BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi permainan hasil pengembangan yaitu Gas Atar. Tahapan ini dilakukan setelah tahap perancangan selesai dilakukan. Adapun hasil dari perancangan tersebut kemudian diimplementasikan pada bahasa pemrograman. Setelah implementasi, maka dilakukan pengujian terhadap aplikasi hasil pengembangan dan akan dilihat kekurangan-kekurangannya untuk pengembangan aplikasi tersebut selanjutnya. Pengujian tersebut terdiri dari pengujian alpha dan beta.

4.1 Implementasi

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi. Tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan, yang terdiri dari penjelasan mengenai lingkungan implementasi, dan implementasi program.

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang akan digunakan untuk menjalankan aplikasi permainan Gas Atar ini tidak harus komputer yang berspesifikasi tinggi. Hal itu disebabkan oleh penggunaan sumber daya yang dioptimalkan pada CPU, walaupun aplikasi ini sebenarnya mengimplementasikan *multithreading*.

Sebagai ilustrasi sistem minimum yang dapat menjalankan aplikasi adalah sebagai berikut:

1. *Processor* dengan kecepatan - lGHz
2. RAM dengan kapasitas lGB
3. VGA 32MB *onboard*
4. *Harddisk* 80 GB
5. Monitor dengan resolusi 1024 X 768
6. *Mouse*
7. *Keyboard*

4.1.2 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk menjalankan aplikasi permainan Gas Atar ini adalah sebagai berikut :

1. Sistem Operasi berbasis 32 bit
2. JVM (*Java Virtual Machine*)

4.1.3 Implementasi Aplikasi

Gas Atar adalah aplikasi permainan berjenis *game snake* yang merupakan pengembangan dari aplikasi *game snake* konvensional. Oleh karena itu ada beberapa hal permainannya mirip dengan aplikasi permainan konvensional. Pengguna aplikasi berperan sebagai pemain yang mengendalikan karakter berupa

ular yang berada di area yang disekitarnya terdapat banyak umpan yang nantinya jadi rebutan antara ular yang dikendalikan user dan ular yang diterapkan AI . Karena adanya rebutan maka disini ada beberapa ular yang dibedakan dengan warna-warna yang sangat jelas, maksimum ular yang dapat dimainkan ada 6 ular. Untuk umpannya sendiri memiliki 11 jenis warna dan reaksi-reaksi yang berbeda jika ular melakukan pemangsaan terhadap umpan tersebut.

Aplikasi ini berbasis desktop yang dibangun dengan menggunakan bahasa pemrograman JAVA. Oleh karena itu, aplikasi permainan ini dapat dijalankan di platform apapun yang telah dipasangi JVM (*Java Virtual Machine*) tentunya. Untuk menjalankan aplikasi ini dilakukan instalasi JVM dengan minimal JVM 4.0, dan juga adanya aplikasi *Eclips helios*. Apabila JVM belum terpasang, maka aplikasi ini pun tak akan bisa dijalankan.

Berikut ini merupakan fungsionalitas pada aplikasi permainan Gas Atar beserta sejumlah *file* yang merupakan komponen pengimplementasiannya:

Tabel 4.1 Tabel Implementasi Aplikasi

Nama Fungsionalitas	Nama *File*
Menampilkan Menu Utama	Game.java
	MenuController.java
	OptionController.java
	PlayerSelect.java
Menampilkan Permainan Baru	Game.java

	MenuController.java
	OptionController.java
	PlayerSelect.java
	SnakeGame.java

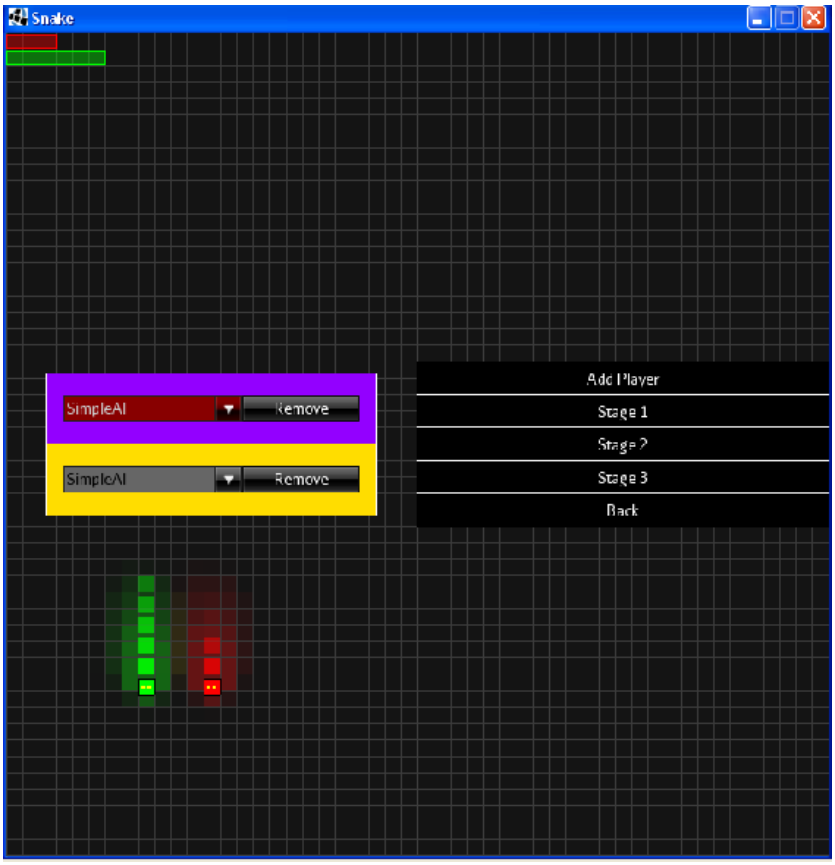
4.1.4 Implementasi Antarmuka

Antarmuka aplikasi ini terdiri dari beberapa bagian seperti yang akan dijelaskan pada sub bab ini, sedangkan sisanya bisa dilihat di lampiran antarmuka. Saat pengguna membuka aplikasi pertama kalinya, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka menu utama seperti pada Gambar 4.1:



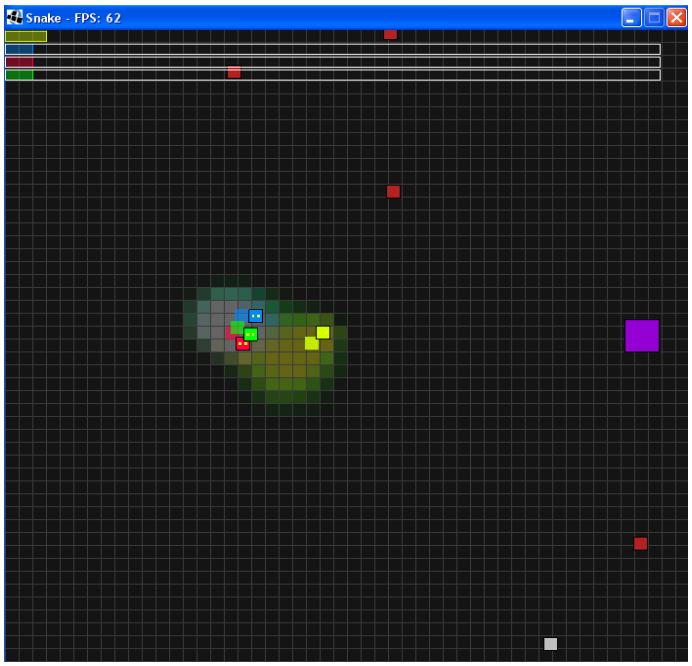
Gambar 4.1 Tampilan Menu Utama Gas Atar

Jika pengguna ingin bermain, cukup dengan mengklik *play* kemudian ada tampilan berapa pemain yang diinginkan dengan mengklik *add player* dan laman (level kecepatan/*stage*) yang bisa dipilih dengan mengklik *stage* yang diinginkan pada Gambar 4.2:



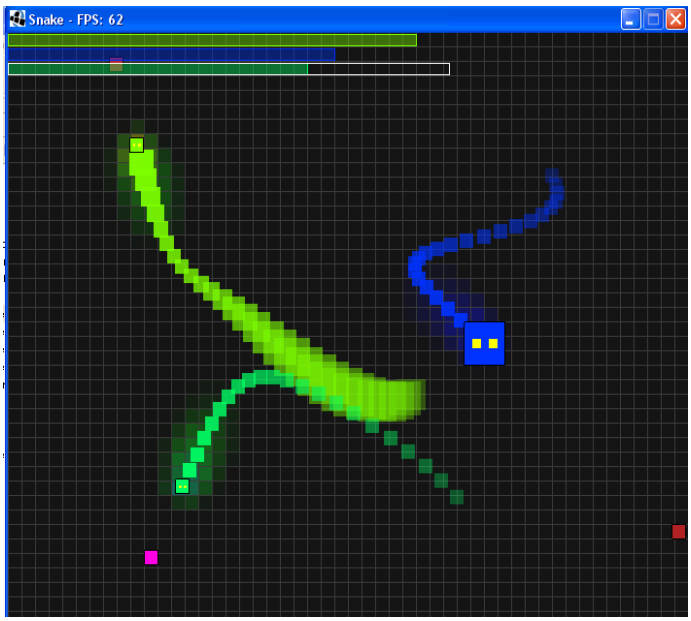
Gambar 4.2 Menu Pilihan Pemain dan Level Gas Atar

Setelah melakukan pemilihan berapa ular yang diinginkankan, maka jika pengguna ingin langsung bermain, cukup dengan mengklik stage (level kecepetan) yang diinginkan, dan akan tampilannya seperti yang terlihat pada gambar 4.2:



Gambar 4.3 Permainan Gas Atar

Perubahan *state* entitas permainan pada aplikasi Gas Atar dapat dilihat pada Gambar 4.4:



Gambar 4.4 Reaksi perubahan State Aplikasi Gas Atar

4.2 Pengujian Perangkat Lunak

Pengujian yang akan digunakan dalam menguji aplikasi Gas Atar ini adalah dengan menggunakan metode pengujian *white box dan black box*.

4.2.1 Pengujian Alpha

Rencana pengujian yang akan dilakukan dengan menguji sistem menggunakan metode *white box* untuk mengetahui apakah setiap *node* pada seluruh *independent paths* dikerjakan, serta untuk menguji alur logika, perulangan, dan validitas parameter masukannya. Berikut ini tabel rencana pengujian dari aplikasi Gas Atar. Sedankan Metode *Black box* untuk mengetahui kemungkinan kesalahn yang terjadi terhadap aplikasi.

4.2.1.1 Pengujian Alpha *White Box*

Pengujian *white box* merupakan pengujian yang didasarkan pada pengecekan terhadap detail perancangan menggunakan struktur kontrol dari desain aplikasi secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur sistem, pengetahuan mengenai aplikasi digunakan untuk mengidentifikasikan kasus uji tambahan.

Tujuan penggunaan *white box* adalah untuk menguji semua *statement* program. Penggunaan metode pengujjan *white box* dilakukan untuk:

a. Memberikan jaminan bahwa semua *independent paths* suatu modul dikerjakan minimal satu kali.

b. Semua keputusan logis untuk kondisi *true* atau *false* pada aplikasi telah dikerjakan.
c. Seluruh kondisi perulangan pada aplikasi telah dikerjakan berdasarkan pada batasan nilai tertentu.
d. Memberikan jaminan bahwa stnrktur data intemal telah digunakan untuk menjamin validitas jalur keputusan.

Apabila setiap *independen paths* telah dikerjakan, maka aplikasi yang bersangkutan telah benar. Tetapi jika terdapat satu saja *node* pada suatu *independent paths* yang tidak dikerjakan, maka masih terdapat kesalahan pada aplikasi tersebut.

Tabel 4.2 Rencana Pengujian *White Box* Aplikasi Gas Atar

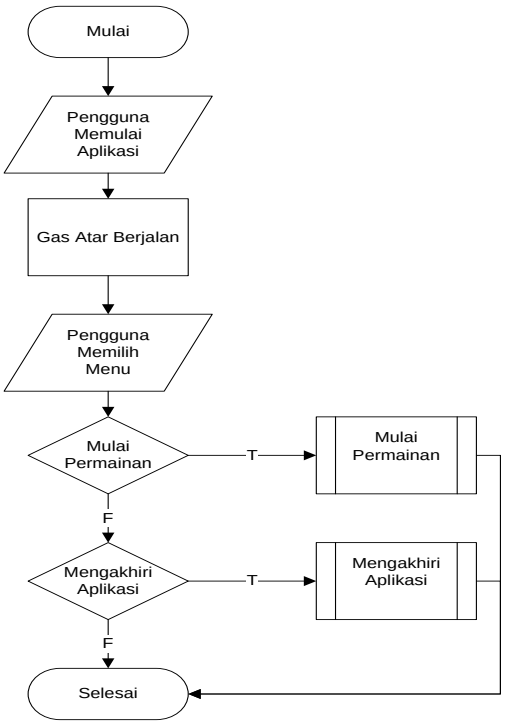
Kelas Uji	**Kasus Uji**	**Jenis Pengujian**
Menu Permainan	Reaksi Aplikasi terhadap masukkan dari pengguna	*White Box*
Interaksi Pengguna	Reaksi Entitas pesawat terhadap perintah masukkan dari pengguna	*White Box*
Alur Permainan	Reaksi keseluruhan Entitas yang terlibat serta pengaruhnya terhadap alur permainan	*White Box*
Algoritma A* (Star)	Reaksi yang terjadi pada Entitas ular lawan terhadap pencarian umpan	*White Box*
Collision Detection	Reaksi yang terjadi pada Entitas ular lawan dalam mendeteksi tabrakan terhadap dinding pembatas	*White Box*

Reaction Algorithm	Reaksi yang terjadi pada Entitas ular setelah memakan umpan dengan pengaruh bagi ular itu sendiri	*White Box*
Reaction Algorithm	Reaksi yang terjadi pada Entitas ular setelah memakan umpan dengan pengaruh terhadap ular lain yang tidak memakan	*White Box*

4.2.1.1.1 Menu Permainan

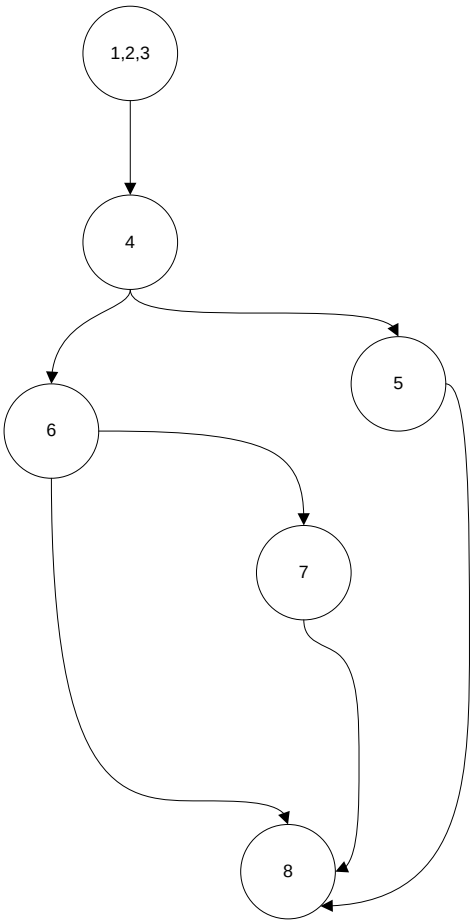
Pengujian menu permainan dilakukan untuk melihat apakah aplikasi akan menampilkan menu yang sesuai dengan yang diinginkan pengguna.

a. FIow Chart



Gambar 4.5 *Flow Chart* Pengujian Menu Permainan

b. Flow Graph



Gambar 4.6 *Flow Graph* Pengujian Menu Permainan

c. *Cyclomatic Complexity* V(G)

E: 7, N = 6

V(G) = E-N + 2 = 7 - 6 + 2 = 1+ 2 = 3

d. Independent Paths

Path l = 1,2,3-4-5-8

Path 2 =1,2,3-4-6-7-8

Path 3 =1,2,3-4-6-8

e. Hasil Uji

Tabel 4.3 Hasil Pengujian *Wite Box* Kelas Uji Menu Permainan

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,2,3,(void) 4,(true) 5,(void) 8,(void)	Permainan dimulai	Permainan dimulai	*Path* dikerjakan
2	1,2,3,(void) 4,(false) 6, (true) 7,(void) 8,(void)	Mengakhiri aplikasi	Mengakhiri aplikasi	*Path* dikerjakan
3	1,2,3,(void) 4,(false) 6,(false) 8,(void)	Menampilkan menu utama	Menampilkan menu utama	*Path* dikerjakan

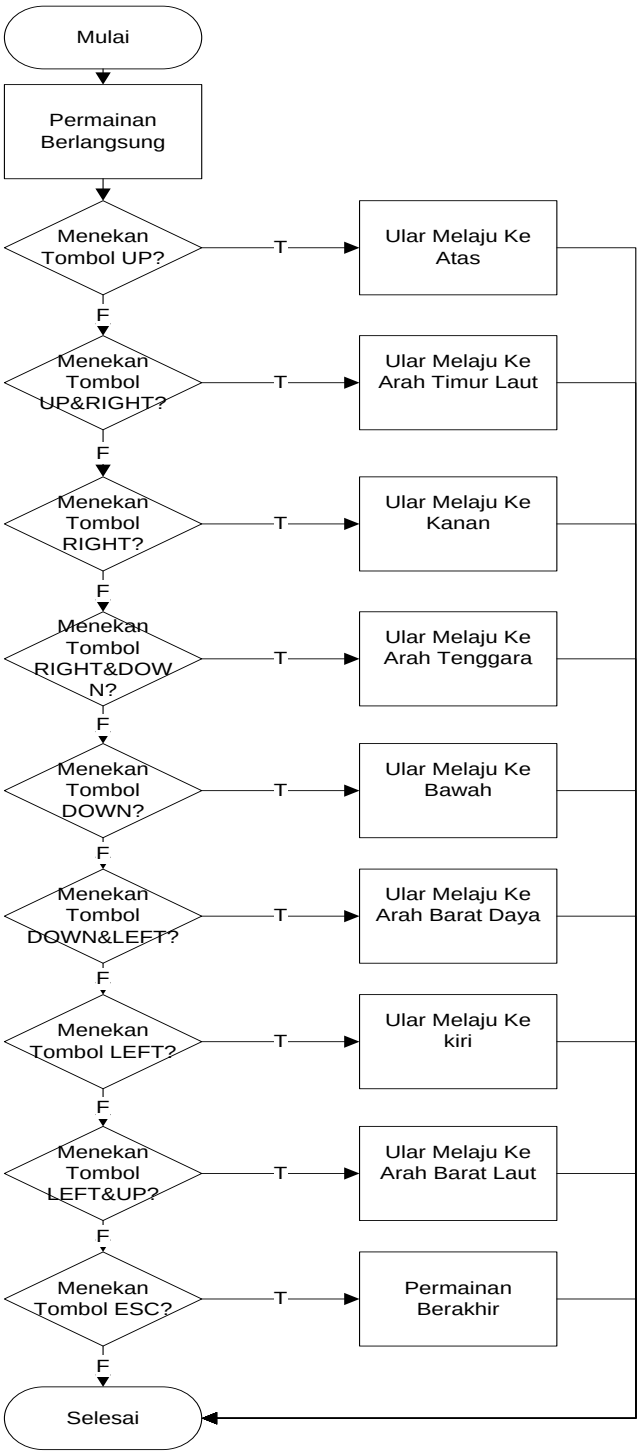
f. Kesimpulan

Dari hasil pengujian menu permainan yang dilakukan didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

4.2.1.1.2 Interaksi Pengguna

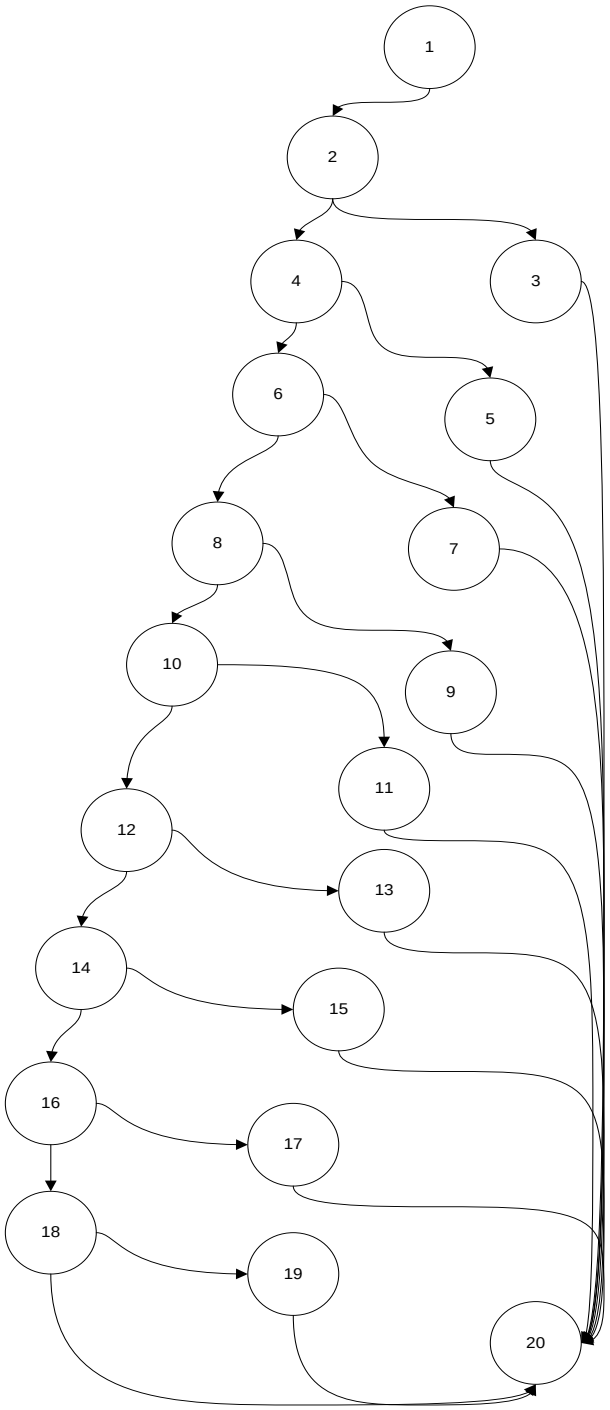
Pengujian interaksi pengguna dilakukan untuk melihat apakah aplikasi menjalankan perintah sesuai yang dimasukkan oleh pengguna. Hal tersebut dapat dilhat pada reaksi entitas ular pada arena permainan.

a. Flow Chart



Gambar 4.7 *Flow Chart* Pengujian Interaksi Pengguna

b. Flow Graph



Gambar 4.8 *Flow Graph* Pengujian Interaksi Pengguna

c. *Cyclomatic Complexity* V(G)

E: 27, N = 20

V(G) = E-N + 2 = 27 - 20+ 2 = 7+ 2 = 9

d. Independent Paths

Path l = l-2-3-20

Path 2 = l-2-4-5-20

Path 3 = l-2-4-6-7-20

Path 4 = l-2-4-6-8-9-20

Path 5 = l-2-4-6-8-10-11-20

Path 6 = 1-2-4-6-8-10-12-13-20

Path 7 = 1-2-4-6-8-10-12-14-15-20

Path 8 = 1-2-4-6-8-10-12-14-16-17-20

Path 9 = 1-2-4-6-8-10-12-14-16-18-19-20

e. Hasil Uji

Tabel 4.4 Hasil Pengujian *White Box* Kelas Uji Interaksi Pengguna

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,(void) 2,(true) 3,(void) 20,(void)	Ular Melaju ke atas	Ular Melaju ke atas	*Path* dikerjakan

2	1,(void) 2,(false) 4,(true) 5,(void) 20,(void)	Ular Melaju ke arah timur laut	Ular Melaju ke arah timur laut	*Path* dikerjakan
3	1,(void) 2,(false) 4,(false) 6,(true) 7,(void) 20,(void)	Ular Melaju ke arah kanan	Ular Melaju ke arah kanan	*Path* dikerjakan
4	1,(void) 2,(false) 4,(false) 6,(false) 8,(true) 9,(void) 20,(void)	Ular Melaju ke arah tenggara	Ular Melaju ke arah tenggara	*Path* dikerjakan
5	1,(void) 2,(false) 4,(false) 6,(false) 8,(false) 10,(true) 11,(void) 20,(void)	Ular Melaju ke bawah	Ular Melaju ke bawah	*Path* dikerjakan
6	1,(void) 2,(false) 4,(false) 6,(false) 8,(false) 10,(false) 12,(true) 13,(void) 20,(void)	Ular Melaju ke arah barat daya	Ular Melaju ke arah barat daya	*Path* dikerjakan
7	1,(void) 2,(false) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(true) 15,(void) 20,(void)	Ular Melaju ke kiri	Ular Melaju ke kiri	*Path* dikerjakan

8	1,(void) 2,(false) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(false) 16,(true) 17,(void) 20,(void)	Ular Melaju ke arah barat laut	Ular Melaju ke arah barat laut	*Path* dikerjakan
9	1,(void) 2,(false) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(false) 16,(false) 18,(true) 19,(void) 20,(void)	Permainan berakhir	Permainan berakhir	*Path* dikerjakan

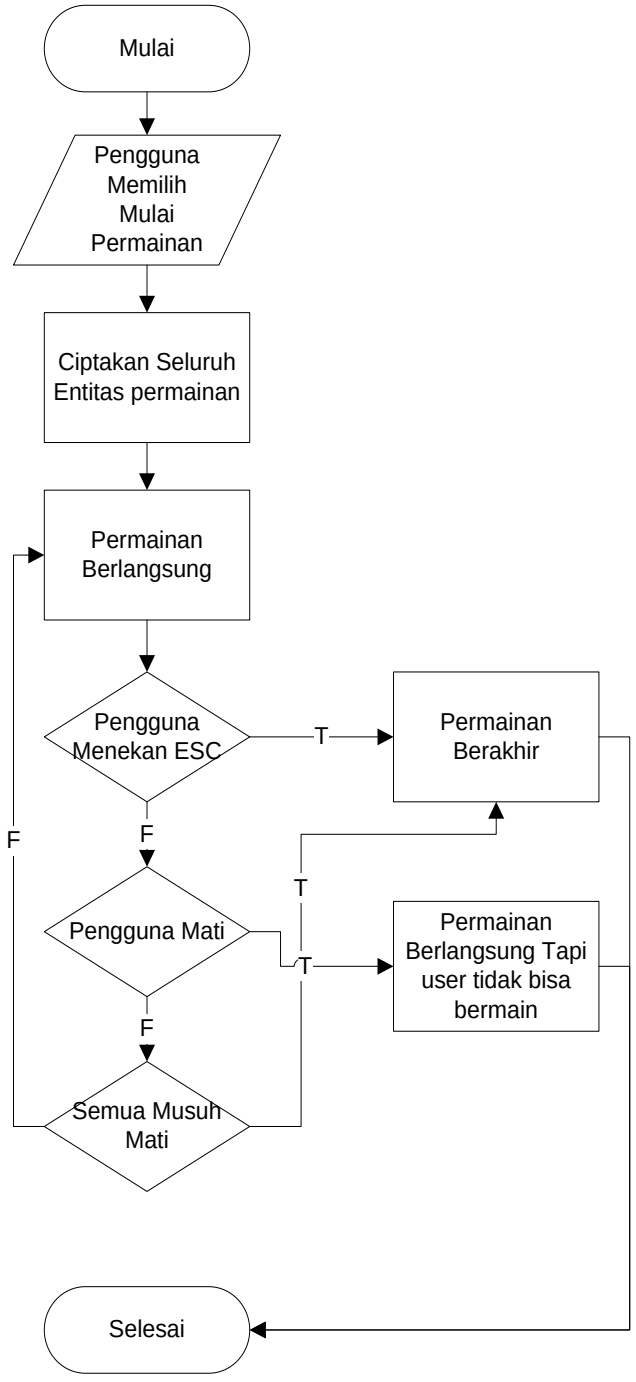
f. Kesimpulan

Dari hasil pengujian interaksi pengguna yang dilakukan, didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

4.2.1.1.3 Alur Permainan

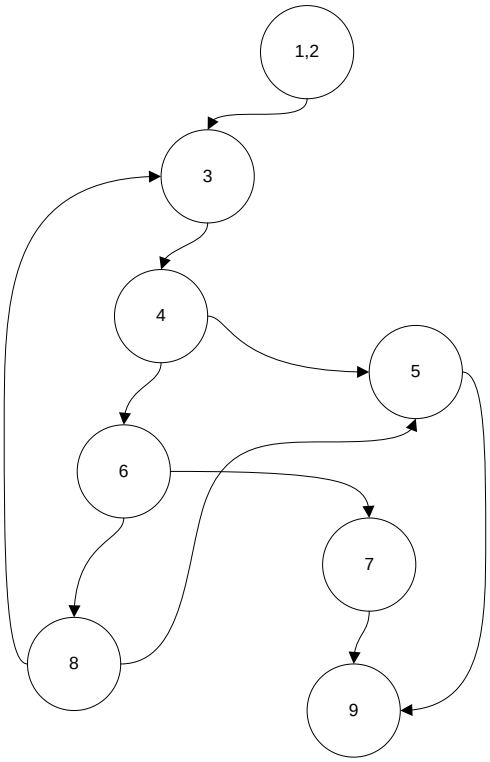
Pengujian alur permainan dilakukan untuk melihat apakah alur permainan pada aplikasi telah berjalan sesuai dengan yang diharapkan.

a. FIow Chart



Gambar 4.9 *Flow Chart* Pengujian Alur Permainan

b. Flow Graph



Gambar 4.10 *Flow Graph* Pengujian Alur Permainan

c. *Cyclomatic Complexity* V(G)

E: 10, N = 8

V(G) = E-N + 2 = 10 - 8 + 2 = 2+ 2 = 4

d. Independent Paths

Path l = 1,2-3-4-5-9

Path 2 = 1,2-3-4-6-7-9

Path 3 = 1,2-3-4-6-8-5-9

Path 4 = 1,2-3-4-6-8-3

e. Hasil Uji

Tabel 4.5 Hasil Pengujian *Wite Box* Kasus Uji Alur Permainan

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,2,(void) 3,(void) 4,(true) 5,(void) 9,(void)	Permainan berakhir	Permainan berakhir	*Path* dikerjakan
2	1,2,(void) 3,(void) 4,(false) 6,(true) 7,(void) 9,(void)	Permainan Berlangsung Tapi user tidak bisa bermain	Permainan Berlangsung Tapi user tidak bisa bermain	*Path* dikerjakan
3	1,2,(void) 3,(void) 4,(false) 6,(false) 8,(true) 5,(void) 9,(void)	Permainan berakhir	Permainan berakhir	*Path* dikerjakan
4	1,2,(void) 3,(void) 4,(false) 6,(false) 8,(false) 3,(void)	Permainan berlangsung	Permainan berlangsung	*Path* dikerjakan

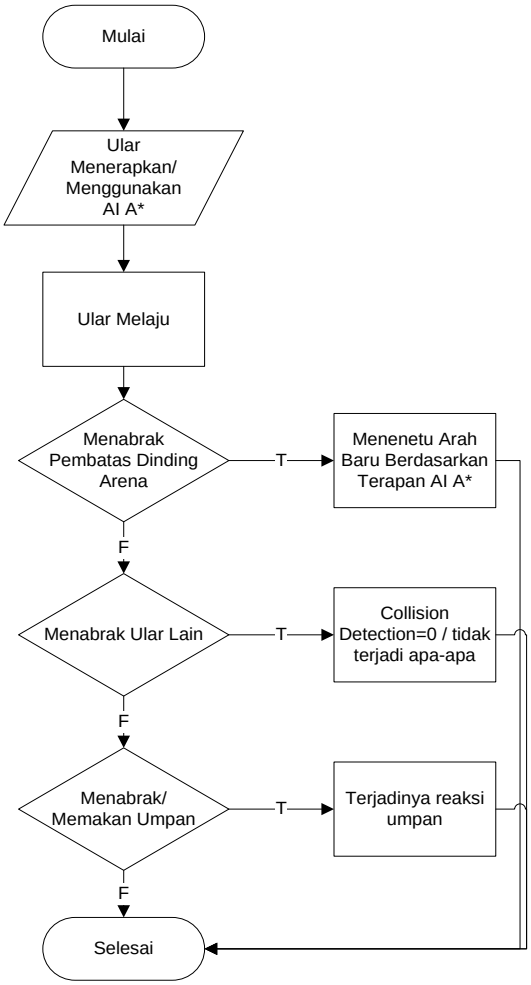
f. Kesimpulan

Dari hasil pengujian alur permainan yang dilakukan, didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

4.2.1.1.4 *Collision Detection*

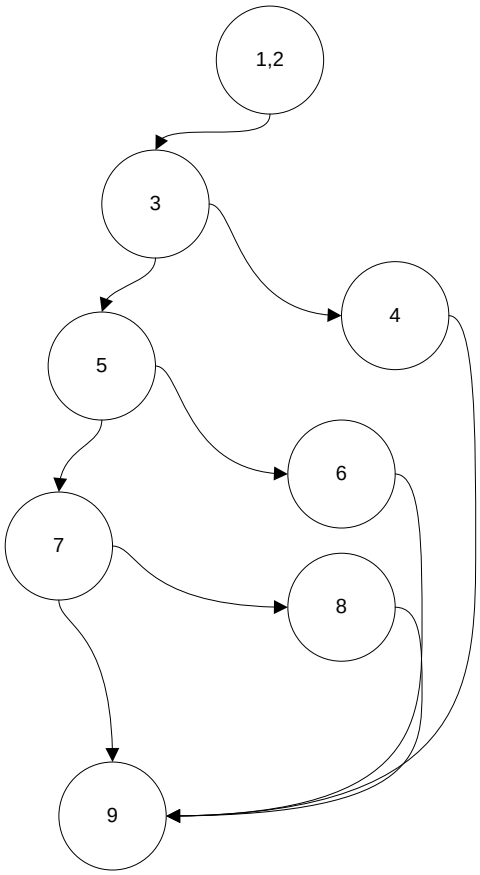
Pengujian *collision detection* dilakukan untuk melihat apakah metode *bounding rectangle collision detection* yang diterapkan sudah berjalan sesuai dengan yang diharapkan. Berikut ini merupakan pengujian *collision detection* yang yang dilakukan terhadap entitas permainan Ular Lawan/Musuh (Ular yang diterapkan AI A* (Star).

a. Flow Chart



Gambar 4.11 *Flow Chart* Pengujian *Collision Detection* Entitas Ular Lawan

b. FIow Graph



Gambar 4.12 *Flow Graph* Pengujian *Collision Detection* Entitas Ular Lawan

c. *Cyclomatic Complexity* V(G)

E: 10, N = 8

V(G) = E-N + 2 = 10 - 8 + 2 = 2+ 2 = 4

d. Independent Paths

Path l = 1,2-3-4-9

Path 2 = 1,2-3-5-6-9

Path 3 = 1,2-3-5-7-8-9

Path 4 = 1,2-3-5-7-9

e. Hasil Uji

Tabel 4.6 Hasil Pengujian *Wite Box Collision Detection* Entitas Ular Lawan

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,2,(void) 3,(true) 4,(void) 9,(void)	Menenetu Arah Baru Berdasarkan Terapan AI A*	Menenetu Arah Baru Berdasarkan Terapan AI A*	*Path* dikerjakan
2	1,2,(void) 3,(false) 5,(true) 6,(void) 9,(void)	Collision Detection=0 / tidak terjadi apa-apa	Collision Detection=0 / tidak terjadi apa-apa	*Path* dikerjakan
3	1,2,(void) 3,(false) 5,(false) 7,(true) 8,(void) 9,(void)	Terjadinya reaksi umpan	Terjadinya reaksi umpan	*Path* dikerjakan
4	1,2,(void) 3,(void) 5,(false) 7,(false) 9,(void)	Collision tidak terdeteksi	Collision tidak terdeteksi	*Path* dikerjakan

f. Kesimpulan

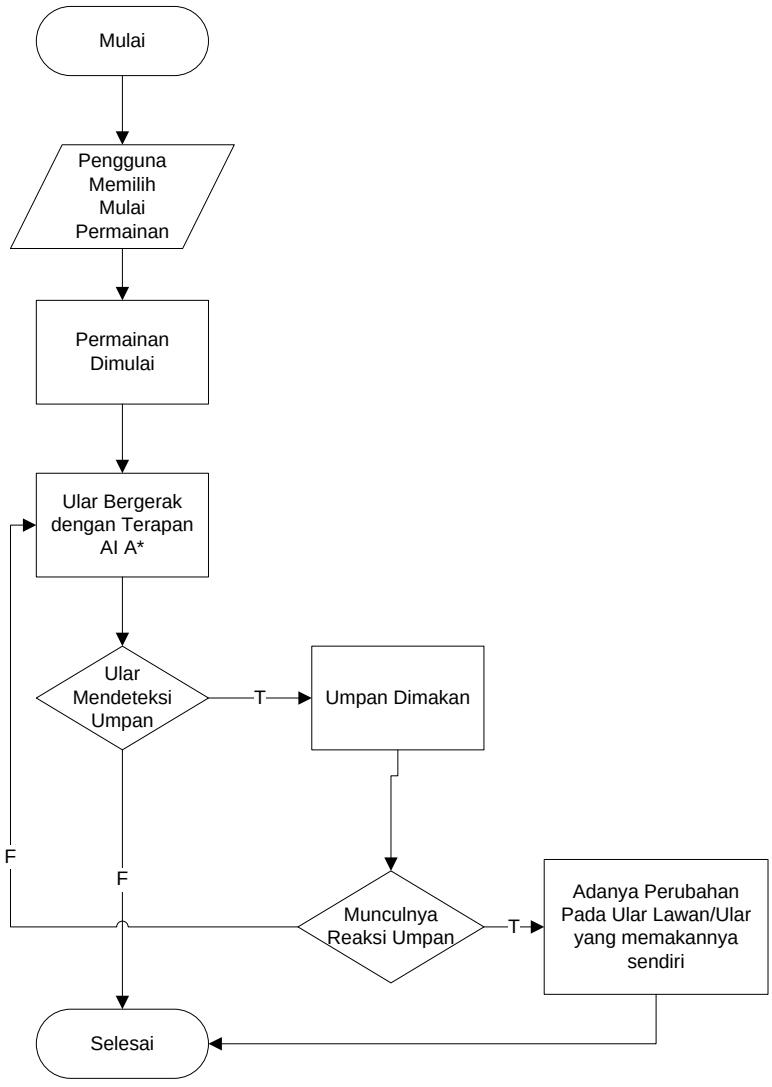
Dari hasil pengujian *collision detection* entitas ular yang dilakukan didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

4.2.1.1.5 Algoritma A* (*Star*)

Pengujian Algoritma A* (Star) dilakukan untuk melihat apakah entitas permainan yang menerapkannya telah berjalan sesuai yang diharapkan tergantung

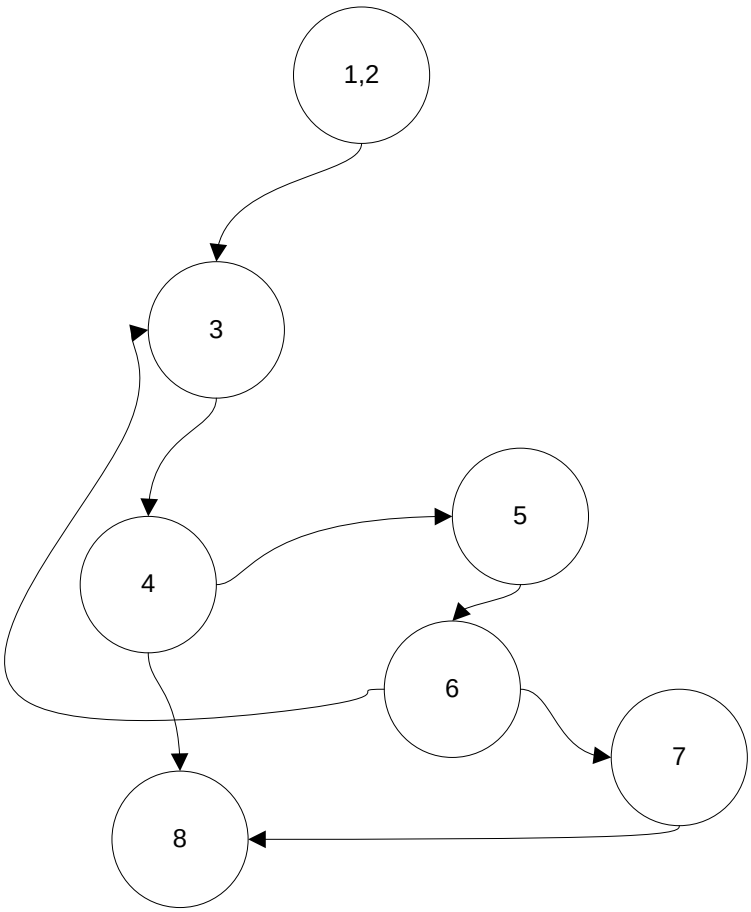
dari kondisi tertentu. Berikut ini merupakan pengujian terhadap metode Algoritma A* (Star) yang diterapkan pada aplikasi Gas Atar dimana di sini AI A* (*star*) diterapkan pada Ular Lawan/Musuh.

a. FIow Chart



Gambar 4.13 *Flow Chart* Pengujian AI A* (star)

b. Flow Graph



Gambar 4.14 *Flow Graph* Pengujian AI A* (star)

c. *Cyclomatic CompIexity* V(G)

E: 8, N = 7

V(G) = E-N + 2 = 8 - 7 + 2 = 1+ 2 = 3

d. Independent Paths

Path l = 1,2-3-4-5-6-7-8

Path 2 = 1,2-3-4-8

Path 3 = 1,2-3-4-5-6-3

e. Hasil Uji

Tabel 4.7 Hasil Pengujian *White Box* AI A* (star)

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,2,(void) 3,(void) 4,(true) 5,(void) 6,true) 7,(void) 8,(void)	Umpan dimakan dan munculnya reaksi umpan	Umpan dimakan dan munculnya reaksi umpan	*Path* dikerjakan
2	1,2,(void) 3,(void) 4,(false) 8,(void)	Umpan tidak dimakan karena telah dimakan ular lain	Umpan tidak dimakan karena telah dimakan ular lain	*Path* dikerjakan
3	1,2,(void) 3,(void) 4,(true) 5,(void) 6,(false) 3,(void)	AI A* telah terselesaikan	AI A* telah terselesaikan	*Path* dikerjakan

f. Kesimpulan

Dari hasil Algoritma A* (Star) terhadap ular yang dilakukan, didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

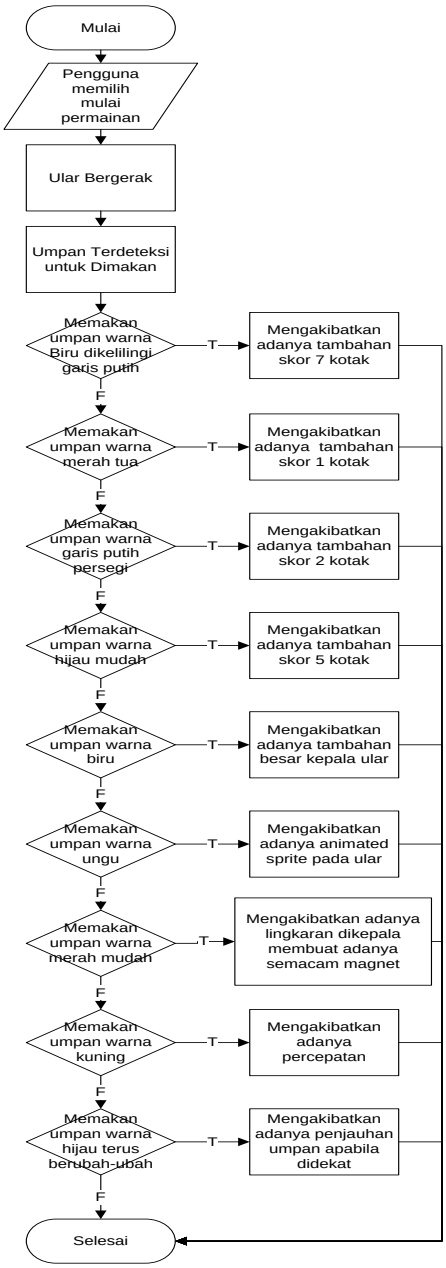
4.2.1.1.6 *Reaction Algorithm*

Pengujian *Reaction Algorithm* dilakukan untuk melihat apakah entitas permainan yang menerapkannya telah berjalan sesuai yang diharapkan tergantung dari kondisi tertentu. Berikut ini merupakan pengujian terhadap metode *Reaction Algorithm* yang diterapkan pada aplikasi Gas Atar:

1. Penyebab terhadap ular yang memakan umpan itu sendiri.

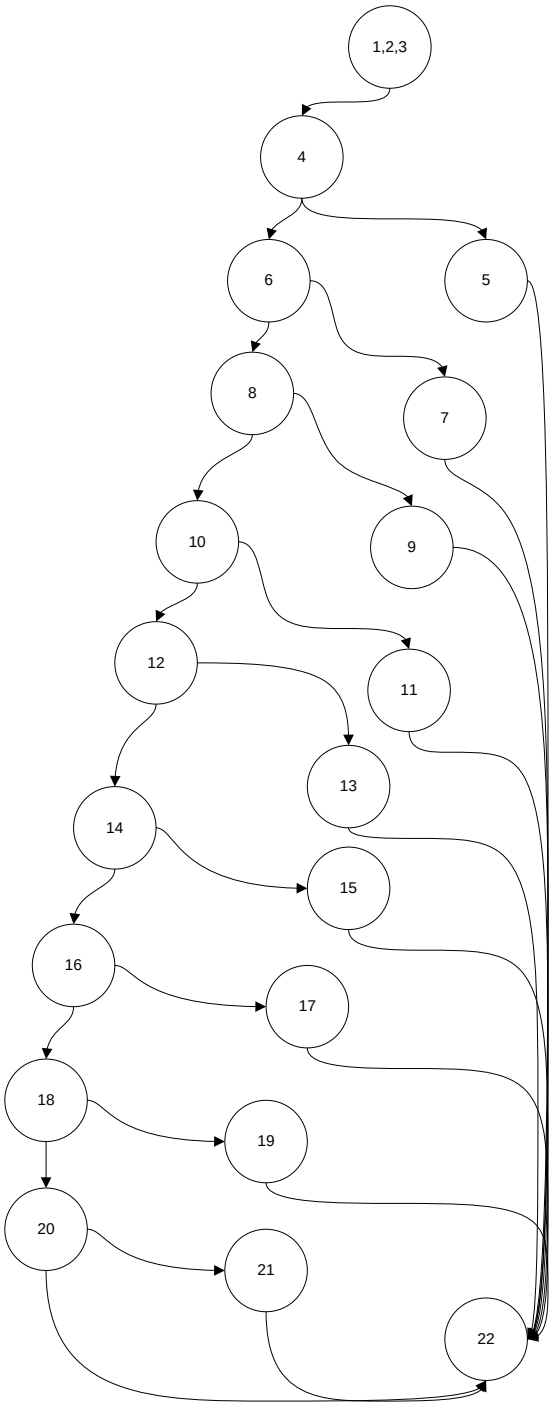
Menguji apakah *state* umpan yang dimakan akan menyebabkan reaksi pada ular itu sendiri terjadi.

a. FIow Chart



Gambar 4.15 *Flow Chart* Pengujian *Reaction Algorithm* pada ular itu sendiri

b. Flow Graph



Gambar 4.16 *Flow Graph* Pengujian *Reaction Algorithm* pada ular itu sendiri

c. *CycIomatic Complexity* V(G)

E: 27, N = 20

V(G) = E-N + 2 = 27 - 20 + 2 = 7+ 2 = 9

d. Independent Paths

Path l = l,2,3-4-5-22

Path 2 = l,2,3-4-6-7-22

Path 3 = l,2,3-4-6-8-9-22

Path 4 = l,2,3-4-6-8-10-11-22

Path 5 = l,2,3-4-6-8-10-12-13-22

Path 6 = l,2,3-4-6-8-10-12-14-15-22

Path 7 = l,2,3-4-6-8-10-12-14-16-17-22

Path 8 = l,2,3-4-6-8-10-12-14-16-18-19-22

Path 9 = l,2,3-4-6-8-10-12-14-16-18-20-21-22

e. Hasil Uji

Tabel 4.8 Hasil Pengujian *White Box Reaction Algorithm* pada ular itu sendiri

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,2,3,(void) 4,(true) 5,(void) 22,(void)	Mengakibatkan adanya tambahan skor 7 kotak	Mengakibatkan adanya tambahan skor 7 kotak	*Path* dikerjakan
2	1,2,3,(void) 4,(false) 6,(true) 7,(void) 22,(void)	Mengakibatkan adanya tambahan skor 1 kotak	Mengakibatkan adanya tambahan skor 1 kotak	*Path* dikerjakan
3	1,2,3,(void) 4,(false) 6,(false)	Mengakibatkan adanya	Mengakibatkan adanya	*Path* dikerjakan

	8,(false) 9,(void) 22,(void)	tambahan skor 2 kotak	tambahan skor 2 kotak	
4	1,2,3,(void) 4,(false) 6,(false) 8,(false) 10,(true) 11,(void) 22,(void)	Mengakibatkan adanya tambahan skor 5 kotak	Mengakibatkan adanya tambahan skor 5 kotak	*Path* dikerjakan
5	1,2,3,(void) 4,(false) 6,(false) 8,(false) 10,(false) 12,(true) 13,(void) 22,(void)	Mengakibatkan adanya tambahan besar kepala ular	Mengakibatkan adanya tambahan besar kepala ular	*Path* dikerjakan
6	1,2,3,(void) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(false) 16,(true) 17,(void) 22,(void)	Mengakibatkan adanya animated sprite pada ular	Mengakibatkan adanya animated sprite pada ular	*Path* dikerjakan
7	1,2,3,(void) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(false) 16,(false) 18,(true) 19,(void) 22,(void)	Mengakibatkan adanya lingkaran dikepala membuat adanya semacam magnet	Mengakibatkan adanya lingkaran dikepala membuat adanya semacam magnet	*Path* dikerjakan
8	1,2,3,(void) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(false)	Mengakibatkan adanya percepatan	Mengakibatkan adanya percepatan	*Path* dikerjakan

	16,(false) 18,(true) 19,(void) 22,(void)			
9	1,2,3,(void) 4,(false) 6,(false) 8,(false) 10,(false) 12,(false) 14,(false) 16,(false) 18,(false) 20,(true) 21,(void) 22,(void)	Mengakibatkan adanya penjauhan umpan apabila didekat	Mengakibatkan adanya penjauhan umpan apabila didekat	*Path* dikerjakan

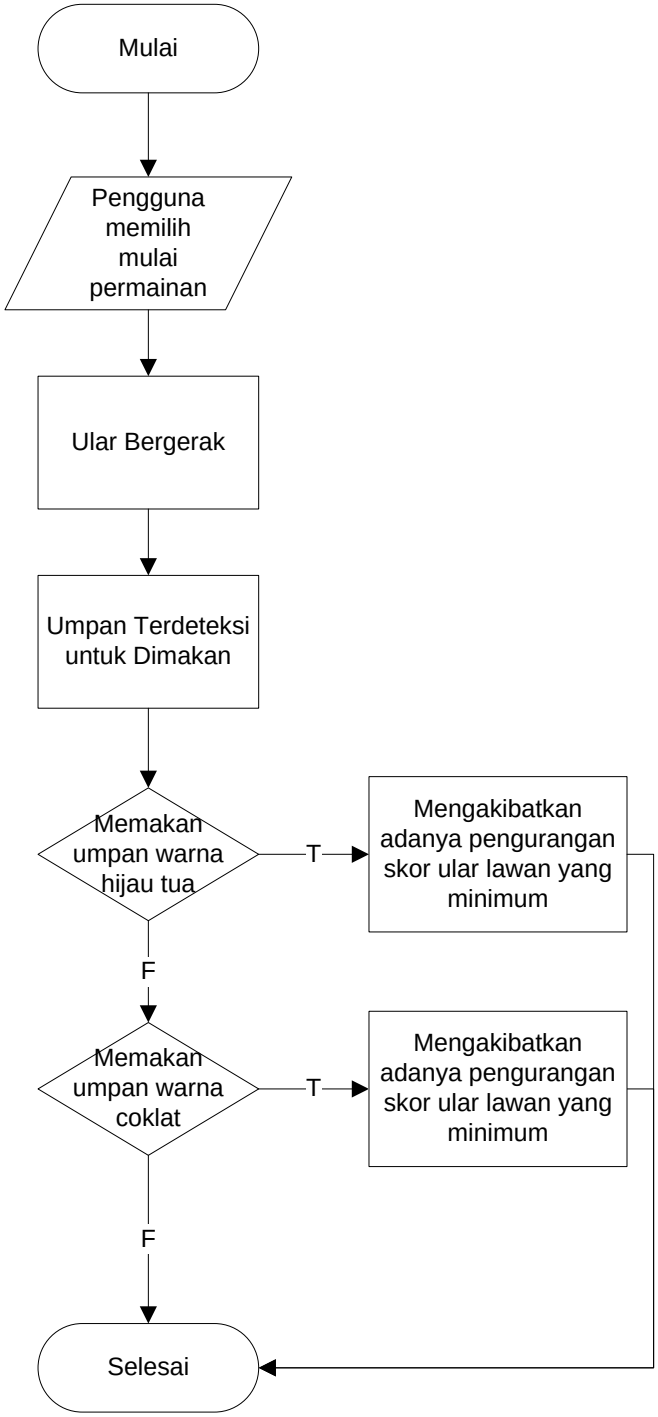
f. Kesimpulan

Dari hasil pengujian *Reaction Algorithm* pada ular itu sendiri yang dilakukan, didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

2. Penyebab terhadap ular lawan yang tidak memakan umpan

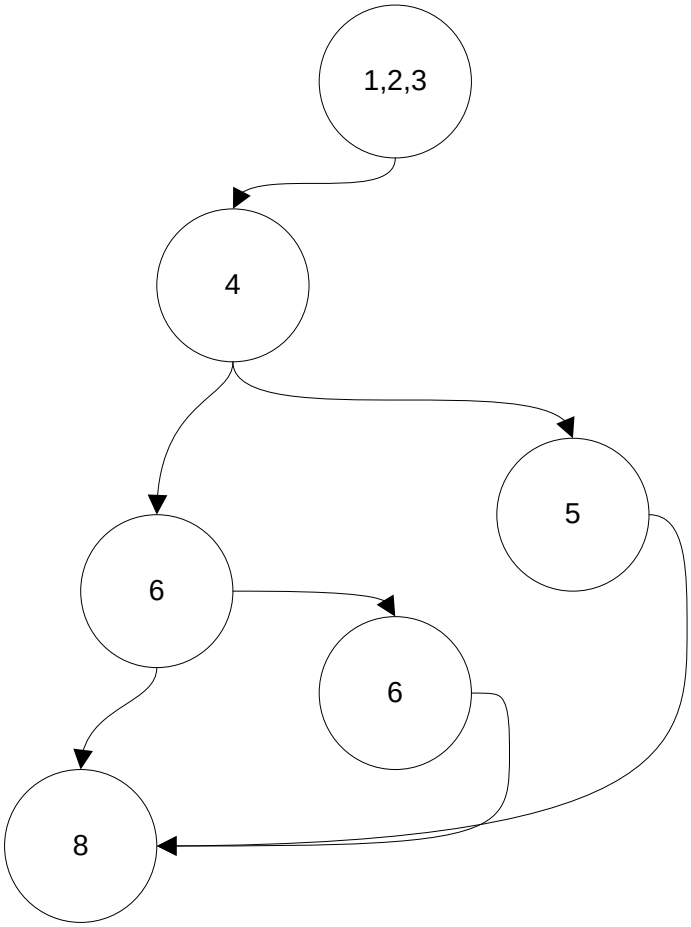
Menguji apakah state umpan yang dimakan akan menyebabkan reaksi pada ular lain yang tidak memakan umpan itu terjadi.

a. FIow Chart



Gambar 4.17 *Flow Chart* Pengujian *Reaction Algorithm* pada ular lain

b. Flow Graph



Gambar 4.18 *Flow Graph* Pengujian *Reaction Algorithm* pada ular lain

c. *CycIomatic Complexity* V(G)

E: 7, N = 6

V(G) = E-N + 2 = 7 - 6 + 2 = 1+ 2 = 3

d. Independent Paths

Path l = 1,2,3-4-5-8

Path 2 = 1,2,3-4-6-7-8

Path 3 = 1,2,3-4-6-8

e. Hasil Uji

Tabel 4.9 Hasil Pengujian *White Box Reaction Algorithm* pada ular lain

Path No.	***Node(n), Input Condition***	***Excpected Result***	***Test Result***	***Conclusion***
1	1,2,3,(void) 4,(true) 5,(void) 8,(void)	Mengakibatkan adanya pengurangan skor ular lawan yang minimum	Mengakibatkan adanya pengurangan skor ular lawan yang minimum	*Path* dikerjakan
2	1,2,3,(void) 4,(false) 6,(true) 7,(void) 8,(void)	Mengakibatkan adanya pengurangan skor ular lawan yang minimum	Mengakibatkan adanya pengurangan skor ular lawan yang minimum	*Path* dikerjakan
3	1,2,3,(void) 4,(false) 6,(false) 8,(void)	Reaction Algorithm terselesaikan	Reaction Algorithm terselesaikan	*Path* dikerjakan

f. Kesimpulan

Dari hasil *Reaction Algorithm* pada ular lain yang dilakukan, didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

4.2.1.1.7 Kesimpulan Pengujian Alpha *White Box*

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa aplikasi permainan Gas Atar sudah berjalan sesuai dengan yang diharapkan. Hal tersebut berdasarkan dari hasil pengujian setiap kelas uji yang dalam hasilnya menyatakan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan,

serta tidak terjadi pengulangan tak berhingga, secara fungsional aplikasi sudah dapat menghasilkan keluaran sesuai yang diharapkan.

4.2.1.2 Pengujian Alpha *Black Box*

Pengujian *black box* adalah pengujian program berdasarkan fungsi dari program. Tujuan dari metode *black box* yaitu untuk menemukan kesalahan fungsi pada program. Pengujian black box ini juga dilakukan untuk melihat bagaimana produk bekerja pada lingkup yang dapat diramalkan. Maka dapat dilakukan pengujian seperti berikut ini:

Tabel 4.10 Rencana Pengujian *Black Box* Aplikasi Gas Atar

Kelas Uji	**Kasus Uji**	**Jenis Pengujian**
Pemilihan Terhadap Entitas Ular yang Diinginkan Menggunakan Metode Bergerak yang Diinginkan	Reaksi Aplikasi terhadap masukkan yang dipilih dari pengguna	*Black Box*

Tabel 4.11 Pengujian Terhadap Pilihan Pengguna Terhadap Entitas Ular

Kasus dan Hasil Uji (Data Benar)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Ular 1 : SimpleAI Ular 2 : SimpleAI Ular 3 : SimpleAI Ular 4 : SimpleAI Ular 5 : SimpleAI Ular 6 : Keyboard Input	Sistem akan berjalan dengan permainan 5 ular AI dan 1 *user* menggunakan *keyboard*	Sistem berjalan dengan permainan 5 ular AI dan 1 *user* menggunakan *keyboard*	[√] Diterima [ ] Ditolak

Ular 1 : SimpleAI Ular 2 : SimpleAI Ular 3 : SimpleAI Ular 4 : SimpleAI Ular 5 : SimpleAI Ular 6 : MouseInput	Sistem akan berjalan dengan permainan 5 ular AI dan 1 *user* menggunakan *Mouse*	Sistem berjalan dengan permainan 5 ular AI dan 1 *user* menggunakan *Mouse*	[√] Diterima [ ] Ditolak
Ular 1 : SimpleAI Ular 2 : SimpleAI Ular 3 : SimpleAI Ular 4 : SimpleAI Ular 5 : KeyboardInput Ular 6 : MouseInput	Sistem akan berjalan dengan permainan 4 ular AI dan 1 *user* menggunakan *Mouse* dan 1 *user* menggunakan *Keyboard*	Sistem berjalan dengan permainan 4 ular AI dan 1 *user* menggunakan *Mouse* dan 1 *user* menggunakan *Keyboard*	[√] Diterima [ ] Ditolak
Kasus dan Hasil Uji (Data Salah)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Memilih *KeyboardInput* lebih dari satu	Sistem akan berjalan menggunakan *Keyboard* terhadap entitas ular pilihan terakhir	Sistem berjalan menggunakan *Keyboard* terhadap entitas ular pilihan terakhir	[√] Diterima [ ] Ditolak
Memilih *MouseInput* lebih dari satu	Sistem akan berjalan menggunakan *Mouse* terhadap entitas ular pilihan terakhir	Sistem berjalan menggunakan *Mouse* terhadap entitas ular pilihan terakhir	[√] Diterima [ ] Ditolak
Memilih *MouseInput* dan *KeyboardInput* lebih dari satu	Sistem akan berjalan menggunakan *Mouse* dan *Keyboard* terhadap entitas ular pilihan terakhir	Sistem berjalan menggunakan *Mouse* dan *Keyboard* terhadap entitas ular pilihan terakhir	[√] Diterima [ ] Ditolak

4.2.2 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif, dimana dilakukan pengujian secara langsung terhadap pengguna dengan menggunakan kuisioner mengenai kepuasan pengguna atas aplikasi yang telah dibangun. Adapun metode penilaian pengujian yang digunakan adalah.metode kuantitatif berdasarkan data dari pengguna. Pengujian dilakukan pada 30 responden dengan rentang usia 6 - 41 tahun dan setiap responden diberikan 5 pertanyaan.

4.2.2.1 Pertanyaan Pengguna

Adapun pertanyaan yang diajukan dalam kuesioner untuk pengguna yang menguji aplikasi Gas Atar adalah sebagai berikut:

1. Apakah tampilan aplikasi permainan ini secara umum sudah baik?
    a. Sangat Baik
    b. Baik
    c. Kurang Baik
    d. Tidak Baik
2. Apakah aplikasi permainan ini secara umum mudah dioperasikan ?
    a. Sangat Mudah
    b. Mudah
    c. Kurang Mudah
    d. Tidak Mudah

3. Apakah alur permainannya seru?
    a. Sangat Seru
    b. Seru
    c. Kurang Baik
    d. Tidak Baik
4. Apakah umpannya sudah banyak dan bervariasi?
    a. Sangat Banyak dan bervariasi
    b. Banyak dan bervariasi
    c. Kurang Banyak dan kurang bervariasi
    d. Tidak Banyak dan tidak bervariasi
5. Apakah Anda menyukai aplikasi permainan ini?
    a. Sangat Suka
    b. Suka
    c. Kurang Suka
    d. Tidak Suka

4.2.2.2 Hasil Pengujian Beta

Berdasarkan data hasil kuisioner tersebut dicari presentase masing-masing jawaban dengan menggunakan rumus R=P/Q*100%, dimana :

P = Banyaknya jawaban responden tiap soal

Q = Jumlah responden

R = Nilai prosentase

4.2.2.3 Hasil Kuesioner

Berikut ini merupakan data yang didapat dari hasil kuesioner :

1. Apakah tampilan aplikasi permainan ini secara umum sudah baik?

Tabel 4.12 Hasil Kuesioner Tampilan Aplikasi

No	Jawaban	Responden	Prosentase (%)
1	Sangat Baik	18	60
2	Baik	12	40
3	Kurang Baik	0	0
4	Tidak Baik	0	0

2. Apakah alur permainannya seru?

Tabel 4.14 Hasil Kuesioner Alur Permainan

No	Jawaban	Responden	Prosentase (%)
1	Sangat Seru	13	43,33
2	Seru	16	53,33
3	Kurang Baik	1	3,33
4	Tidak Baik	0	0

3. Apakah karakter ularnya mudah dikendalikan ?

Tabel 4.13 Hasil Kuesioner Pengunaan Aplikasi seperti Navigasi Ular

No	Jawaban	Responden	Prosentase (%)
1	Sangat Mudah	8	26,67
2	Mudah	22	73,33
3	Kurang Mudah	0	0
4	Tidak Mudah	0	0

4. Apakah umpannya sudah banyak dan bervariasi?

Tabel 4.16 Hasil Kuesioner umpan

No	Jawaban	Responden	Prosentase (%)
1	Sangat Banyak dan sangat bervariasi	5	16,67
2	Banyak dan bervariasi	23	76,67
3	Kurang Banyak dan kurang bervariasi	2	6,67
4	Tidak Banyak dan tidak bervariasi	0	0

1. Apakah Anda menyukai aplikasi permainan ini?

Tabel 4.17 Hasil Kuesioner Tingkat Ketertarikan

No	Jawaban	Responden	Prosentase (%)
1	Sangat Suka	6	20
2	Suka	24	80
3	Kurang Suka	0	0
4	Tidak Suka	0	0

4.2.2.4 Kesimpulan Pengujian Beta

Berdasarkan Hasil dari pengujian beta dapat disimpulkan bahwa aplikasi yang telah dibangun dapat menumbuhkan kembali dan meningkatkan tingkat ketertarikan pengguna aplikasi permainan terhadap permainan bergenre *game snake*.