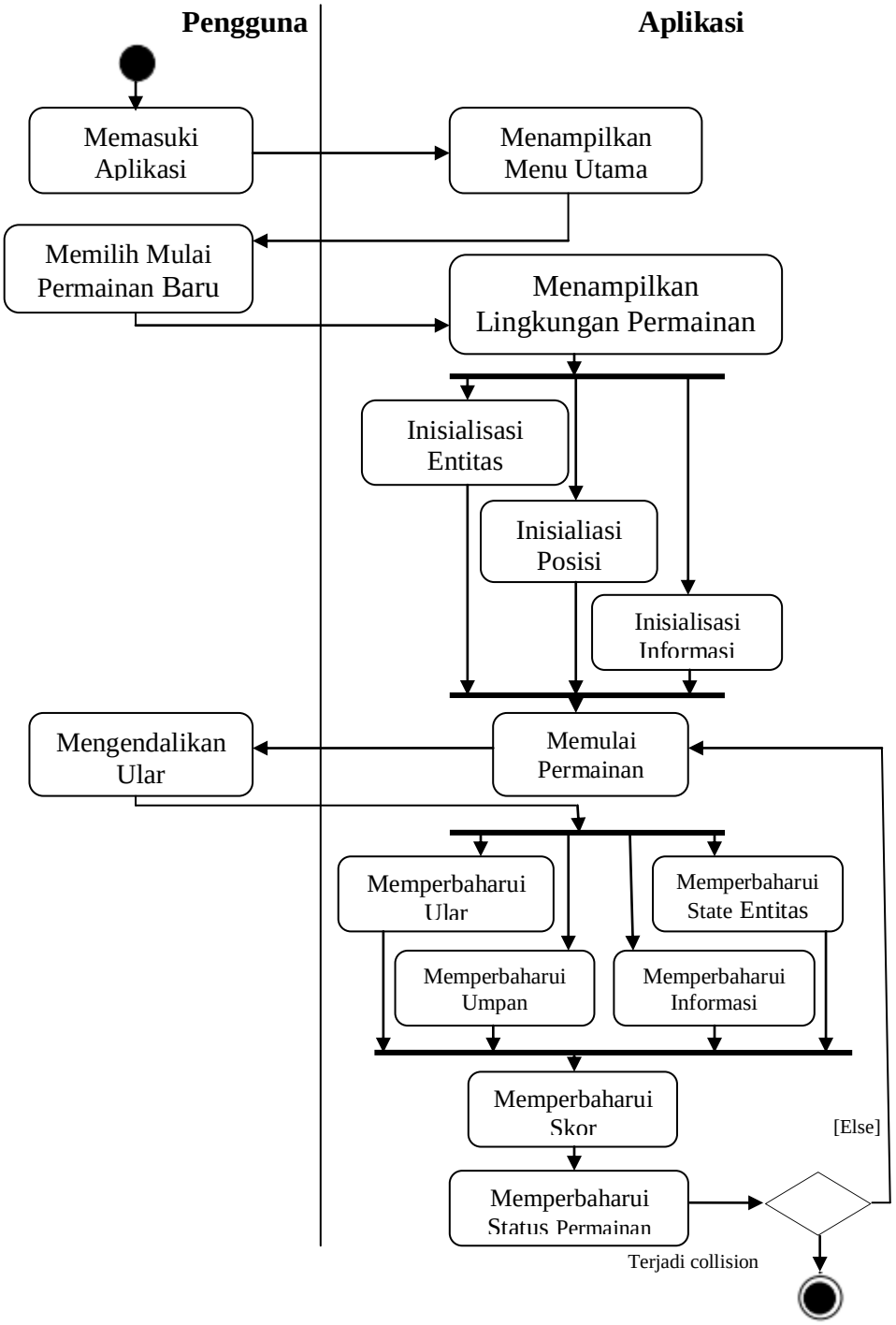
Tujuan Utama : Mencari umpan sebanyak-banyaknya dengan waktu minimal.

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

Kondisi Akhir : Ular menabrak badan sendiri

Tabel 3.9 Skenario Utama *Game Snake* Konvensional

Pengguna	Aplikasi
1.Skenario ini dimulai ketika pengguna menjalankan aplikasi *game snake*.	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 ular.
	6. Menentukan posisi awal umpan.
	7. Menampilkan informasi permainan.
	8. Memulai Permainan.
9. Mengendalikan ular.	10. Memperbaharui entitas permainan.
	11. Memperbaharui posisi umpan.
	12. Memperbaharui status entitas.
	13. Memperbaharui Informasi
	14. Memperbaharui skor.
	15. Memperbaharui status permainan
	16. Skenario ini berakhir ketika ular menabrak badan ular sendiri.



Gambar 3.2 *Activity Diagram* Metode Mulai Permainan Baru

Alternatif Skenario #1

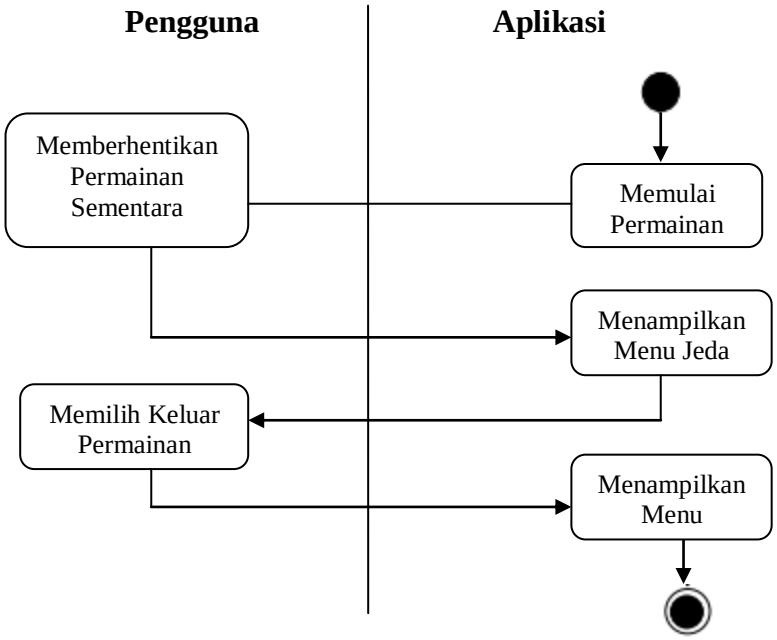
Penjelasan : Pengguna memilih meninggalkan permainan sebelum level berakhir.

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

Kondisi Akhir : Kembali ke menu utama.

Tabel 3.10 Alternatif Skenario #1

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 3.3 *Activity Diagram* Alternatif Skenario #1

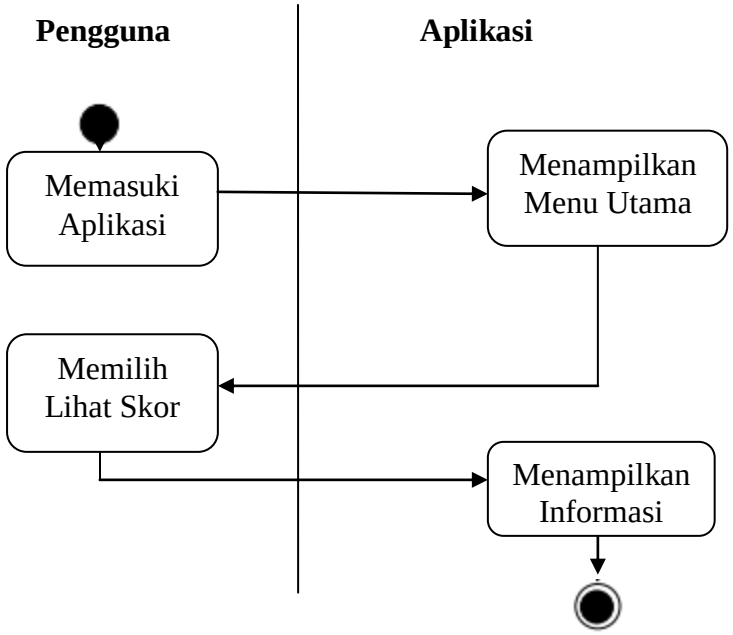
Alternatif Skenario #2

Penjelasan : Pengguna melihat informasi skor pemain.

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

Tabel 3.11 Alternatif Skenario #2

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



Gambar 3.4 *Activity Diagram* Alternatif Skenario #2

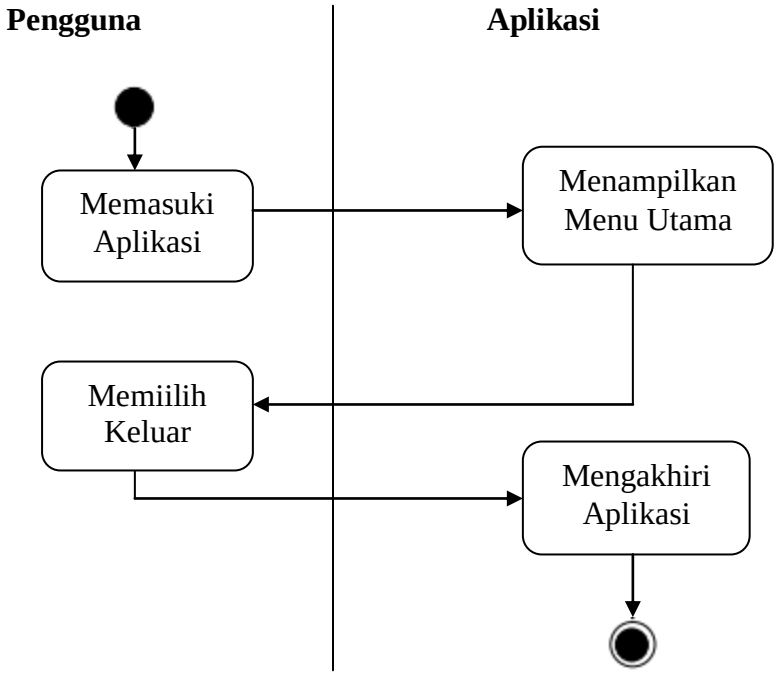
Alternatif Skenario #3

Penjelasan : Pengguna keluar dari aplikasi.

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

Tabel 3.12 Altetnatif Skenario #3

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



Gambar 3.5 *Activity Diagram* Alternatif Skenario #3

Beberapa metode yang digunakan aplikasi *game snake* konvensional pada *gameplay* alur permainannya dapat dilihat pada Tabel 3.13 berikut:

Metode Pengendalian Ular

Penjelasan : Pengguna mengendalikan ular dengan memasukkan perintah melalui papan ketik kemudian objek ular merespon *input* perintah tersebut dengan suatu aksi.

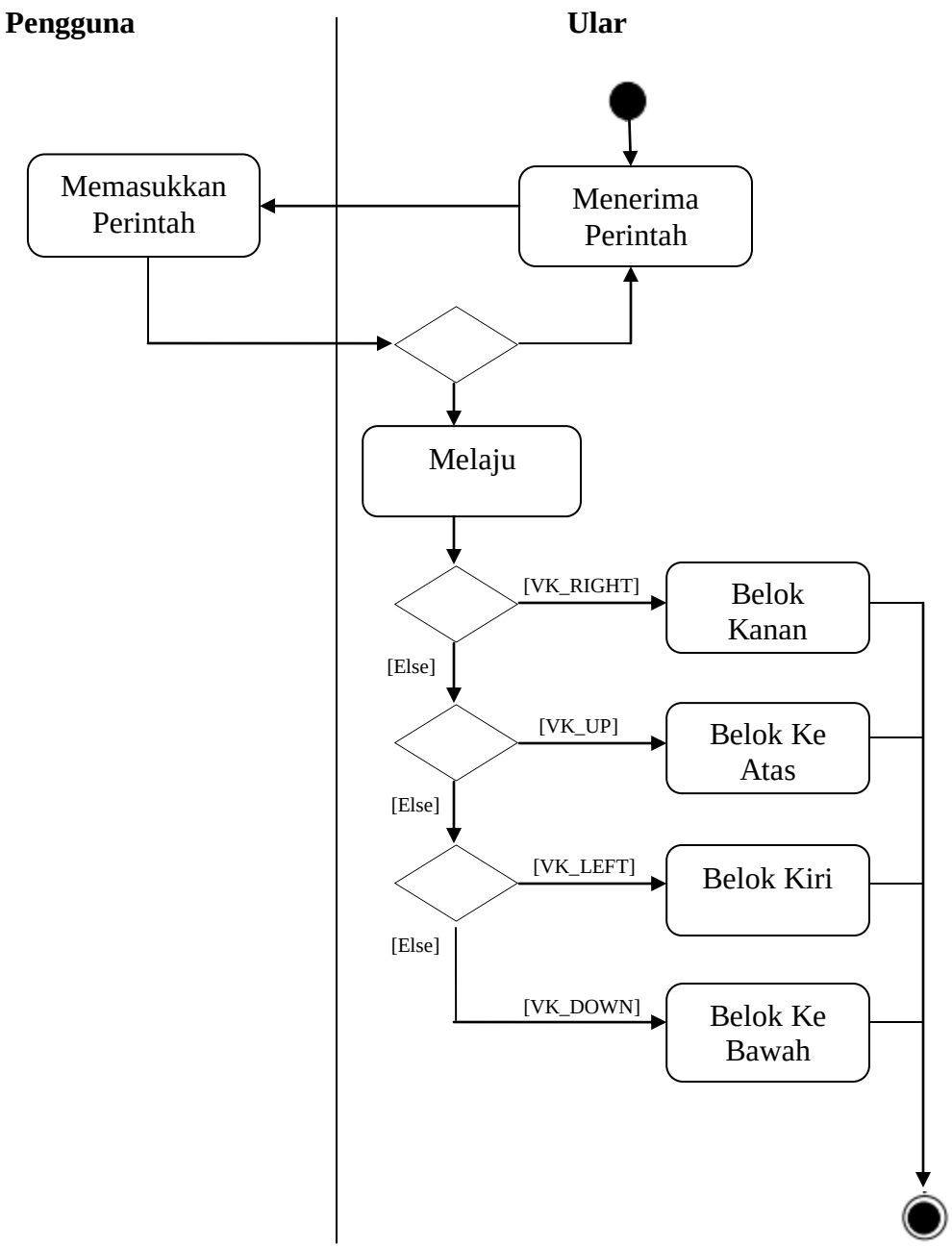
Tujuan : Objek ular melakukan aksi sesuai *input* dari pengguna.

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

Kondisi Akhir: Objek ular melakukan aksi.

Tabel 3.13 Metode Pengendalian Ular

Pengguna	Ular
1. Bersiap memasukkan perintah	2. Bersiap menerima aksi dari pengguna.
3. Menekan tombol atas (VK_UP)	4. Arah ular ke atas
5. Menekan tombol bawah (VK_DOWN)	6. Arah ular ke bawah
7. Menekan tombol kanan(VK_RIGHT)	8. Arah ular ke kanan.
9. Menekan tombol kiri (VK_LEFT)	10. Arah ular ke kiri.



Gambar 3.6 *Activity Diagram* Metode Pengendalian Objek Ular

Metode Pola Kemunculan Umpan (*Pattern Bused Random Argorithm*)

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

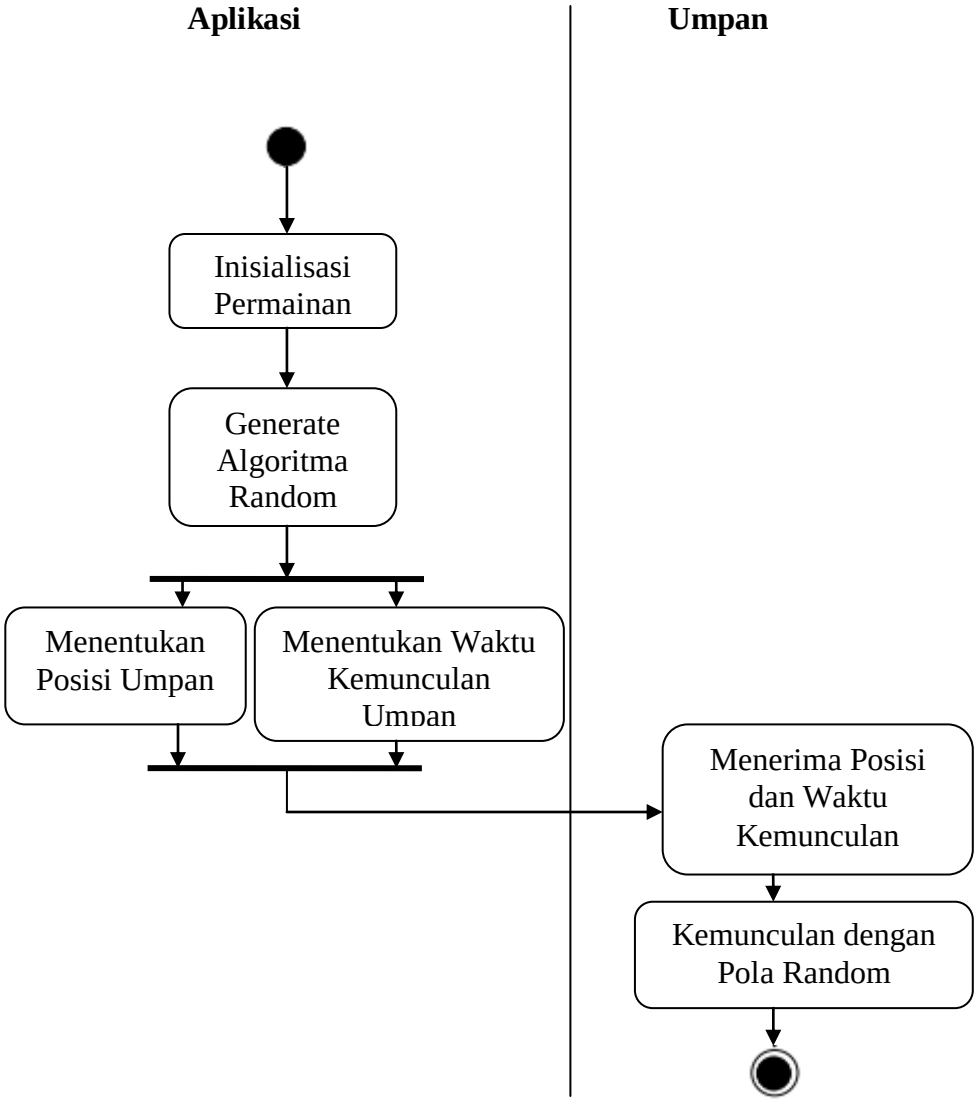
Tujuan : Menanamkan algoritma random pada objek umpan sehingga dapat muncul dengan pola acak

Kondisi Awal : Permainan berada pada tahap inisialisasi.

Kondisi Akhir : Umpan muncul dengan pola acak.

Tabel 3.14 Metode Pergerakan Objek Umpan

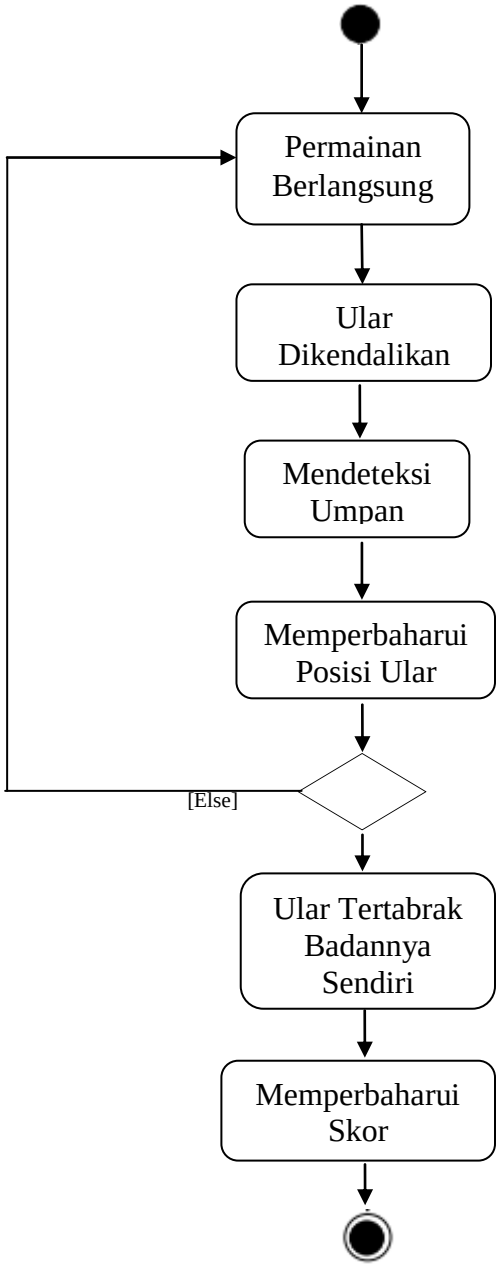
Aplikasi	Objek Umpan
1. Memulai tahap inisialisasi permainan.	
2. Menerapkan algoritma random dalam basis pengetahuan umpan dengan menggunakan pustaka random pada bahasa pemrograman dan mengacak nilai untuk : 2.1 Menentukan posisi koordinat objek musuh pada lingkungan permainan. 2.2 Menentukan pola kemunculan objek umpan.	
	3. Menerima posisi dan pola kemunculan.
	4. Bergerak dengan pola acak.



Gambar 3.7 *Activity Diagram* Metode Pergerakan Objek Umpan

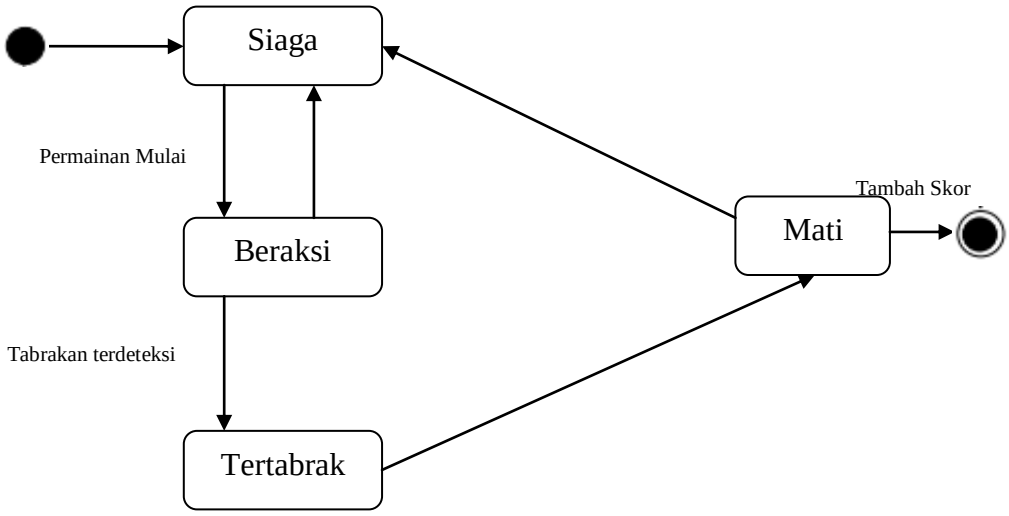
Gambar 3.7 di atas merupakan *activity diagram* pada saat pemunculan umpan, tapi pada Gas Atar nanti akan terus ada perubahan saat awal kemunculan umpan.

Sedangkan Gambar 3.8 berikut ini merupakan *activity diagram* pada saat pemain memakan umpan.



Gambar 3.8 *Activity Diagram* Metode Memakan Umpan

Adapun *state diagram* yang berlaku pada sebagian entitas dalam permainan *game snake* konvensional dapat dilihat pada Gambar 3.9 di bawah ini:



Gambar 3.9 *State Diagram* Entitas *Game Snake* Konvensional

Berdasarkan analisis metode pada *game snake* konvensional didapatkan beberapa metode yang mabih memiliki kelemahan. Hal tersebut dijelaskan pada Tabel 3.15 dibawah ini:

Tabel 3.15 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. Keadaan ular juga setelah star yang dengan sendiri

	melaju tanpa adanya reaksi dari user terlebih dahulu.
Alternatif Skenario #1 (Ular tertabrak badan sendiri)	Setelah tabrakan terjadi dengan menabrak badan sendiri makan *game snake* konvensional akan terjadinya *game over*, dan skor yang ada akan tersimpan, sehingga tantangan yang ada kurang maksimal.
Pengendalian Objek Ular	Event Handler hanya dapat menerima satu perintah dalam satu waktu. Akibatnya, pergerakan ular terasa kaku dan kasar karena *event handler* tidak mendukung fasilitas penekanan tombol secara bersamaan (*combinekey*), sehingga pergerakan ular jadi tidak maksimal.
Arena Permainan	Arena pada *game snake* konvensional terlalu sempit dan adanya proses *spaw*, sedangkan pada *game snake* pengembangan arena akan lebih luas dan tidak bisa menembus dinding pembatas yang ada, sehingga membuat tertantang.
Pergerakan Objek Umpan	Masih menggunakan pergelakan terpola (*pattern based*) dan algoritma random, sehingga posisi umpan 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 *game snake* 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 *game snake*. Oleh karena itu *game* AI pada permainan tersebut harus diperbaharui untuk mengatasi kelemahan dari *game snake* konvensional.

Pengembangan metode yang dilakukan akan menggunakan teknik *reasoning*, dimana teknik *reasoning* yang akan digunakan adalah A* (*star*). A* (*star*) akan dijalankan pada objek ular lawan, sehingga ada objek musuh cukup cerdas dan mempunyai semangat perlawan yang lebih tinggi. Dimana tadinya tidak adanya objek lawan/musuh yang menyebab terjadinya perlawan ular. Dengan bertambah gesitnya perlawanan dari musuh pengguna pun akan merasa lebih tertantang karenanya.

Selain Algoritma A* (*star*) pada *game snake* pengembangan (Gas Atar) nantinya akan diterapkan AI lainnya antara lain *reaction algorithm* dan *collision detection*. Pertama mengenai algoritma *reaction algorithm* merupakan algoritma reaksi yang akan diterapkan pada beberapa umpan yang akan dimakan oleh ular. Dimana nantinya akan ada reaksi apabila ular melakukan pemakaman terhadap umpan tersebut. Umpan-umpan tersebut ada yang memberi efek positif dan juga efek negatif, baik bagi ular yang memakannya maupun ular lawan.

Mengenai algoritma *collision detection* pada *game snake* pengembangan (Gas Atar) nantinya akan diterapkan pada ular lawan/musuh (ular yang tidak dikendalikan

usr). Dimana nanti *collision detection* akan membantu ular lawan agar tidak menabrak dinding pembatas pada arena permainan. Saat melakukan pencarian umpan ular akan melakukan pengejaran umpan sampain dinding pembatas arena permainan sampai mendapatkan umpan-umpan tersebut. Jika sampai dinding pembatas maka ular lawan akan menjalankan algoritma *collision detection* dan larinya kalau tidak ke kiri, ke kanan, ke atas, atau juga kebawah sesuai pencarian selanjutnya umpan yang terdekat sehingga tidak langsung melaju terus ke depan sampai menabrak dinding pembatas.

Berikut ini beberapa gambaran ular lawan pada saat mencari makanan/umpan yang akan dimangsa :

a. Algoritma A * (*star*)

Ular lawan ini akan memiliki kecerdasan yang mampu mendeteksi/mencari makanan dengan metode perhitungan jarak dan perkiraan proses terdekat yang diperoleh untuk mendapatkan makanan dengan sebanyak-banyaknya. Sehingga ular ini menjadi cerdas atau bisa bergerak sendiri dalam mencari makanannya dalam proses melawan ular yang dikendalikan oleh *user*.

Dimana algoritma A* (star) diuraikan pada Tabel 3.16 di bawah ini :

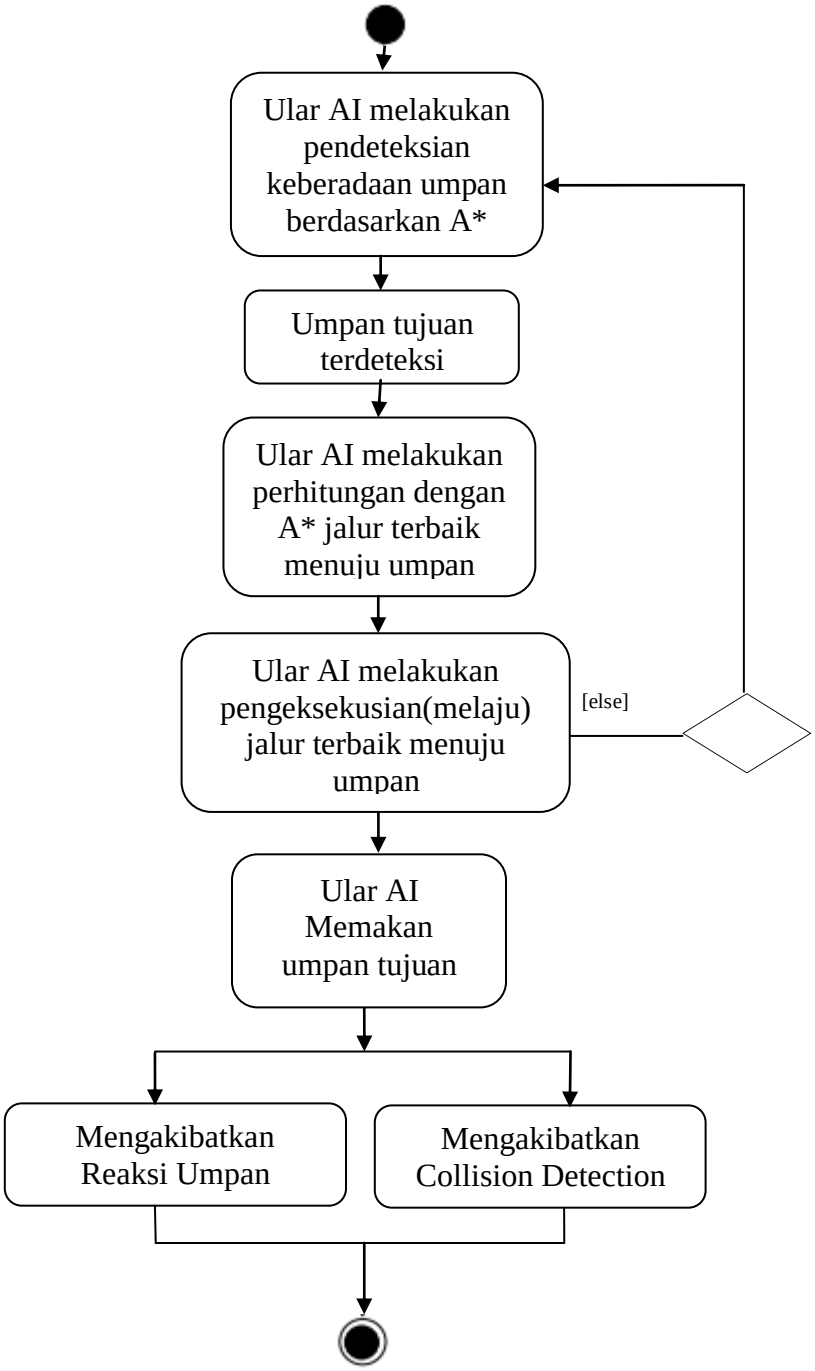
Tabel 3.l6 Pseude Code Algoritma A* (Star)

Nama Entitias	Algoritma

Ular	{I.S: Even A* terpicu}
	{T.S:Mencari umpan terdekat sesuai perhitungan A*}
	{F.S:Memakan Umpan hasil pencarian A*}
	
	function a*(start,goal)
	closedset = the empty set // The set of nodes already evaluated.
	openset = {start} // The set of tentative nodes to be evaluated, initially containing the start node
	came_from = the empty map // The map of navigated nodes.
	
	g_score[start] = 0 // Cost from start along best known path.
	// Estimated total cost from start to goal through y.
	f_score[start] = g_score[start] + heuristic_cost_estimate(start, goal)
	
	while openset is not empty
	current = the node in openset having the lowest f_score[] value
	if current = goal
	return reconstruct_path(came_from, goal)
	
	remove current from openset
	add current to closedset
	for each neighbor in neighbor_nodes(current)
	if neighbor in closedset
	continue
	tentative_g_score = g_score[current] + dist_between(current,neighbor)

	if neighbor not in openset or tentative_g_score < g_score[neighbor] add neighbor to openset came_from[neighbor] = current g_score[neighbor] = tentative_g_score f_score[neighbor] = g_score[neighbor] + heuristic_cost_estimate(neighbor, goal) {Jika umpan sudah tidak ada} return failure function reconstruct_path(came_from, current_node) if came_from[current_node] is set p = reconstruct_path(came_from, came_from[current_node]) return (p + current_node) else return current_node

Adapun *acivity diagram* dari algoritma yang dijelaskan pada Tabel 3.16 di atas dapat dilihat pada Gambar 3.10 di bawah ini:

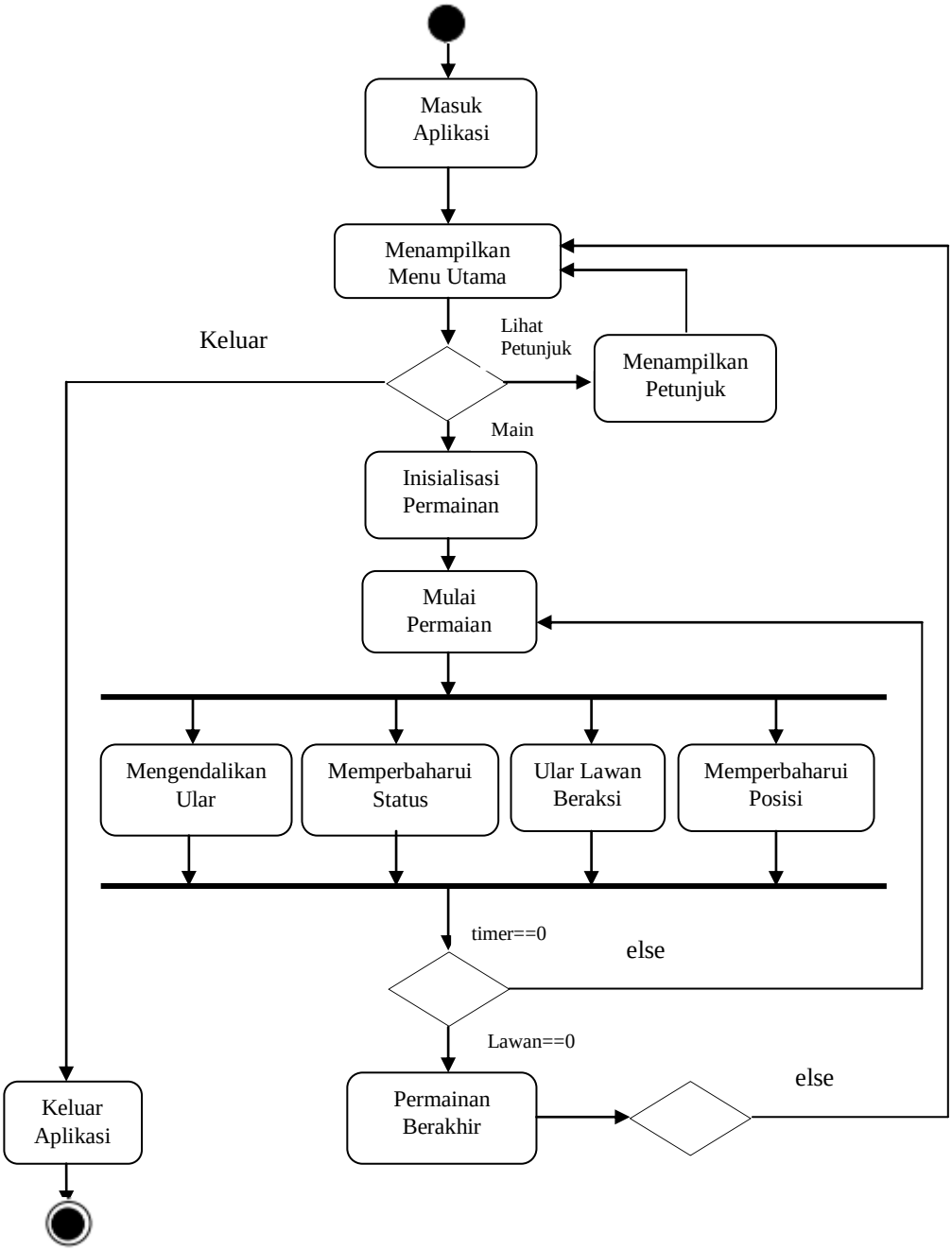


Gambar 3.10 *Activity Diagram* Beberapa Algoritma yang Terjadi pada Entitas

Scripting sebenarnya adalah sinonim dari teknik *reasoning* pada bidang *academical* AI. Pada aplikasi *game snake* 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 nrusuh. Penggunaan metode ini dapat menghemat waktu terutama pada tahap pengkodean.

Dengan diperbaharuinya AI pada *game snake* menggunakan genre AI yang telah dibahas, diharapkan dapat meningkatkan *game experience* bagi penggunanya sehingga permainan 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 penyebab 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 *game snake* konvensional harus dimodifikasi untuk memperbaiki kelemahan-kelemahannya. Gambar 3.11 berikut ini merupakan *activity diagram* pada metode yang akan dikembangkan:



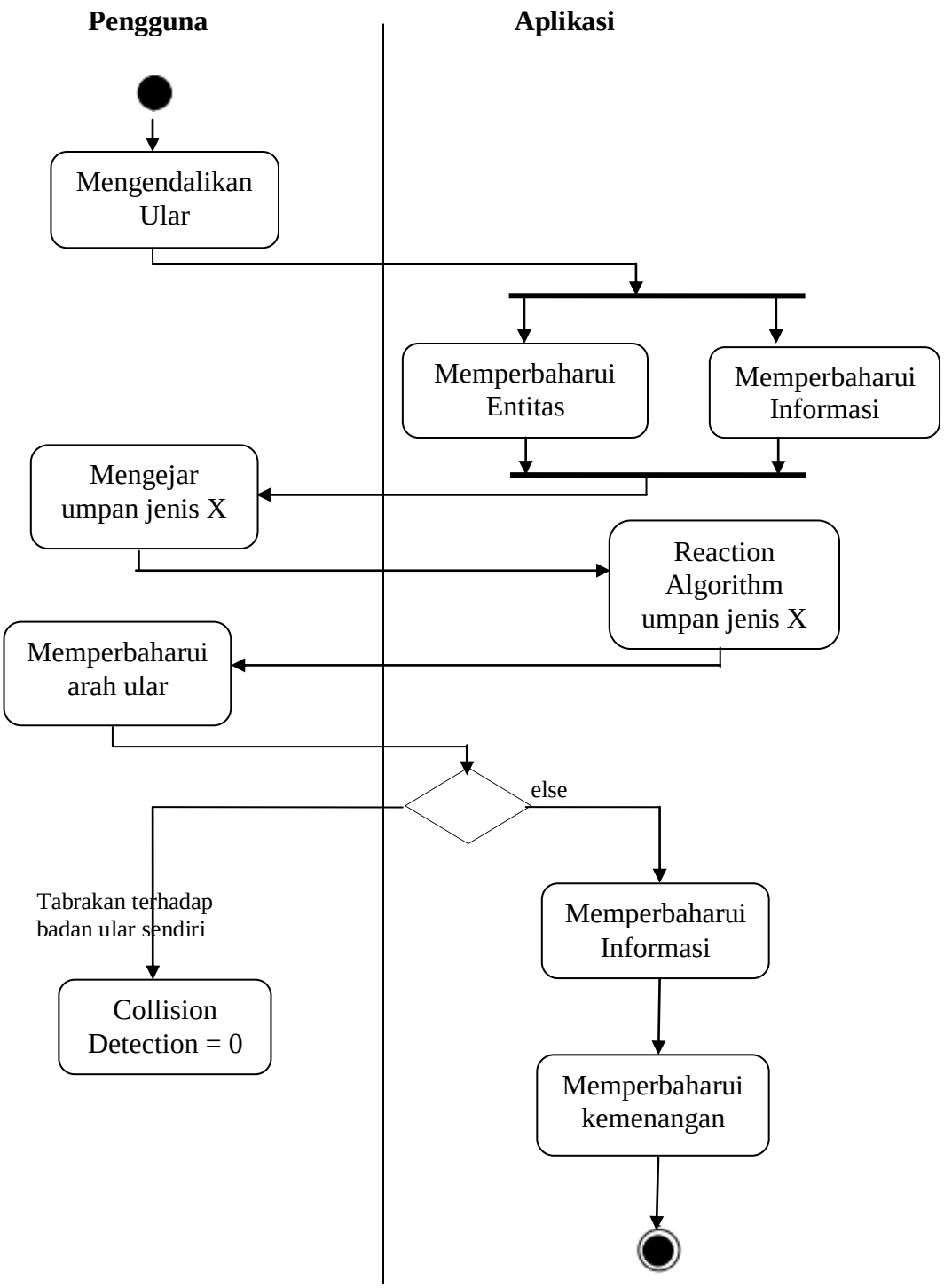
Gambar 3.11 *Activity Diagram* skenario Gas Atar

Pada Gambar 3.11 di atas 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, *multi threading* dapat diterapkan.

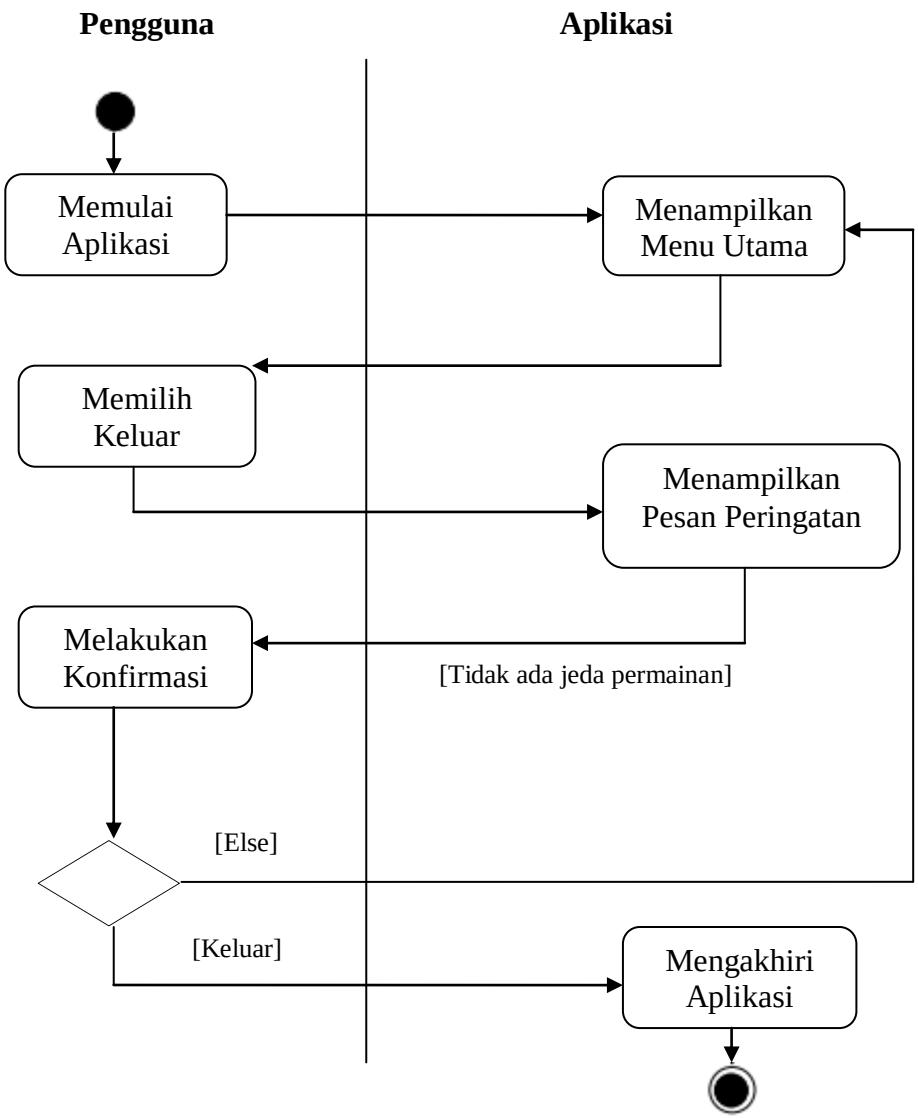
Pengembangan metode pada alternatif skenario #1 dilakukan dengan menambahkan proses aktivasi *reaction algorithm*. Hal itu terjadi ketika *user* mau melakukan pemakaman terhadap umpan yang jenis x, maka aktifnya *reaction algorithm* umpan jenis x. Kemudian apabila *user* melakukan perbaharuan arah kepala ular sampai menabrak badan sendiri makan tidak ada pengaruh apapun, dimana pada permainan yang lama akan terjadi kematian (permainan berakhir).

Pengembangan metode pada alternatif skenario #2 sebenarnya cukup mirip. Pada alur prosesnya ditambahkan proses konfirmasi yang akan memunculkan pesan peringatan kepada pengguna. Dengan adanya pesan ini diharapkan peluang terjadinya kejadian aplikasi keluar atau permainan berakhir secara tidak sengaja dapat ditekan. Saat *user* melakukan pemberhentian saat bermain, maka tidak ada jeda permainan.

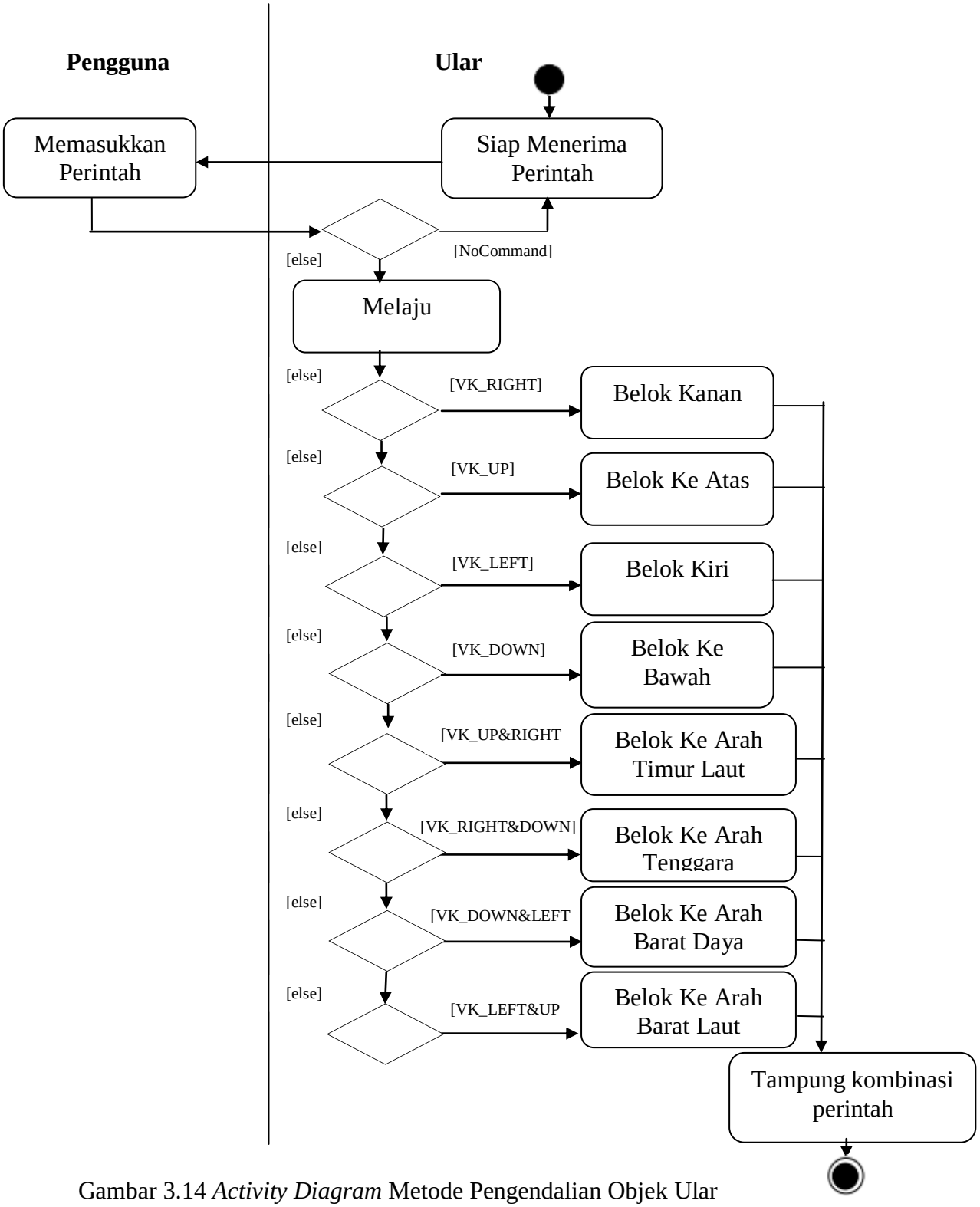
Pengembangan pada metode kontrol permainan adalah dengan menambahkan template untuk menampung sekumpulan perintah yang dimasukkan oleh pengguna. Dengan maksud itu, kini aplikasi dapat mendukung kombinasi dari penekanan perintah *input*-an pengguna.



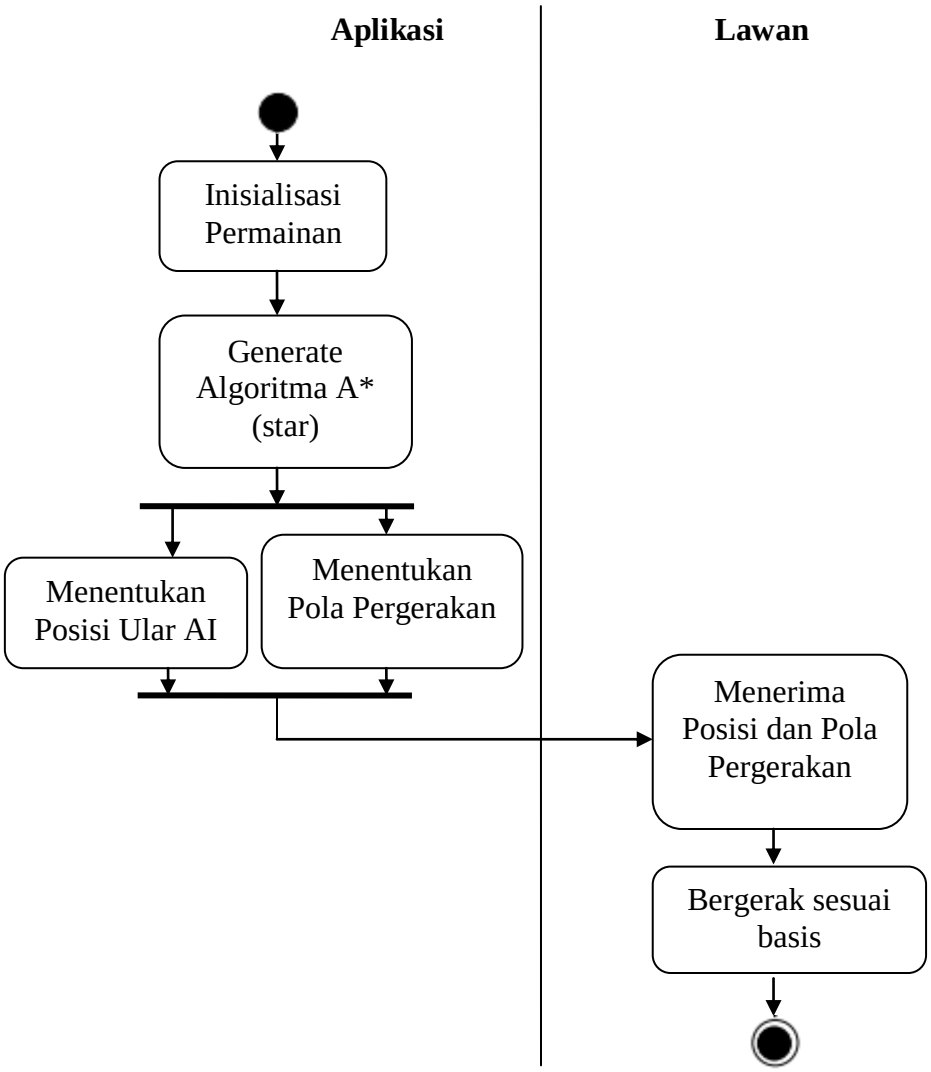
Gambar 3.12 Activity Diagram Alternatif Skenario #1 Gas Atar



Garnbar 3.13 *Activity Diagram* Altematif Skenario #2 Gas Atar

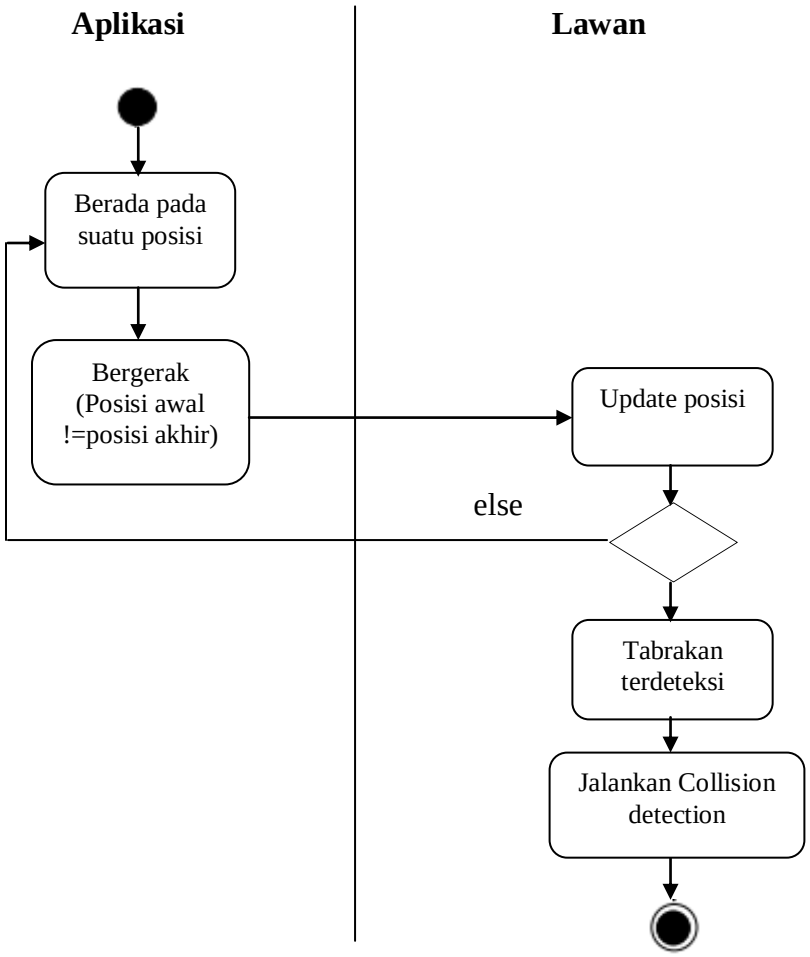


Gambar 3.14 *Activity Diagram* Metode Pengendalian Objek Ular



Gambar 3.15 *Activity Diagram* Metode Pergerakan Ular Lawan Gas Atar

Pada Gambar 3.15 di atas dapat dilihat bahwa aplikasi yang dikembangkan kini telah mendukung penerapan *searching* AI. Metode pergerakan lawan/musuh dengan algoritma pencarian terhadap umpan dengan penanaman *knowledge base* pada objek lawan/musuh, sehingga lawan/musuh akan terasa memiliki kecerdasan tersendiri dalam pergerakannya.

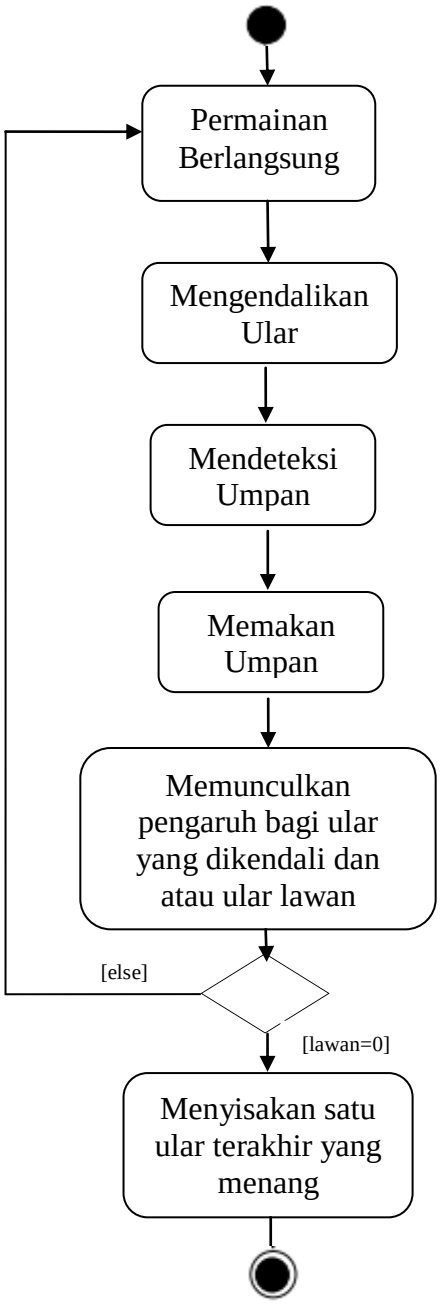


Gambar 3.16 *Activity Diagram* Metode *Collison Detection* Ular Lawan Gas Atar

Metode deteksi tabrakan ini akan diterapkan pada ular lawan, dimana pada *game snake* sebelumnya belum adanya deteksi tabrakan. Deteksi ini diterapkan pada saat ular lawan melakukan pengejaran terhadap umpan sampai ke pembatas dinding arena permainan, maka ular tersebut tidak terus melaju (menabrak dinding) tapi langsung berbelok ke arah umpan yang dekat dengan ular tersebut.

Metode memakan umpan pada aplikasi baru tidak mengalami perubahan pada aplikasi yang dikembangkan. Namun nantinya akan ada beberapa reaksi-reaksi yang

terjadi setelah memakan umpan-umpan tersebut, baik reaksi bagi ular yang memakannya ataupun bagi lawan ular tesebut.

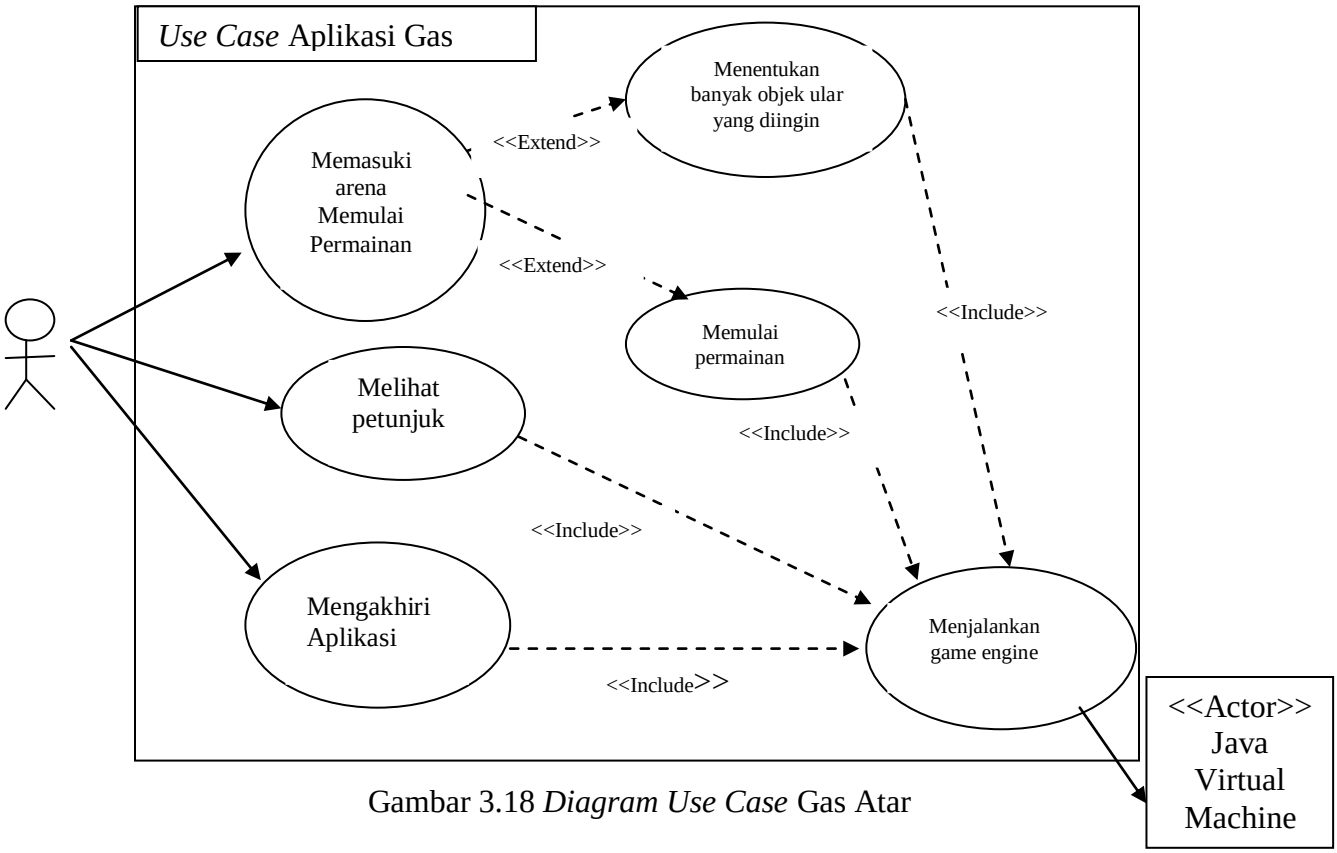


Gambar 3.17 *Activity Diagram* Metode Memakan Umpan Gas Atar

3.2.4 Analisis Kebutuhan Fungsional

3.2.4.1 *Diagram Use Case*

Secara umum, interaksi antara pengguna yaitu pemain dengan aplikasi pernainan Gas Atar digambarkan pada *diagram use case* Gambar 3.18 berikut ini:



Gambar 3.18 *Diagram Use Case* Gas Atar

3.2.4.2 Skenario *Use Case*

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

1. Skenario *Use Case* Memasuki arena Memulai Permainan (*Play*)

Skenario *use case* ini menjelaskan interaksi antaru aktor, yaitu pemain dengan *use case* mulai permainan yang dijelaskan pada Tabel 3.17 berikut ini:

Tabel 3.17 Skenario *Use Case* Memasuki arena Memulai Permainan (*Play*)

Identifikasi		
Nama	Memasuki arena Memulai Permainan (*Play*)	
Tujuan	Masuk ke arena permainan kemudian mulai bermain	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Menu Utama ditampilkan	
Aksi Aktor		Reaksi Sistem
a. Klik Play		
b. Klik Add Player		c. Menampilkan berapa pemain yang diinginkan dalam permainan nanti
d. Klik Stage 1		e. Menampilkan Arena Permainan dengan kecepatan rendah.
f. Klik Stage 2 dan Stage 3		g. Menampilkan Arena Permainan dengan kecepatan sedang dan dang paling cepat.
h. Klik Back		i. Menampilkan kembali menu utama
Kondisi akhir	Permainan dimulai	

2. Skenario *Use Case* Melihat Petunjuk (*How To Play*)

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* lihat petunjuk yang dijelaskan pada Tabel 3.18 berikut ini:

Tabel 3.18 Skenario *Use Case* Melihat Petunjuk (*How To Play*)

Identifikasi	
Nama	Melihat Petunjuk (*How To Play*)
Tujuan	Masuk ke submenu yang berisi mengenai petunjuk perrnainan
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 berrnain

3. Skenario *Use Case* Mengakhiri Aplikasi (*Quit*)

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* mengakhiri aplikasi yang dijelaskan pada Tabel 3.19 berikut ini:

Tabel 3.19 Skenario *Use Case* Mengakhiri Aplikasi (*Quit*)

Identifikasi		
Nama	Mengakhiri aplikasi (*Quit*)	
Tujuan	Masuk ke submenu untuk keluar dari aplikasi permainan	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Menu Utama ditampilkan	
Aksi Aktor		Reaksi Sistem
a. Klik Quit		
		b. Aplikasi Keluar
Kondisi akhir	*Game Snake* telah keluar	

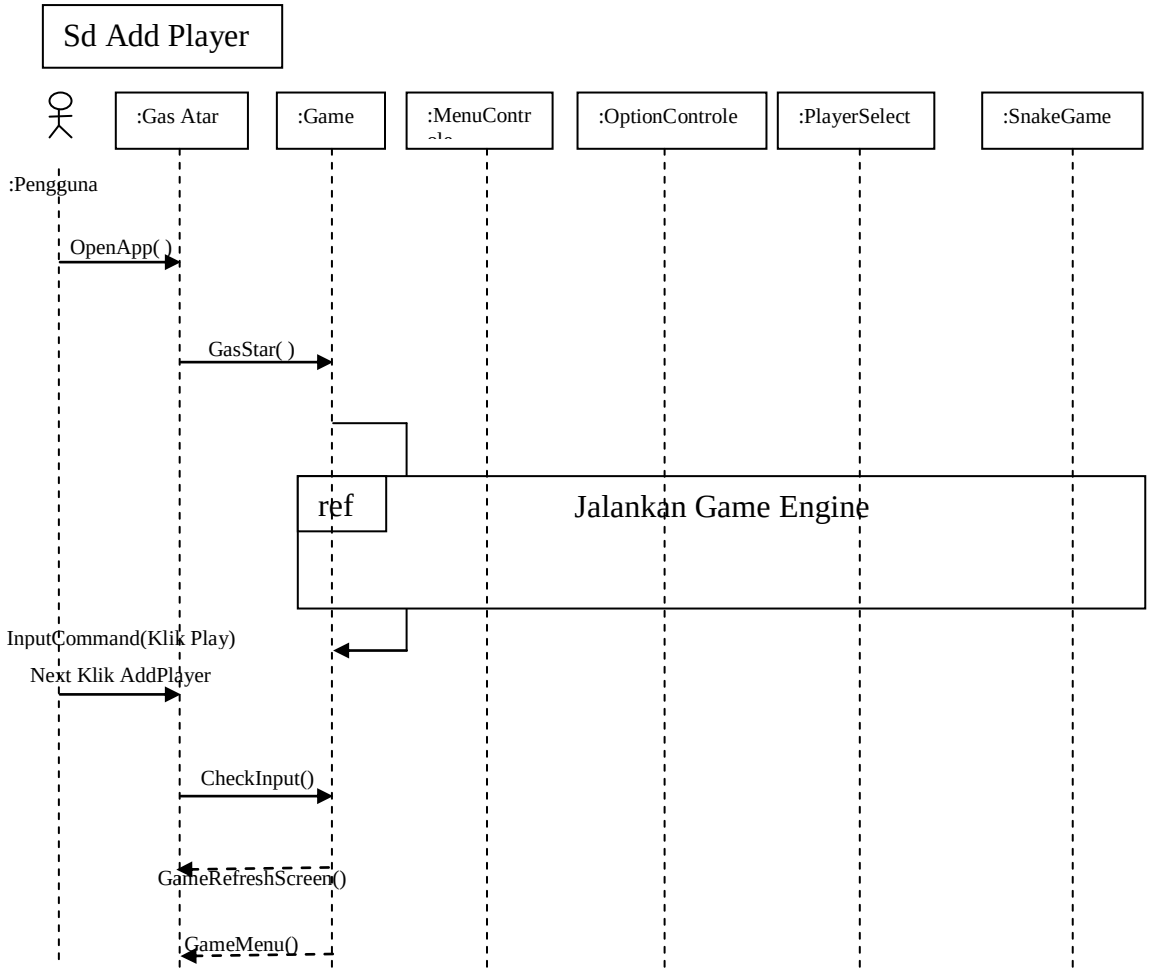
3.2.4.3 *Diagram Sequence*

Pada bagian ini berisi penjelasan dari masing-masing *use case* yang ada pada Gas Atar 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* Memulai Permainan (*Play*)

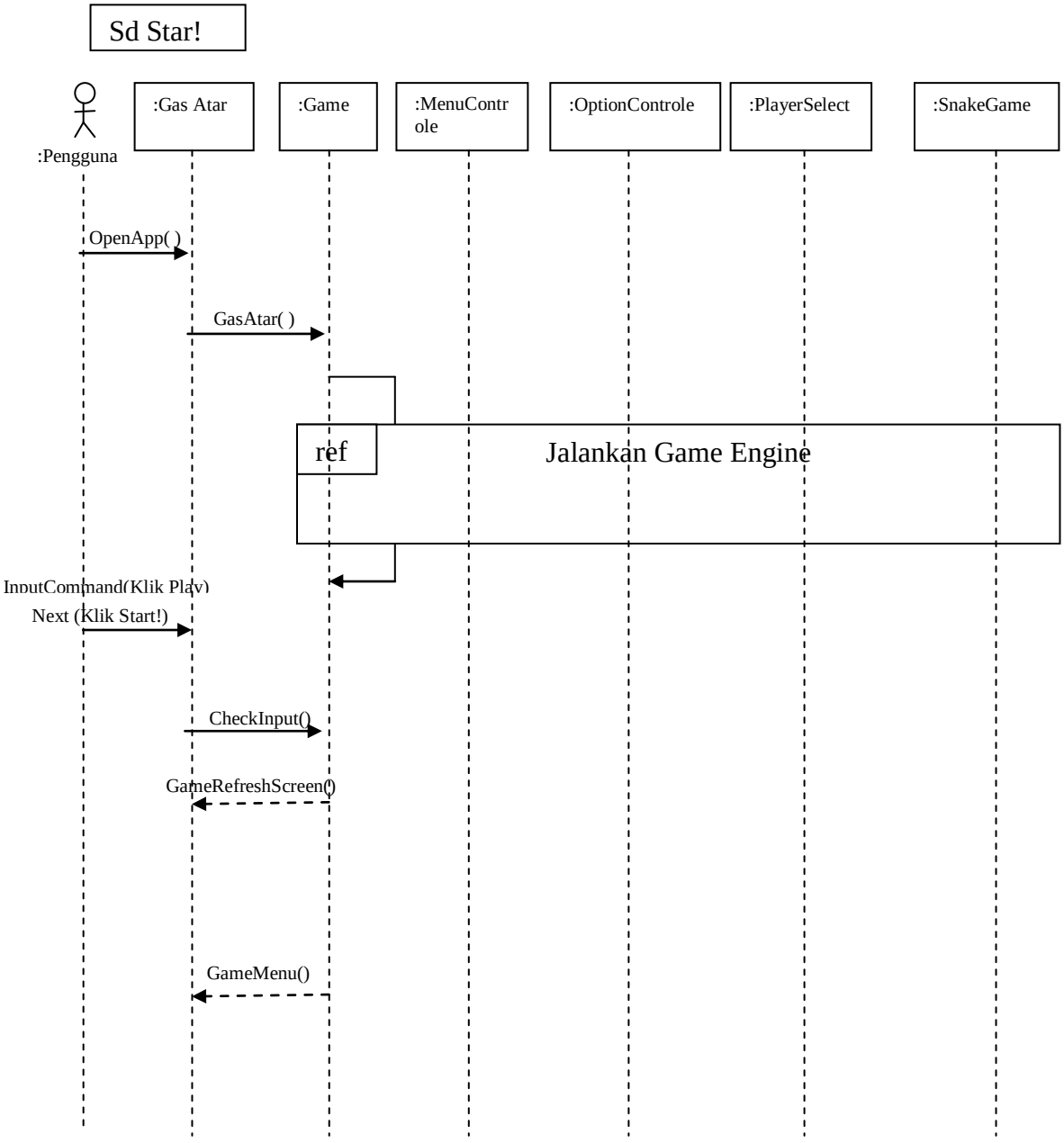
Berikut ini merupakan *diagram sequence* dari *use case* Memasuki arena Memulai Permainan, dimana akan ada 2 (dua) kegiatan yaitu menentu banyak ular yang diinginkan dan memulai permainan langsung tersebut :

a. *Diagram Sequence* Memasuki Arena Memulai Permainan (*Play*) dalam Menentukan banyak pemain yang diinginkan.



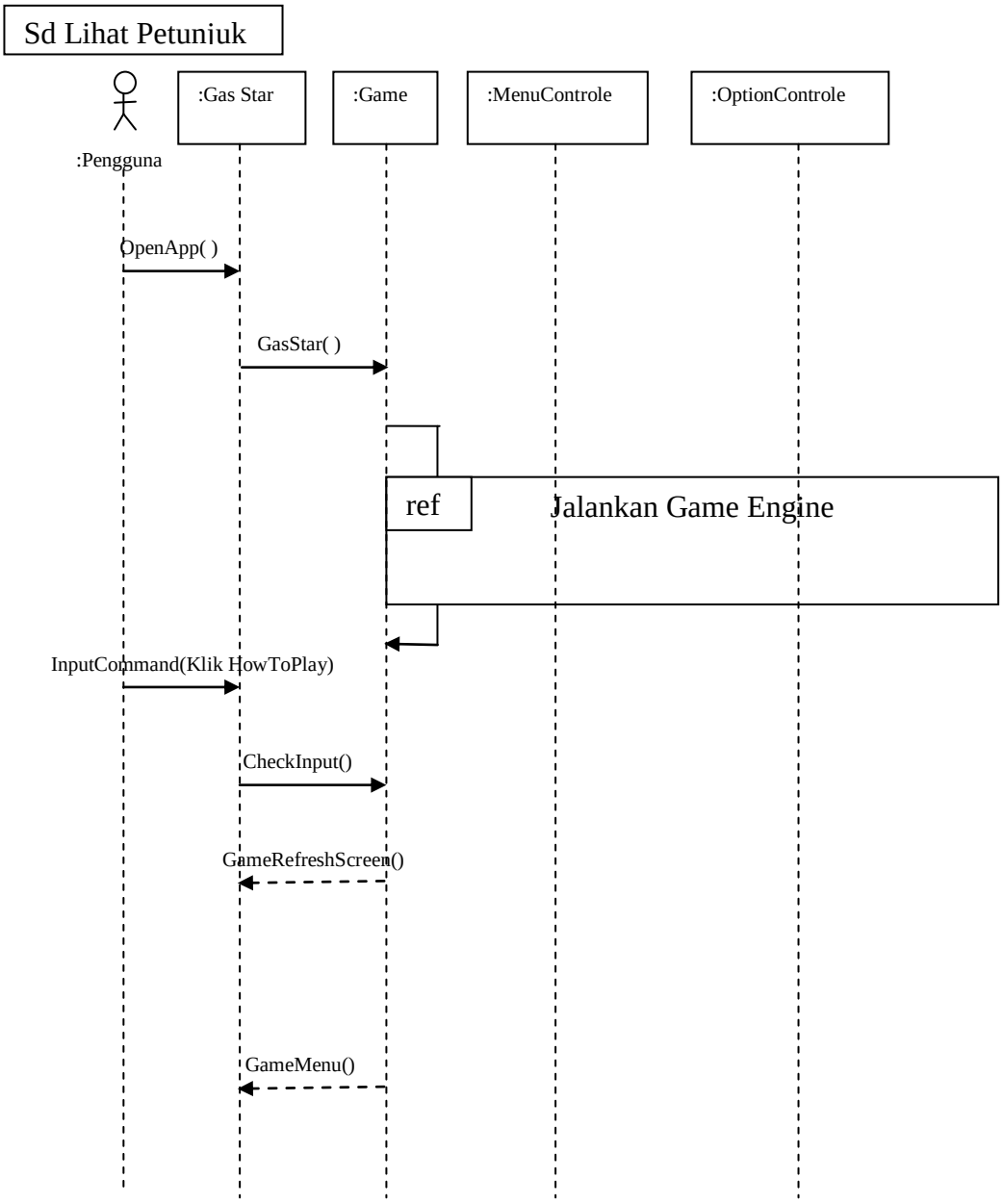
Gambar 3.19 *Diagam Sequence* Memulai Permainan (*Play*) dalam Menentukan banyak pemain yang diinginkan Aplikasi Gas Atar

b. *Diagram Sequence* Memasuki Arena Memulai Permainan (*Play*) dan Memulai Langsung Permainan dengan 3 Stage (kecepatan yang beda).



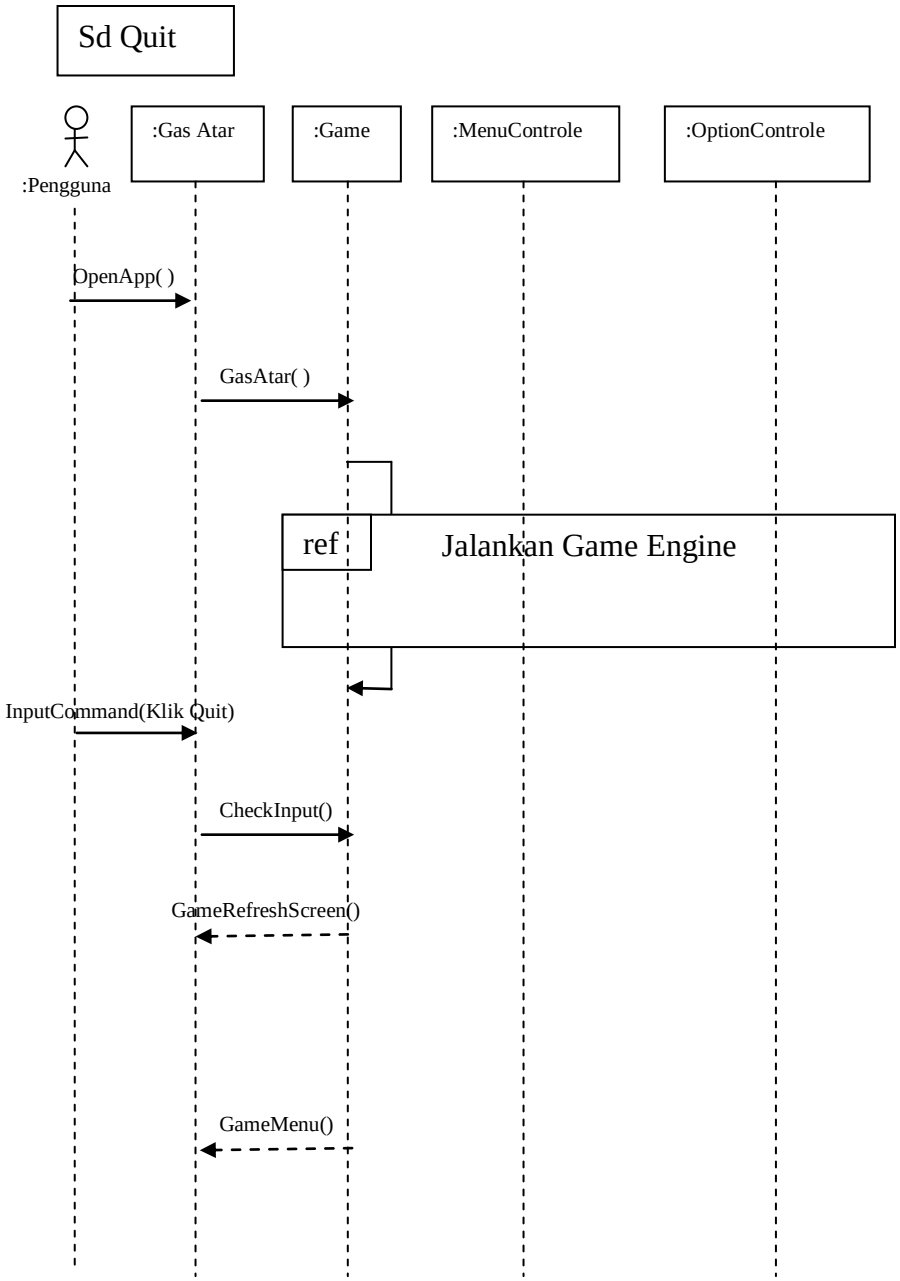
Gambar 3.20 *Diagam Sequence* Memulai Permainan (*Play*) dan Memulai Langsung Permainan Aplikasi Gas Atar

3.2.4.3.2 *Diagram Sequence* Melihat Petunjuk (*How To Play*)



Gambar 3.21 *Diagram Sequence* Melihat Petunjuk (*How To Play*) Gas Atar

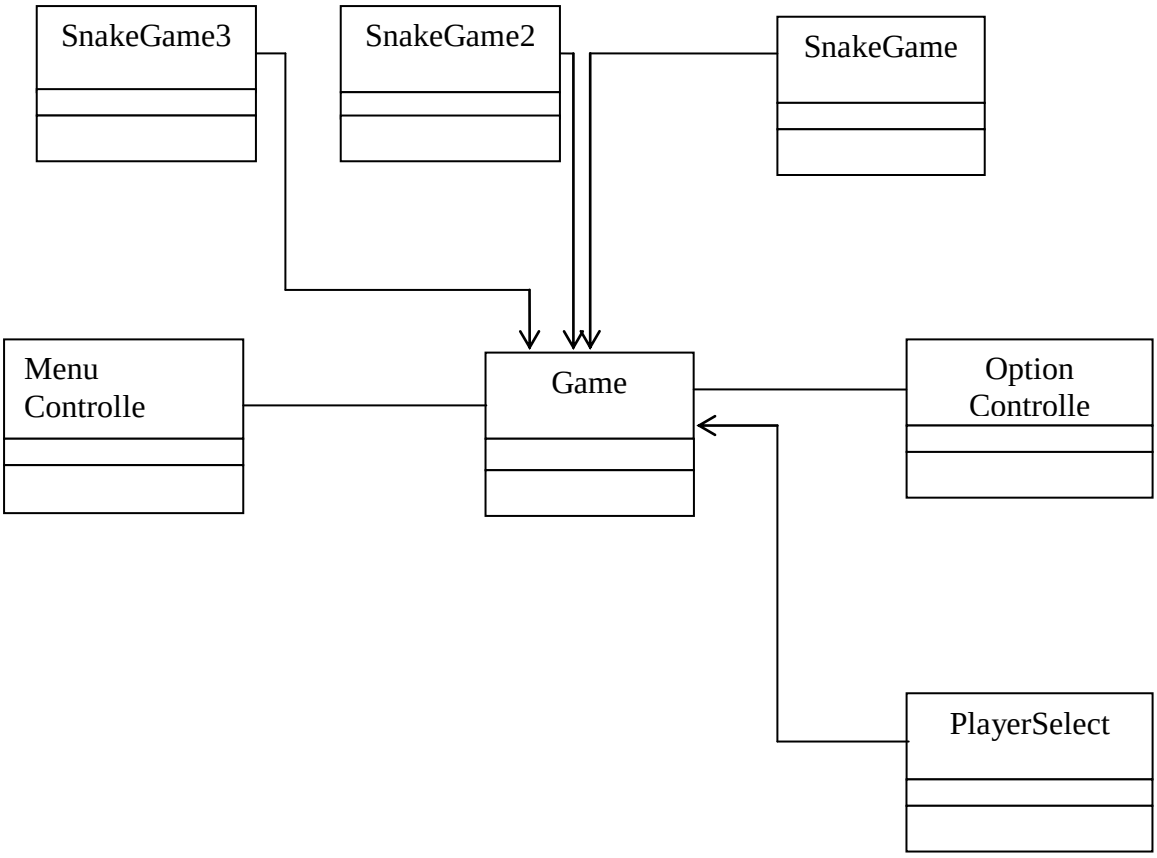
3.2.4.3.3 *Diagram Sequence* Mengakhiri Aplikasi (*Quit*)



Gambar 3.22 *Diagram Sequence* Mengakhiri Aplikasi (*Quit*) Gas Atar

3.2.4.4 *Diagram Class*

Diagram Class pada aplikasi Gas Atar dapat dilihat pada Garnbar 3.23 berikut ini :



Gambar 3.23 *Diagram Class* Gas Atar

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

MenuControlle
-game : Game -nifty : Nifty
+MenuController(Game) +back() : void +bind(Nifty, Screen) : void +exit() : void +onEndScreen() : void +onStartScreen() : void +Play(0 : void

Gambar 3.24 MenuControlle *Class* pada *Class Diagram* Gas Atar

OptionControlle
-game : Game -id : Int -nifty : Nifty -screen : Screen
+OptionController(Game) +AddPanel() : void +bind(Nifty, Screen) : void +onEndScreen() : void +onStartScreen() : void +play() : void +quit() : void +test() : void

Gambar 3.25 OptionControlle *Class* pada *Class Diagram* Gas Atar

PlayerSelect
+angle : double +NextColor : color -element : Element +game : Game -nifty : Nifty +player : Player -screen : Screen

+bind(Nifty, Screen, Element, Properties, Atributes) : void +bindGame(Game) : void +getPlayer() : Player +init(Properties, Atributes) : void +inputEvent(Nifty, InputEvent) : boolean +onFocus(boolean) : void +onMyListBoxSelectionChanged(StringListBpxSelectionChangedEvent<string>) : void +onStartScreen() : void +removePanel() : void

Gambar 3.26 PlayerSelect *Class* pada *Class Diagram* Gas Atar

Game
+Controllers : List <input> +FullScreen : boolean +GameSpeed : Float +renderer : lwjgrendererdevice +SCREEN_SIZE : int[] +VIEW_SIZE : Float[] +WINDOW_SIZE : int[] +hotFullScreen : boolean -MenuController : MenuController -OptionController : OptionController -Players : list<Player> +State : GameState
+Game() -getAvailableIstances(class<T>type) : list <T> +getMenuController() : MenuController +getNifty() : Nifty +getOptionController() : OptionController +getPlayer() : List<Player> -pollinput() : void +SetDisplayMode() :int -Start() : void +Main(string[]) : void

Gambar 3.27 Game *Class* pada *Class Diagram* Gas Atar

SnakeGame
+DIAGONAL : float +GRIDSIZE : int +MAXSIZE : int +SPEED : float +TILESIZE :float +WINDELAY : long +board : Board +finished : boolean +FPS : int -game : Game +gui : gameGUI +hotPause : boolean +lastFPS : long +lastTick : long +running : boolean
+getTime() : long +SnakeGame (Game) +getBoard() : Board -rollInput() : void +Start() : void -updateFPS() : void

Gambar 3.28 SnakeGame *Class* pada *Class Diagram* Gas Atar

SnakeGame2
+DIAGONAL : float +GRIDSIZE : int +MAXSIZE : int +SPEED : float +TILESIZE :float +WINDELAY : long +board : Board +finished : boolean +FPS : int -game : Game +gui : gameGUI +hotPause : boolean +lastFPS : long +lastTick : long

+running : boolean
+getTime() : long +SnakeGame (Game) +getBoard() : Board -rollInput() : void +Start() : void -updateFPS() : void

Gambar 3.29 SnakeGame2 *Class* pada *Class Diagram* Gas Atar

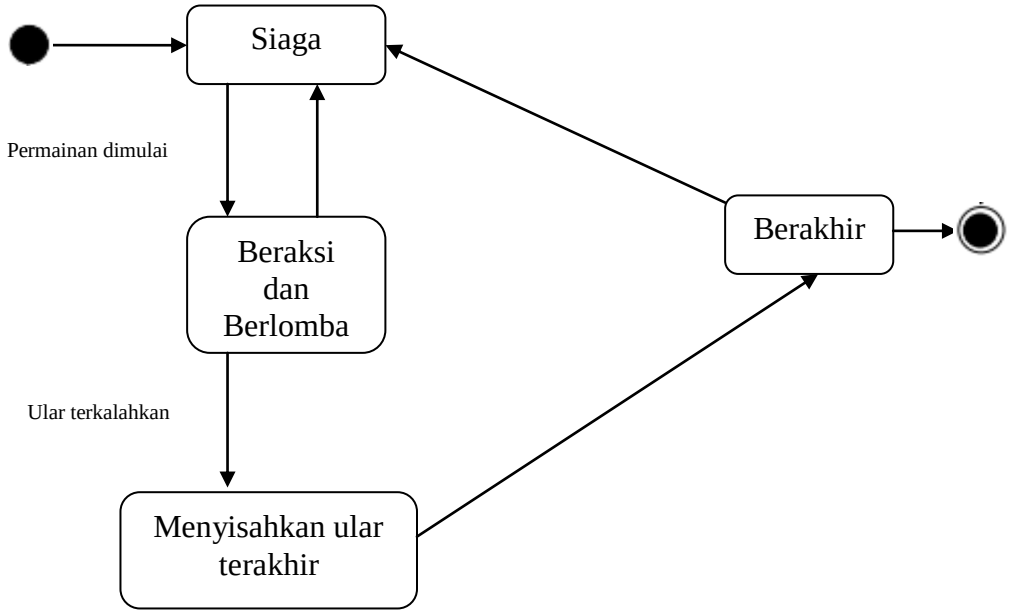
SnakeGame3
+DIAGONAL : float +GRIDSIZE : int +MAXSIZE : int +SPEED : float +TILESIZE :float +WINDELAY : long +board : Board +finished : boolean +FPS : int -game : Game +gui : gameGUI +hotPause : boolean +lastFPS : long +lastTick : long +running : boolean
+getTime() : long +SnakeGame (Game) +getBoard() : Board -rollInput() : void +Start() : void -updateFPS() : void

Gambar 3.30 SnakeGame3 *Class* pada *Class Diagram* Gas Atar

3.2.4.5 *Diagram State*

3.2.4.5.1 *Diagram State*

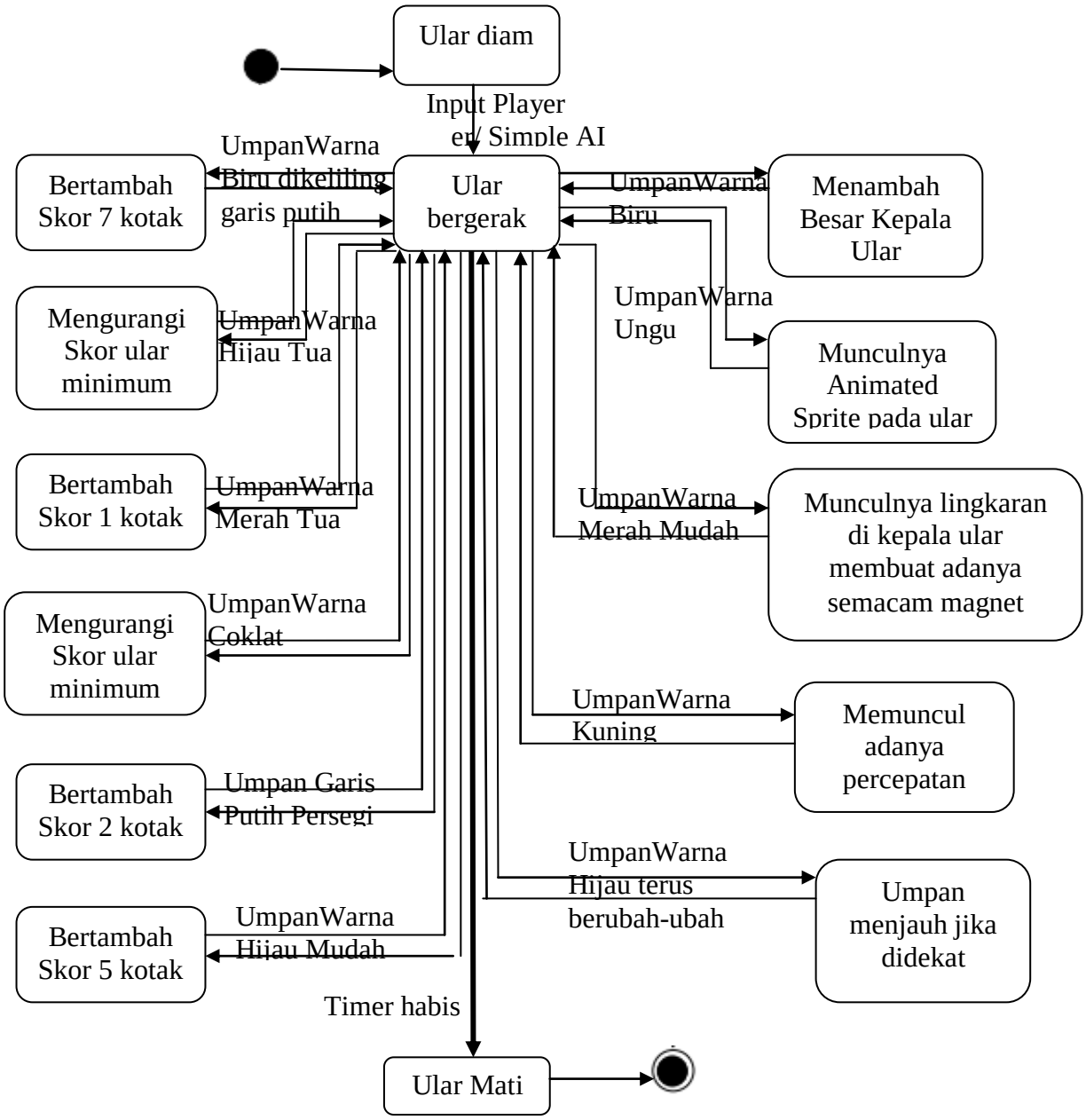
Adapun *state diagram* yang ada nantinya pada Gas Atar mengalami sedikit perbedaan yang dialami ular. Untuk lebih jelasnya dapat dilihat pada Gambar 3.31 Berikut ini :



Gambar 3.31 *State Diagram* Gas Atar secara Umum

Dari gambar 3.30 di atas dapat dilihat proses secara umum, dimana ular harus melakukan perlombaan secepat-cepatnya mencari makanan sehingga tidak terkalahkan dan menyebabkan kemenangan dengan menyisakan satu ular terakhir.

Adapun *State Diagram* yang menunjukkan proses ular AI A* dan Ular yang dikendalikan *user* melakukan permangsaan umpan dan mengakibatkan beberapa reaksi yang terjadi pada Gambar 3.32 Berikut ini.



Gambar 3.32 *State Diagram* ular memakan umpan Gas Atar

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:

l. Peran : Pemain

2. Usia : Lebih dari 5 tahun

3. Tingkat Keterampilan : Dapat menggunakan computer dan aplikasi komputer

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 pada tabel 3.20 berikut:

Tabel 3.20 Spesifikasi Perangkat Keras (Pengembang)

Nama Perangkat	Spesilikasi
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* pada Tabel 3.21 berikut :

Tabel 3.21 Spesifikasi Perangkat Lunak (Pengembang)

Nama Perangkat Lunak	Jenis
Sistem Operasi	Microsoft Windows 7 Prof 64 Bit
IDE	Eclips Helios
SDK	jdk-6u18-windows-i586.exe,

3.2.5.4 Analisis Perangkat Keras (Pengguna)

Pada sisi pengguna, perangkat keras yang dibutuhkan untuk menjalankan aplikasi ini adalah pada Tabel 3.22 berikut :

Tabel 3.22 Spesifikasi Perangkat Keras (Pengguna)

Nama Perangkat	Spesifikasi
Prosesor	Core Duo
RAM	1 GB
Hard Disk	40MB
Display	Resolusi 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 pada Tabel 3.23 berikut :

Tabel 3.23 Spesifikasi Perangkat Lunak (Pengguna)

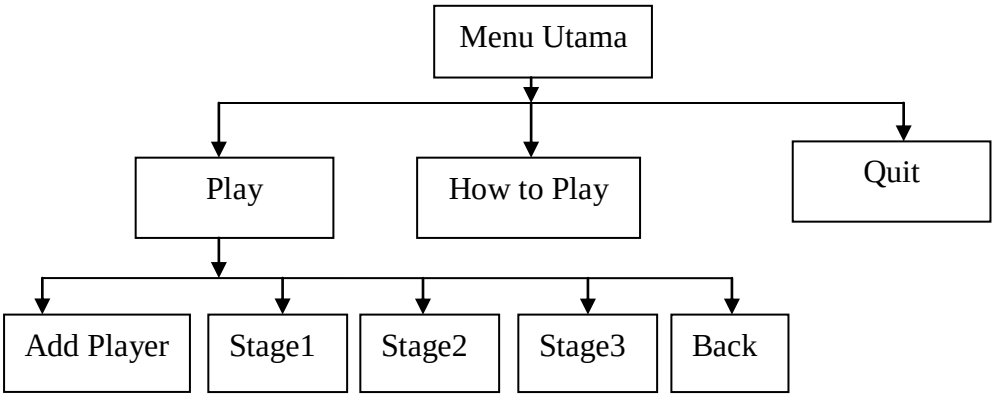
Nama Perangkat Lunak	Jenis
Sistem Operasi	Microsoft Windows XP
Utilities	JVM

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 3.33 di bawah ini menunjukkan perancangan arsitektur menu dari sistem yang akan dibangun.



Gambar 3.33 Arsitektur Menu Aplikasi Gas Atar

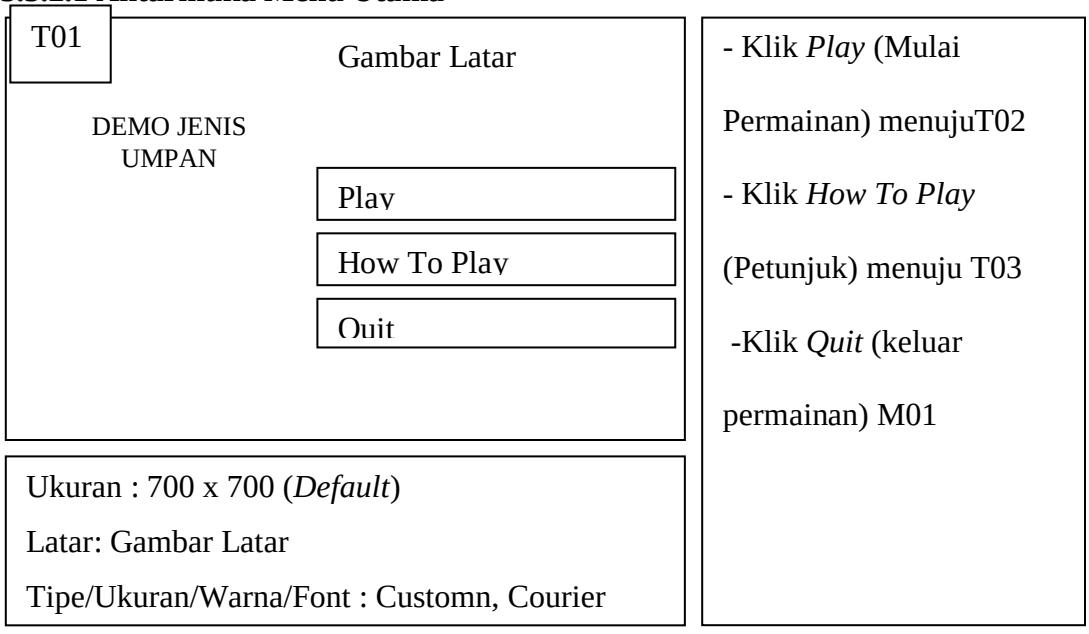
1. *Play* (Mulai Permainan) : Submenu untuk memulai *game snake*.
2. *How to Play* (Petunjuk Permainan) : Submenu untuk melihat petunjuk.
3. *Quit* (Keluar) : Submenu untuk keluar dari aplikasi *game snake*.

4. *Add Player* : Menambah/menentukan jumlah pemain yang diinginkan.
5. *Stage1* : Menuju langsung ke laman permainan dengan tingkat kecepatan paling rendah.
6. *Stage2* : Menuju langsung ke laman permainan dengan tingkat kecepatan sedang.
7. *Stage2* : Menuju langsung ke laman permainan dengan tingkat kecepatan paling rendah.

3.3.2 Perancangan Antarmuka

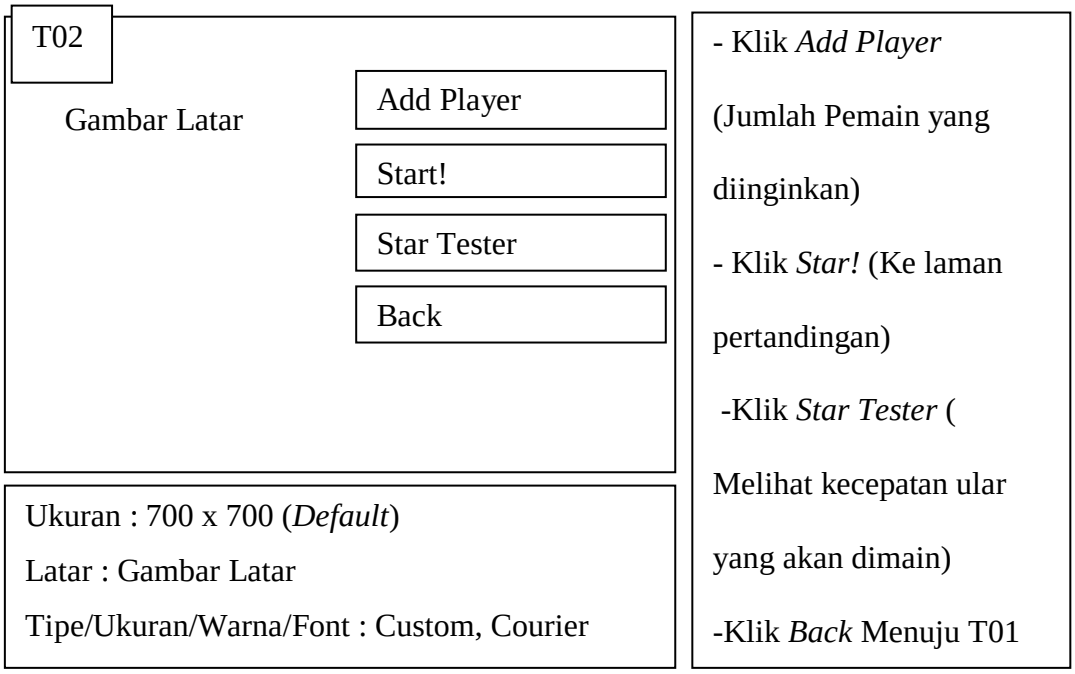
User interface (antarmuka pengguna) adalah aspek sistem komputer atau program yang dapat dilihat didengar atau dipersepsikan oleh pengguna dan perintah-perintah alau mekanisme yang digunakan mengendalikan operasi dalam aplikasi.

3.3.2.1 Antarmuka Menu Utama



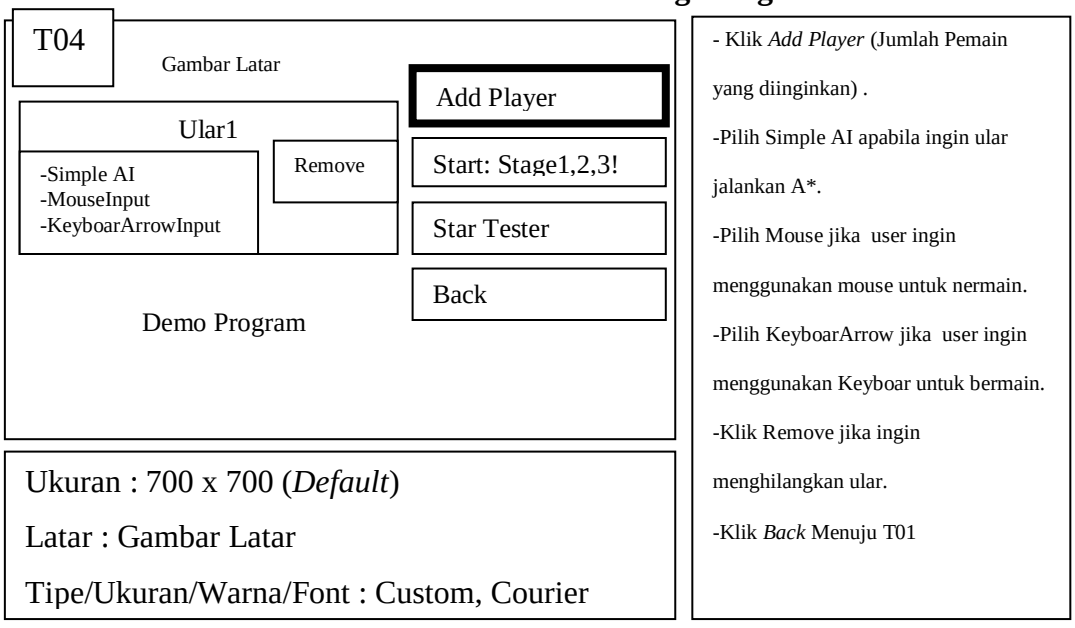
Gambar 3.34 Antarmuka Menu Utama

3.3.2.2 Antarmuka Menu Memulai Permainan



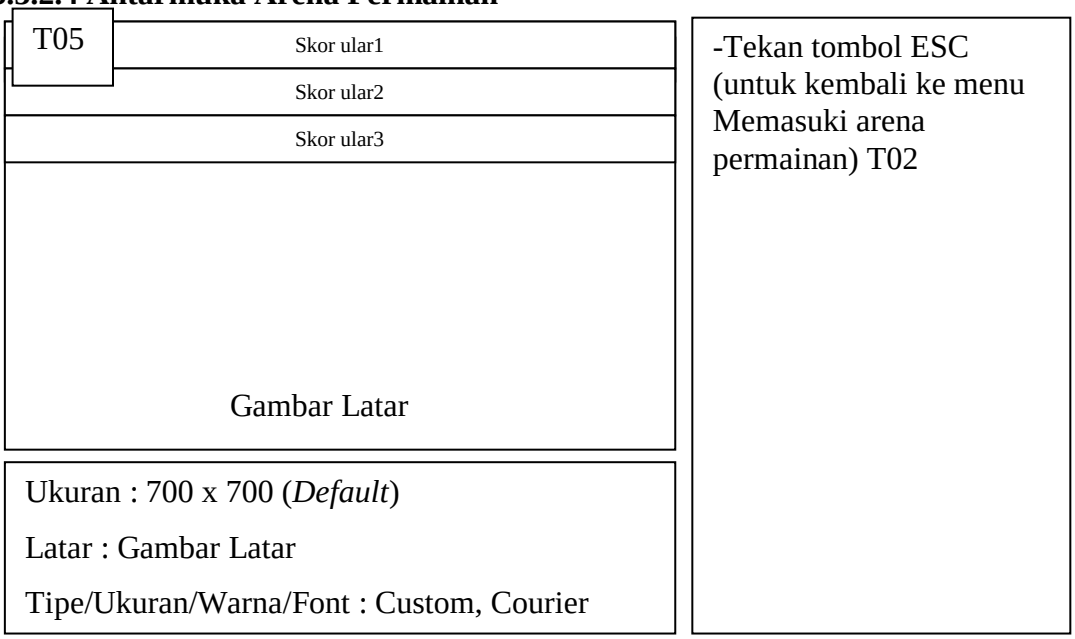
Gambar 3.35 Antarmuka Memulai Permainan

3.3.2.3 Antarmuka Menentukan Jumlah Ular Yang Diinginkan



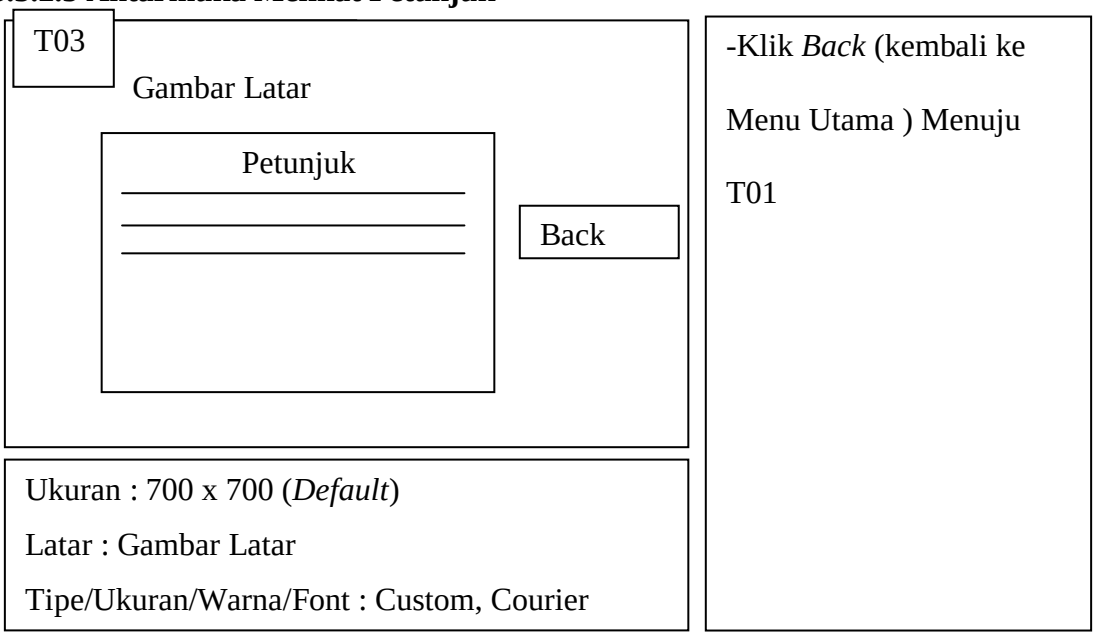
Gambar 3.36 Antarmuka Menentukan Jumlah Ular yang Diinginkan

3.3.2.4 Antarmuka Arena Permainan



Gambar 3.37 Antarmuka Arena Permainan

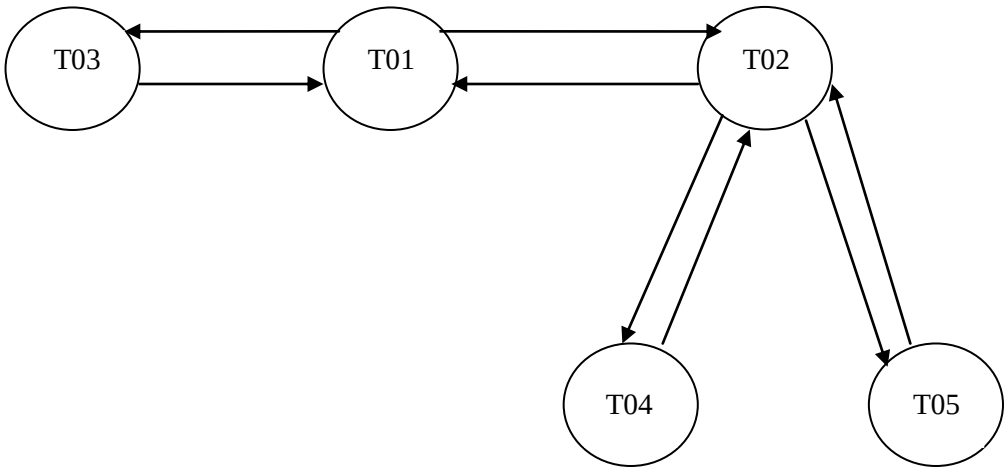
3.3.2.5 Antarmuka Melihat Petunjuk



Gambar 3.38 Antarmuka Melihat Petunjuk

3.3.3 Jaringan Semantik

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



Gambar 3.39 Jaringan Semantik Aplikasi Gas Atar