BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Implementasi merupakan terjemahan perancangan yang berdasarkan hasil analisis pada bab sebelumnya ke dalam bahasa pemograman yang dapat dimengerti oleh komputer.

4.1.1 Perangkat Lunak Implementasi dan Pengujian

Perangkat lunak yang digunakan untuk mengimplementasikan dan menguji aplikasi game simulasi adalah sebagai berikut :

1. Windows 7 Ultimate sebagai sistem operasi
2. Greenfoot sebagai perangkat lunak pembangun

4.1.2 Perangkat Keras Implementasi dan Pengujian

Perangkat keras yang digunakan untuk mengimplementasikan dan menguji aplikasi game simulasi adalah sebagai berikut :

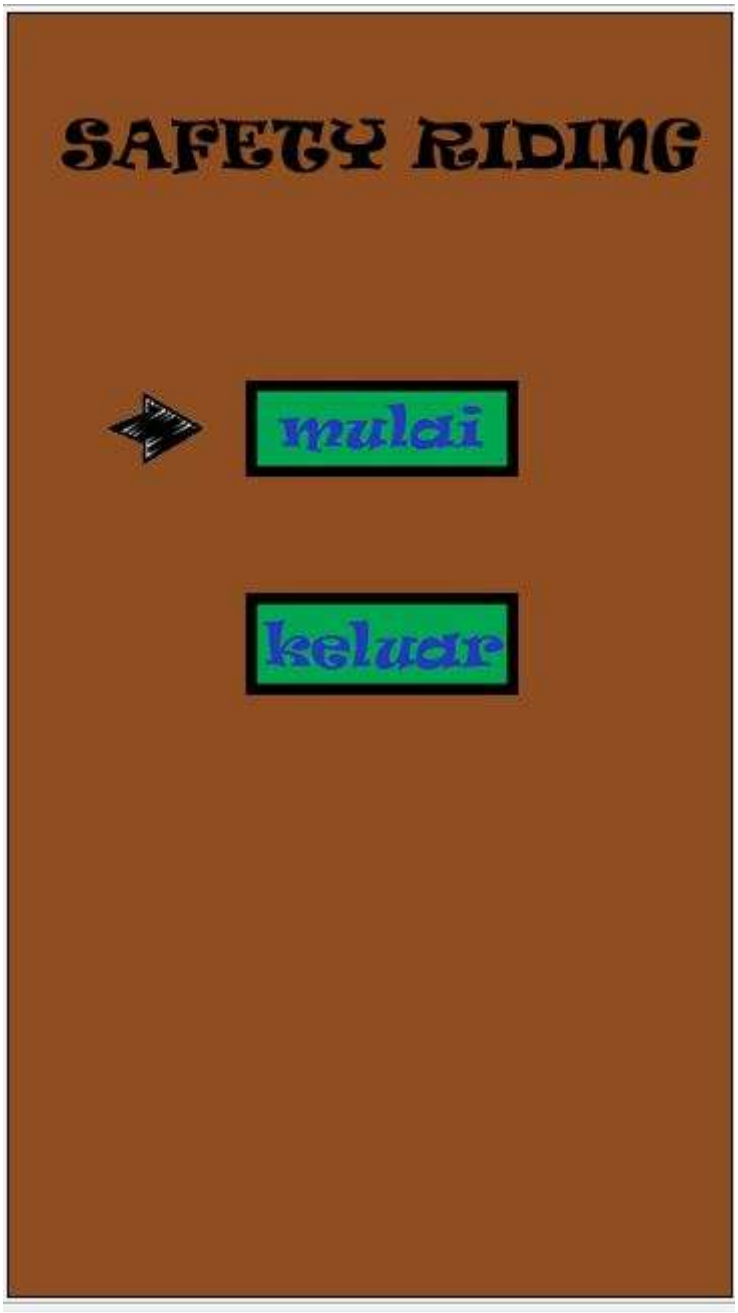
1. Prosesor Intel Core i3-2350M 2.3GHz
2. RAM 2GB
3. Harddisk 250GB
4. Monitor 14”
5. Keyboard dan Mouse

4.1.3 Implmentasi Antarmuka

Pada bagian ini akan ditampilkan bentuk dari implementasi antarmuka menu utama, menu mulai (pilih level), menu main.

4.1.3.1 Implementasi Antarmuka Menu utama

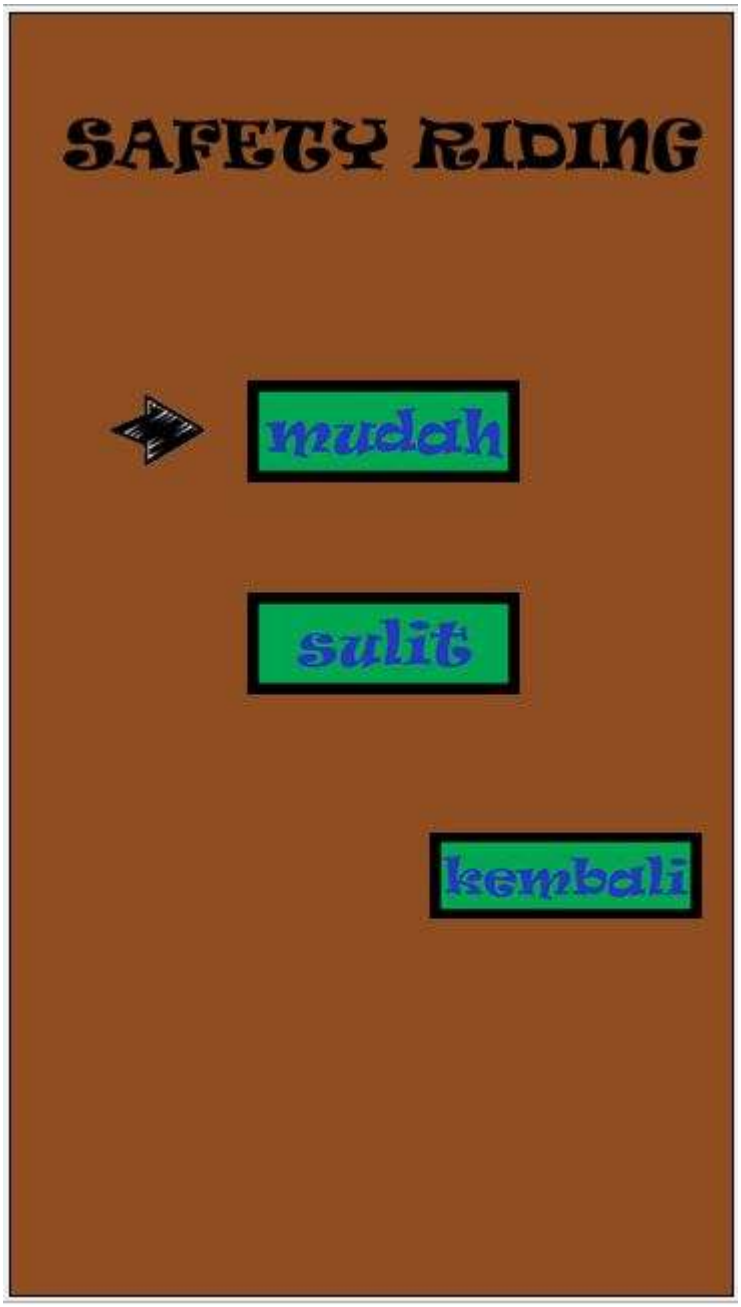
Berikut adalah Implementasi Antarmuka Menu Utama yang ditunjukkan oleh Gambar 4.1.



Gambar 4.1 Antarmuka Menu Utama

4.1.3.2 Implementasi Antarmuka Menu Pilih Level

Berikut adalah Implementasi Antarmuka Menu Pilih Level yang ditunjukkan oleh gambar berikut yang dimana terdapat antarmuka level mudah, antarmuka level sulit. Tampilan antarmuka pilih level dapat dilihat pada Gambar 4.2.



Gambar 4.2 Antarmuka Menu Pilih Level

1. Antarmuka arena permainan mudah

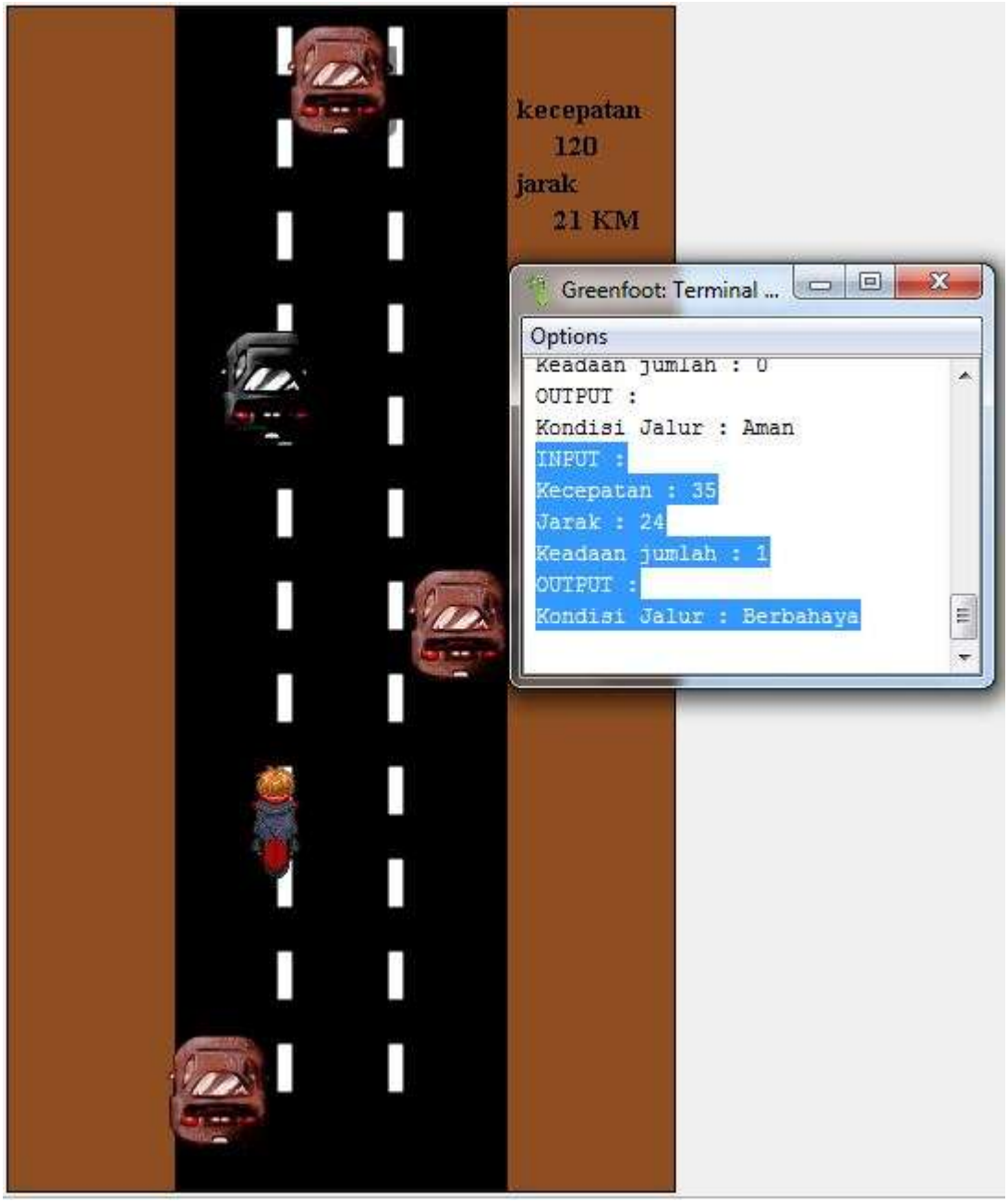
   Antarmuka arena permainan mudah merupakan tampilan yang NPC-nya tidak menggunakan *fuzzy*. Tampilan antarmuka arena permainan mudah dapat dilihat pada Gambar 4.3.



Gambar 4.3 Antarmuka Arena Permainan Mudah

2. Antarmuaka arena permainan sulit

   Antarmuka arena permainan sulit merupakan tampilan yang NPC-nya menggunakan *fuzzy*. Tampilan antarmuka arena permainan sulit dapat dilihat pada gambar 4.4.



Gambar 4.4 Antarmuka Arena Permainan Sulit

4.2 Pengujian Sistem

Pengujian sistem merupakan hal terpenting yang dilakukan untuk menemukan kekurangan atau kesalahan pada aplikasi yang diuji. Pengujian bermaksud untuk mengetahui implementasi algoritma logika *fuzzy* yang dibuat telah memenuhi kriteria yang sesuai dengan tujuan perancangan. Adapun metode pengujian yang digunakan dalam penelitian ini adalah pengujian *alpha*.

Pengujian alpha adalah pengujian yang berfokus pada persyaratan fungsional aplikasi. Pengujian dilakukan terhadap aplikasi *game* simulasi yang sebelumnya telah dirancang dan diterapkan algoritma *fuzzy*. Untuk memastikan bahwa perancangan dan penerapan algoritma *fuzzy* dapat berjalan dengan benar sesuai kebutuhan dan tujuan yang diharapkan.

4.2.1 Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game* Simulasi. Skenario pengujian yang akan dilakukan pada aplikasi ini dapat dilihat pada Tabel 4.1.

Tabel 4.1 Skenario Pengujian Aplikasi

No.	Komponen yang diuji	Skenario pengujian	Jenis pengujian
1.	Menu Utama	Memilih Menu Mulai Baru	*Black Box*
		Memilih Menu Keluar	*Black Box*
2.	Menu Memilih Level	Memilih Level Mudah	*Black Box*
		Memilih Level Sulit	*Black Box*
3.	Permainan Level Mudah	Menekan Tombol A (ke kiri) dan D (ke kanan)	*Black Box*
4.	Permainan Level Sulit	Menekan Tombol A (ke kiri) dan D (ke kanan)	*Black Box*
5.	Algoritma Logika *Fuzzy*	Algoritma Logika Fuzzy pada NPC mobil rintangan	*White Box*

4.2.2 Pengujian *Black Box*

Black box bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya. Berdasarkan pada tabel 4.1 skenario pengujian aplikasi, berikut ini adalah hasil pengujian perangkat lunak yang sudah diimplementasikan menggunakan metode *Black Box*, Pengujian metode *Black Box* dapat dilihat pada tabel 4.2.

Tabel 4.2 Hasil Pengujian *Black Box*

No.	Komponen yang diuji	Skenario pengujian	Hasil Yang Diharapkan	Hasil Pengujian
1.	Menu Utama	Memilih Menu Mulai Baru	Memilih Menu Level	[√] Berhasil Tidak Berhasil
		Memilih Menu Keluar	Keluar dari permainan	[√] Berhasil Tidak Berhasil
2.	Menu Memilih Level	Memilih Level Mudah	Masuk ke Permainan Level Mudah	[√] Berhasil Tidak Berhasil
		Memilih Level Sulit	Masuk ke Permainan Level Sulit	[√] Berhasil Tidak Berhasil
3.	Permainan Level Mudah	Menekan Tombol A (ke kiri) dan D (ke kanan)	Pergerakan Player	[√] Berhasil Tidak Berhasil
4.	Permainan Level Sulit	Menekan Tombol A (ke kiri) dan D (ke kanan)	Pergerakan Player	[√] Berhasil Tidak Berhasil

4.2.2.1 Kasus Dan Hasil Pengujian (*White Box Testing)*

White Box adalah teknik meramalkan cara kerja perangkat lunak secara rinci, karenanya *logical path* (jalur logika) perangkat lunak akan di *test* dengan menyediakan *test case* yang akan mengerjakan kumpulan kondisi atau pengulangan secara spesifik.

Berikut ini adalah proses pengujian logika *fuzzy* untuk menentukan keputusan mendahului mobil yang di depannya atau tetap berada di jalur yang di laluinya pada NPC mobil rintangan.

Tabel 4.3 Algoritma Logika Fuzzy Pada NPC

1.	`private int JALUR1 = 300;`
	`private int JALUR2 = 500;`
	`private int JALUR3 = 700;`
	`private int FKECEPATAN;`
	`private int FJARAK;`
	`private int FKEADAAN;`
	`private int DEFUZZY;`
	`private boolean PINDAH = false;`
	`int speed=1;`

```
int move = 0;
int posxawal, posxakhir;
ArrayList<player> pl = new ArrayList<player>();
ArrayList<npc> npc = new ArrayList<npc>();

public void Defuzzy(){
2.        if(fuzzyKecepatan()==1 && fuzzyJarak()==1 &&
      fuzzyKeadaan()==1){
            DEFUZZY = 1;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==1 &&
      fuzzyKeadaan()==2){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==1 &&
      fuzzyKeadaan()==3){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==2 &&
      fuzzyKeadaan()==1){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==2 &&
      fuzzyKeadaan()==2){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==2 &&
      fuzzyKeadaan()==3){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==3 &&
      fuzzyKeadaan()==1){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==3 &&
      fuzzyKeadaan()==2){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==1 && fuzzyJarak()==3 &&
      fuzzyKeadaan()==3){
            DEFUZZY = 3;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==1 &&
      fuzzyKeadaan()==1){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==1 &&
      fuzzyKeadaan()==2){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==1 &&
      fuzzyKeadaan()==3){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==2 &&
      fuzzyKeadaan()==1){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==2 &&
      fuzzyKeadaan()==2){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==2 &&
      fuzzyKeadaan()==3){
            DEFUZZY = 3;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==3 &&
      fuzzyKeadaan()==1){
            DEFUZZY = 2;
        }else if(fuzzyKecepatan()==2 && fuzzyJarak()==3 &&
      fuzzyKeadaan()==2){
```

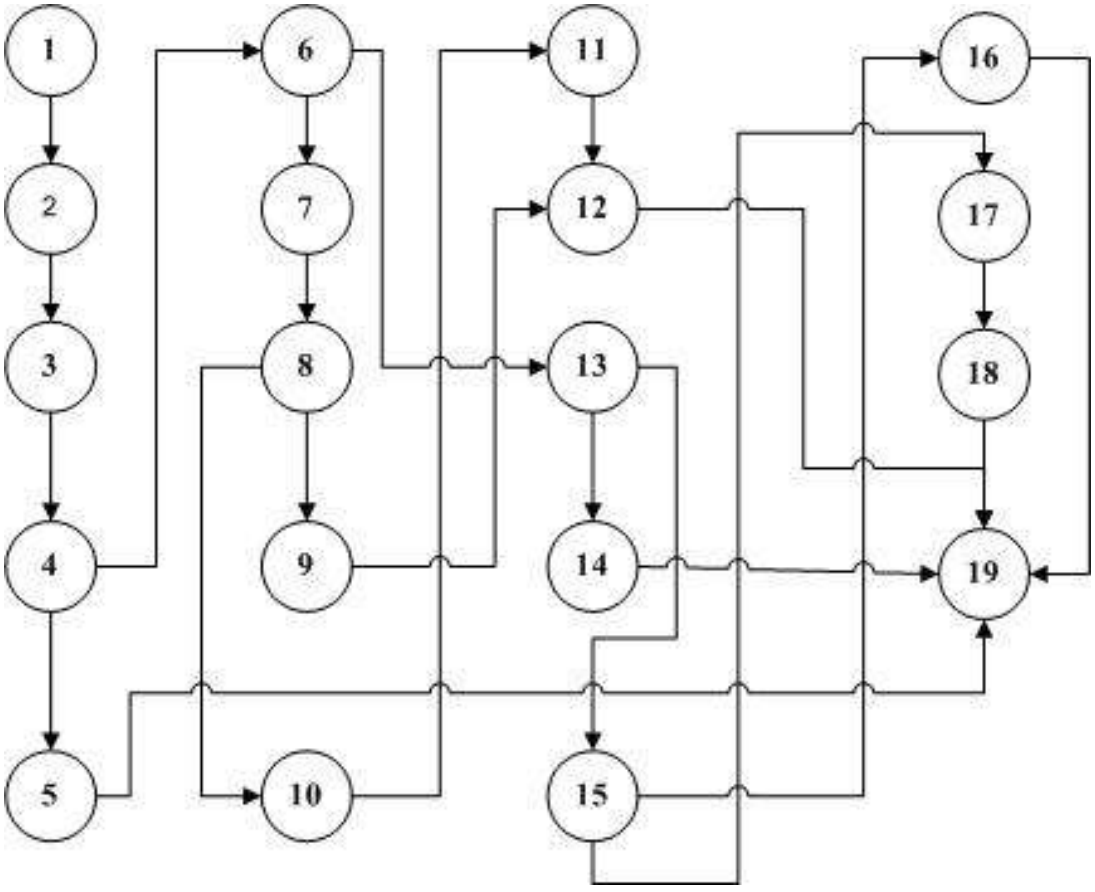
```
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==2 && fuzzyJarak()==3 &&
        fuzzyKeadaan()==3){
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==1 &&
        fuzzyKeadaan()==1){
            DEFUZZY = 2;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==1 &&
        fuzzyKeadaan()==2){
            DEFUZZY = 2;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==1 &&
        fuzzyKeadaan()==3){
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==2 &&
        fuzzyKeadaan()==1){
            DEFUZZY = 2;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==2 &&
        fuzzyKeadaan()==2){
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==2 &&
        fuzzyKeadaan()==3){
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==3 &&
        fuzzyKeadaan()==1){
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==3 &&
        fuzzyKeadaan()==2){
            DEFUZZY = 3;
          }else if(fuzzyKecepatan()==3 && fuzzyJarak()==3 &&
        fuzzyKeadaan()==3){
            DEFUZZY = 3;
          }

3.        if(DEFUZZY>2){
4.            if(getX()>=200 && getX()<400){
5.                move=1;
                  PINDAH = true;
6.            }else if(getX()>=400 && getX()<600 && !PINDAH){
7.                int ran = Greenfoot.getRandomNumber(1);
8.                if(ran==0){
9.                    move=1;
10.               }else{
11.                   move=-1;
                  }
12.               PINDAH = true;
13.           }else if(getX()>=400 && getX()<500 && PINDAH){
14.               move=-1;
                  PINDAH = true;
15.           }else if(getX()>=500 && getX()<600 && PINDAH){
16.               move=1;
                  PINDAH = true;
17.           }else if(getX()>=600 && getX()<800){
18.               move=-1;
                  PINDAH = true;
              }
19.           FKECEPATAN = 1;
```

```
                FJARAK = 1;
                FKEADAAN = 1;
            }
        }
```

4.2.2.2 *Flow Graph*

Berikut ini merupakan *flow graph* logika *fuzzy* berdasarkan pada Gambar 4.5.



Gambar 4.5 *Flow Graph* Logika *Fuzzy* Pada NPC

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

4.2.2.3 *Cyclomatic Complexity V(G)*

Dari *flow graph* di atas, dapat dihitung *cyclomatic complexity* yang di dapat dari perhitungan dibawah ini

V(G) = E – N + 2

Di mana :

E adalah edge = Jumlah Aksi, dan

N adalah node = Jumlah Kondisi. Jadi :

V(G) = 22 edge – 19 node + 2 = 5

4.2.2.4 *Independent Path*

Dari hasil perhitungan *cyclomatic complexity* terdapat 5 *independent path* yaitu :

Path 1 = 1-2-3-4-5-19

Path 2 = 1-2-3-4-6-13-14-19

Path 3 = 1-2-3-4-6-7-8-9-12-19

Path 4 = 1-2-3-4-6-7-8-10-11-12-19

Path 5 = 1-2-3-4-6-13-15-17-18-19

4.2.2.5 *Graph Matric*

Penomoran ulang *flowgraph* atau penyederhanaan berdasarkan kondisi atau simpul dapat dilihat pada tabel

Tabel 4.4 *Graph Matric* Logika***Fuzzy***

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	N(E)-1
1		1																		1-1 = 0
2			1																	1-1 = 1
3				1																1-1 = 0
4				1	1															2-1 = 1
5																			1	1-1 = 0
6							1						1							2-1 = 1
7								1												1-1 = 0
8									1	1										2-1 = 1
9												1								1-1 = 0
10											1									1-1 = 0
11												1								1-1 = 0
12																			1	1-1 = 0
13														1	1					2-1 = 1
14																			1	1-1 = 0
15																1	1			2-1 = 1
16																			1	1-1 = 0
17																		1		1-1 = 0
18																			1	1-1 = 0
19																				0-1 =-1
																	Sum(E)		+1	4+1 = 5

Keterangan :

1. Baris dan kolom merepresentasikan simpul (*node*).
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul.

4.2.2.6 Hasil Pengujian Logika *Fuzzy*

Berikut ini adalah hasil dari pengujian *whitebox* penelusuran logika *Fuzzy* perhitungan keputusan pada NPC mobil rintangan.

Tabel 4.5 Hasil Pengujian Logika *Fuzzy*

No	Path	Yang Diharapkan	Hasil Yang Dicapai	Kesimpulan
1.	1-2-3-4-5-19	Melakukan proses *fuzzy*fikasi apabila NPC mendekati	NPC mobil rintangan berpindah jalur	[√] Berhasil [ ] Tidak Berhasil

		mobil rintangan di depannya	dengan mendahului mobil yang di depannya	
2.	1-2-3-4-6-13-14-19	Melakukan proses *fuzzy*fikasi apabila NPC mendekati mobil rintangan didepannya	NPC mobil rintangan berpindah jalur dengan mendahului mobil yang di depannya	[√] Berhasil [ ] Tidak Berhasil
3.	1-2-3-4-6-7-8-9-12-19	Melakukan proses *fuzzy*fikasi apabila NPC mendekati mobil rintangan didepannya	NPC mobil rintangan berpindah jalur dengan mendahului mobil yang di depannya	[√] Berhasil [ ] Tidak Berhasil
4.	1-2-3-4-6-7-8-10-11-12-19	Mendapatkan nilai keputusan dari proses *fuzzy*fikasi sesuai dengan input berupa kecepatan, jarak dan keadaan dari NPC mobil rintangan	Nilai keputusan didapat berdasarkan input berupa kecepatan, jarak dan keadaan dari NPC mobil rintangan	[√] Berhasil [ ] Tidak Berhasil
5.	1-2-3-4-6-13-15-17-18-19	Mendapatkan nilai keputusan dari proses *fuzzy*fikasi sesuai dengan input berupa kecepatan, jarak dan keadaan dari NPC mobil rintangan	Nilai keputusan didapat berdasarkan input berupa kecepatan, jarak dan keadaan dari NPC mobil rintangan	[√] Berhasil [ ] Tidak Berhasil