BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap kelanjutan dari kegiatan perancangan algoritma. Wujud dari hasil implementasi ini adalah sebuah aplikasi *game Tower Deffense* dengan *file extension* .swf.

4.1.1 Implementasi Perangkat Keras

Berikut adalah spesifikasi perangkat keras (*hardware*) yang digunakan untuk pengujian algoritma *fuzzy*. Untuk lebih jelasnya dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Perangkat Keras Yang Digunakan

Nama Perangkat	Spesifikasi
Processor	CPU 1.80 Ghz
Memory	1 GB
HDD	100 MB Free Hardisk
Display	Resolusi 1028 x 768
Monitor	1 Unit
Mouse	1 Unit
Keyboard	1 Unit

4.1.2 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak (Software) yang digunakan untuk pengujian algoritma *fuzzy* dapat dilihat pada Tabel 4.2.

Tabel 4. 2 Perangkat Lunak Yang Digunakan

Nama Perangkat Lunak	Jenis
Windows 7/8	Sistem Operasi
Flash Player 9/10/11	Player untuk menjalankan file .swf
Adobe Flash CS 6 / CS 5.5	Sebagai program pembangun

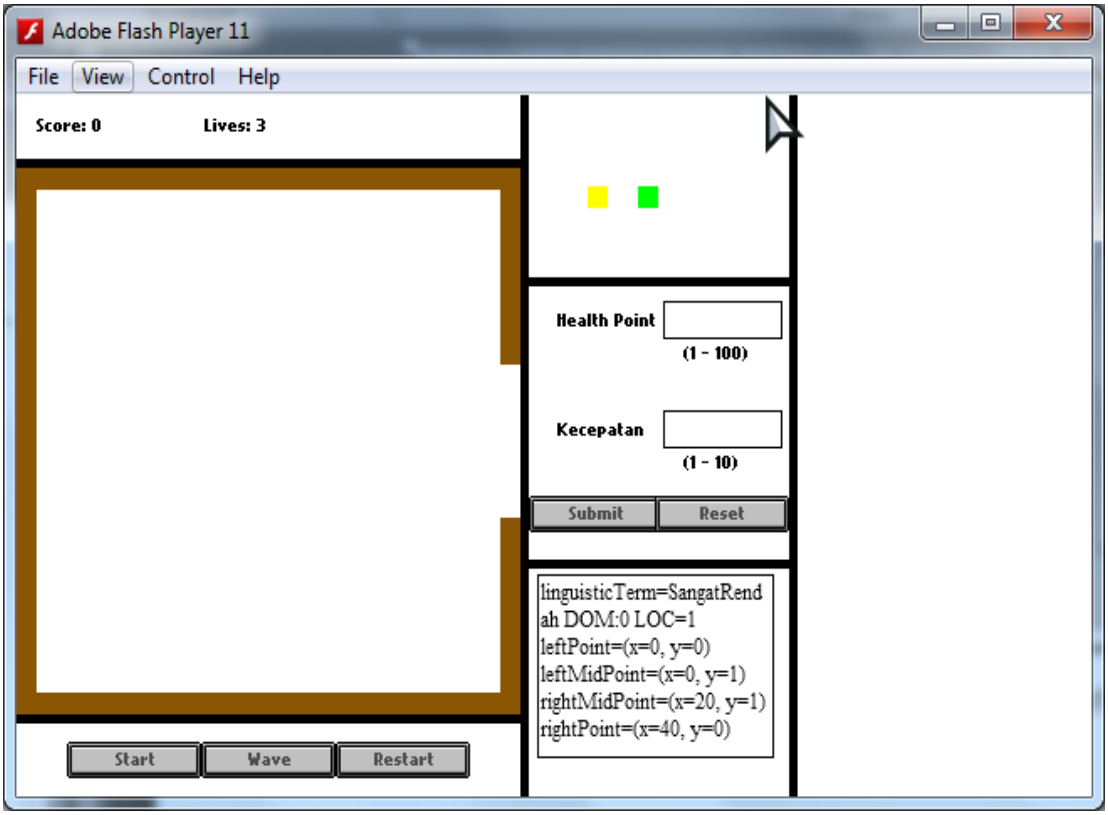
4.1.3 Implementasi Instalasi

Berikut ini adalah beberapa proses sebelum masuk ke-tahapan pengujian.

1. *Copy* file *FuzzyTowerDeffense*.swf pada direktori komputer.
2. *Instal flash player* untuk menjalankan file .swf.
3. Buka file *FuzzyTowerDeffense*.swf.

4.1.4 Implementasi Antar Muka

Implementasi antar muka merupakan tahap dimana sebuah sistem siap dioperasikan pada tahap yang sebenarnya, sehingga bisa diketahui apakah perancangan sistem yang telah dibuat sudah sesuai dengan apa yang sudah dirancang sebelumnya. Berikut ini adalah tampilan menu utama *FuzzyTowerDeffense*.



Gambar 4. 1 Tampilan Menu Utama *Fuzzy Tower Deffense*

4.2 Pengujian Sistem

Pengujian sistem dilakukan bertujuan untuk menemukan kesalahan atau kekurangan pada penerapan algoritma yang diuji. Pengujian bermaksud untuk mengetahui implementasi algoritma *fuzzy* yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan pengembangan pada aplikasi *game Tower Deffense*. Pengujian yang akan digunakan pada penelitian ini adalah dengan menggunakan pengujian *alpha*.

Pengujian alpha adalah pengujian yang berfokus pada persyaratan fungsional aplikasi atau perangkat lunak. Pengujian dilakukan terhadap aplikasi *game Tower Defense* 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 Tower Defense*. Skenario pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada Tabel 4.3.

Tabel 4. 3 Skenario Pengujian Aplikasi

No	Komponen Pengujian	Skenario Pengujian	Jenis Pengujian
1	Mulai	Pilih Tombol *Start*	*Black Box*
		Pilih Tombol *Restart*	*Black Box*
2	Mengeluarkan Musuh	Pilih Tombol *Reset*	*Black Box*
		Pilih Tombol *Submit*	*Black Box*
		Pilih Tombol *Wave*	*Black Box*
3	Menghentikan Pergerakan Musuh	Menempatkan Tower	*Black Box*
4	Logika *Fuzzy*	Logika *Fuzzy* Pada NPC *Tower*	*White Box*

4.2.2 Kasus Dan Hasil Pengujian (*Black Box Testing*)

Black box bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang

diharapkan. Berikut ini merupakan kasus untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *black box testing* berdasarkan skenario pengujian aplikasi yang terdapat pada Tabel 4.3 yang sudah dijabarkan. Hasil pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada Tabel 4.4.

Tabel 4. 4 Pengujian *Black Box*

No	Kasus yang akan diuji	Skenario dan hasil uji		
		Skenario uji	**Hasil yang diharapkan**	**Kesimpulan**
1	Mulai	Pilih Tombol *Start*	Sistem mengatur jalur musuh	[√] Berhasil [ ] Tidak Berhasil
			Sistem menampilkan pergerakan musuh	[√] Berhasil [ ] Tidak Berhasil
		Pilih Tombol *Restart*	Sistem mengulang permainan ke kondisi awal	[√] Berhasil [ ] Tidak Berhasil
2	Mengeluarkan Musuh	Pilih Tombol *Reset*	Sistem menghapus nilai input *health point* dan kecepatan di *textfield*	[√] Berhasil [ ] Tidak Berhasil
		Pilih Tombol *Submit*	Sistem men-set kecepatan dan *healthpoint* unit musuh	[√] Berhasil [ ] Tidak Berhasil
		Pilih Tombol *Wave*	Sistem menampilkan musuh di arena permainan	[√] Berhasil [ ] Tidak Berhasil
3	Menghentikan Pergerakan Musuh	Menempatkan *Tower* di Arena Permainan	Sistem menampilkan tower di arena permainan	[√] Berhasil [ ] Tidak Berhasil

4.2.3 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 target tembak pada NPC *Tower*.

Tabel 4. 5 *Psedocode* Logika *Fuzzy*

No	Script
1	getTargetPriority();
2	if (jarakMusuh < minRange || jarakMusuh > maxRange);
3	return 0;

4	//Deklarasi input variabel fuzzy jarak
	jarakMusuhStatusInput = new FuzzyInput();

5	//Deklarasi input variabel fuzzy kecepatan
	kecepatanMusuhStatusInput = new FuzzyInput();

6	//Deklarasi input variabel fuzzy HP
	hpMusuhStatusInput = new FuzzyInput();

7	//Deklarasi variabel fuzzy jarak
	Manifold = new FuzzyManifold (“jarak”);

8	//Deklarasi input variabel fuzzy jarak
	Manifold.input = jarakMusuhStatusInput;

9	// Deklarasi variabel fuzzy kecepatan
	Manifold = new FuzzyManifold (“Kecepatan”);

10	//Deklarasi input variabel fuzzy kecepatan
	Manifold.input = kecepatanMusuhStatusInput;

11	// Deklarasi variabel fuzzy HP (*Health Point*)
	Manifold = new FuzzyManifold (“HP”);

12	//Deklarasi input variabel fuzzy HP
	Manifold.input = hpMusuhStatusInput;

13	// Deklarasi variabel fuzzy Target *Priority*
	Manifold = new FuzzyManifold (“TargetPriority”);

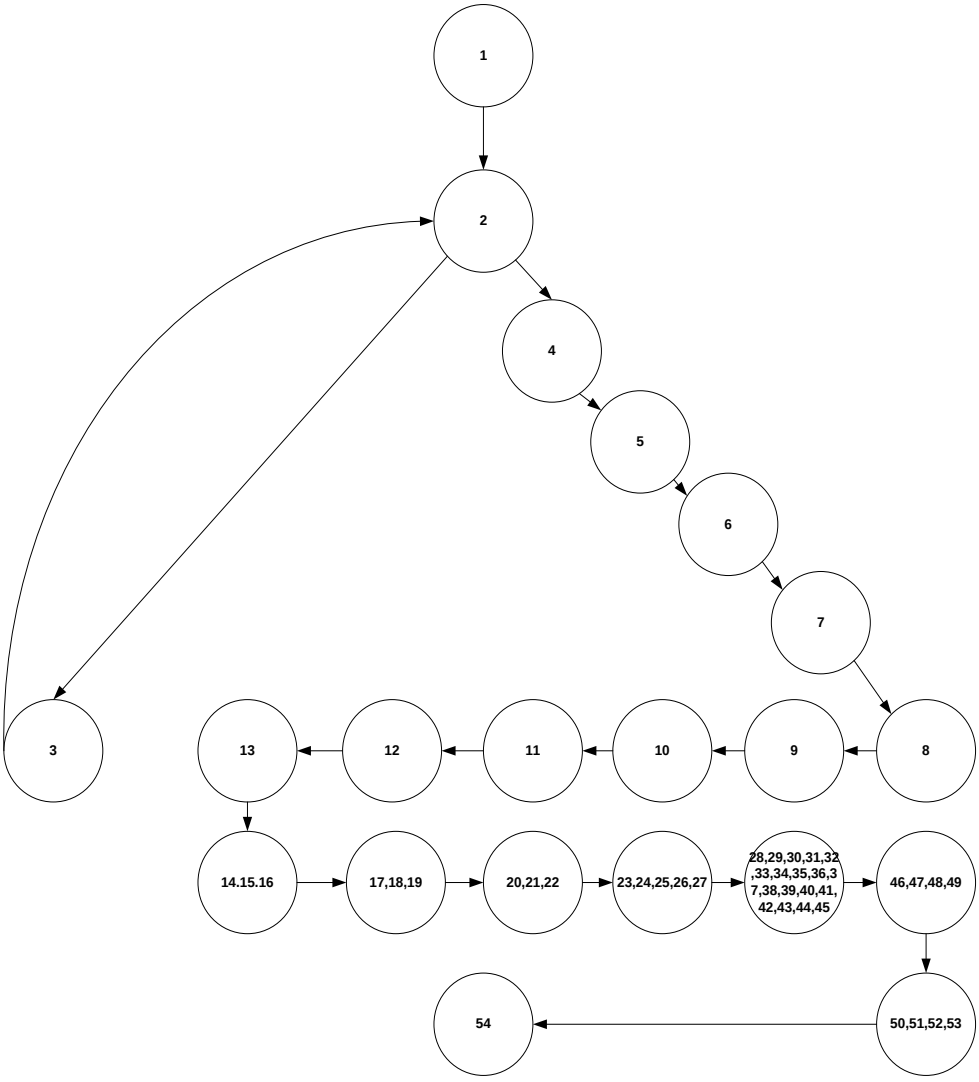
14	/* set himpunan fuzzy DEKAT dengan fungsi keanggotaan TRAPESIUM dan nilai parameter (0,0,25,125) */ func = factory.create(FuzzyMembershipFunction.TRAPEZOID, "Dekat", [0,0,25, 125]);
15	/*set himpunan fuzzy SEDANG dengan fungsi keanggotaan SEGITIGA dan nilai parameter (25,150,300) */ func = factory.create(FuzzyMembershipFunction.TRIANGLE, "Sedang", [25, 150, 300]);
16	/* set himpunan fuzzy JAUH dengan fungsi keanggotaan TRAPESIUM dan nilai parameter (150,300,400,400) */ func = factory.create(FuzzyMembershipFunction.TRAPEZOID, "Jauh", [150, 300,400,400]);
17	/* set himpunan fuzzy CEPAT dengan fungsi keanggotaan func=factory.create(FuzzyMembershipFunction.TRAPEZOID,"Cepat", 4,4,8,10);

```
18  // set himpunan fuzzy LAMBAT dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (0,0,3)
    func = factory.create(FuzzyMembershipFunction.TRIANGLE,
    "Lambat",  0, 0, 3);
19  // set himpunan fuzzy NORMAL dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (1,4,8)
    func = factory.create(FuzzyMembershipFunction.TRIANGLE,
    "Normal", 1, 4, 8 );
20  /*  set himpunan fuzzy SEDIKIT dengan fungsi keanggotaan
    SEGITIGA dan nilai parameter (0,0,500) */
    func=factory.create(FuzzyMembershipFunction.TRIANGLE,
    "Sedikit", 0,0,500);
21  /* set himpunan fuzzy Sedang dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (200,500,800) */
    func=factory.create(FuzzyMembershipFunction.TRIANGLE, "Sedang",
    200,500,800);
22  /* set himpunan fuzzy Banyak dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (500,1000,1000) */
    func=factory.create(FuzzyMembershipFunction.TRIANGLE,"Banyak",
    [500,1000,1000] );
23  /*  set himpunan fuzzy SangatRendah dengan fungsi keanggotaan
    TRAPESIUM dan nilai parameter (0,0,20,40) */
    func=factory.create(FuzzyMembershipFunction.TRAPEZOID,"SangatRe
    ndah", 0,0,20,40 );
24  /* set himpunan fuzzy Rendah dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (20,40,60) */
    func = factory.create(FuzzyMembershipFunction.TRIANGLE,
    "Rendah", [20, 40, 60] );
25  /* set himpunan fuzzy Normal dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (40,60,80) */
    func = factory.create(FuzzyMembershipFunction.TRIANGLE,
    "Normal", [40, 60, 80] );
26  /*set himpunan fuzzy Tinggi dengan fungsi keanggotaan SEGITIGA
    dan nilai parameter (60,80,100) */
    func = factory.create(FuzzyMembershipFunction.TRIANGLE,
    "Tinggi", [60, 80, 100] );
27  /* set himpunan fuzzy SangatTinggi dengan fungsi keanggotaan
    TRAPESIUM dan nilai parameter (80,100,120) */
    func = factory.create(FuzzyMembershipFunction.TRAPEZOID,
    "SangatTinggi", [80, 100,120] );
28  rule = new FuzzyRule( "IF Jarak IS Dekat AND Kecepatan IS Cepat
    THEN TargetPriority IS SangatTinggi");
29  rule = new FuzzyRule( "IF Jarak IS Dekat AND Kecepatan IS
    Normal THEN TargetPriority IS Tinggi");
30  rule = new FuzzyRule( "IF Jarak IS Dekat AND Kecepatan IS
    Lambat THEN TargetPriority IS Normal");
31  rule = new FuzzyRule( "IF Jarak IS Sedang AND Kecepatan IS
    Cepat THEN TargetPriority IS Tinggi");
32  rule = new FuzzyRule( "IF Jarak IS Sedang AND Kecepatan IS
    Normal THEN TargetPriority IS Normal");
33  rule = new FuzzyRule( "IF Jarak IS Sedang AND Kecepatan IS
    Lambat THEN TargetPriority IS Rendah");
```

```
34  rule = new FuzzyRule( "IF Jarak IS Jauh AND Kecepatan IS Cepat
    THEN TargetPriority IS Normal");
35  rule = new FuzzyRule( "IF Jarak IS Jauh AND Kecepatan IS Normal
    THEN TargetPriority IS Rendah");
36  rule = new FuzzyRule( "IF Jarak IS Jauh AND Kecepatan IS Lambat
    THEN TargetPriority IS SangatRendah");
37  rule = new FuzzyRule( "IF Jarak IS Dekat AND HP IS Banyak THEN
    TargetPriority IS SangatTinggi");
38  rule = new FuzzyRule( "IF Jarak IS Dekat AND HP IS Sedang THEN
    TargetPriority IS Tinggi");
39  rule = new FuzzyRule( "IF Jarak IS Dekat AND HP IS Sedikit THEN
    TargetPriority IS Normal");
40  rule = new FuzzyRule( "IF Jarak IS Sedang AND HP IS Banyak THEN
    TargetPriority IS Tinggi");
41  rule = new FuzzyRule( "IF Jarak IS Sedang AND HP IS Sedang THEN
    TargetPriority IS Normal");
42  rule = new FuzzyRule( "IF Jarak IS Sedang AND HP IS Sedikit
    THEN TargetPriority IS Rendah");
43  rule = new FuzzyRule( "IF Jarak IS Dekat AND HP IS Banyak THEN
    TargetPriority IS Normal");
44  rule = new FuzzyRule( "IF Jarak IS Dekat AND HP IS Sedang THEN
    TargetPriority IS Rendah");
45  rule = new FuzzyRule( "IF Jarak IS Dekat AND HP IS Sedikit THEN
    TargetPriority IS SangatRendah");
46  //fungsi implikasi dengan metode Min
    tempStek[0] = new Token(0, FuzzyOperator.fDOM, [" ", "", fuz]);
47  tempStek[1] = new Token(1, FuzzyOperator.fDOM, [" ", " ",
    fuz]);
48  var result:Token=tempStek[2] = new Token(2, FuzzyOperator.fMIN,
    [tempStek[0], tempStek[1]]);
49  // Compile ANTESENDEN dari tiap aturan fuzzy
     rule.antCompiledStek = tempStek;
50  //komposisi aturan dengan metode Max
    tempStek = new Vector.(2,true);
51  tempStek[0] = new Token(0,
    fuz.implication,[result,newToken(0,null,null,1)]);
52  tempStek[1] = new Token(1, FuzzyOperator.fAGGREGATE,[" ","
    ",fuz,rule.weight,tempStek[0]]);
53  // Compile hasil dari fungsi implikasi
    rule.conCompiledStek = tempStek;
54  //Defuzzifykasi dengan menggunakan metode centroid
    fuzzyManifolds = fuz.Defuzzify(DefuzzificationMethod.CENTROID);
```

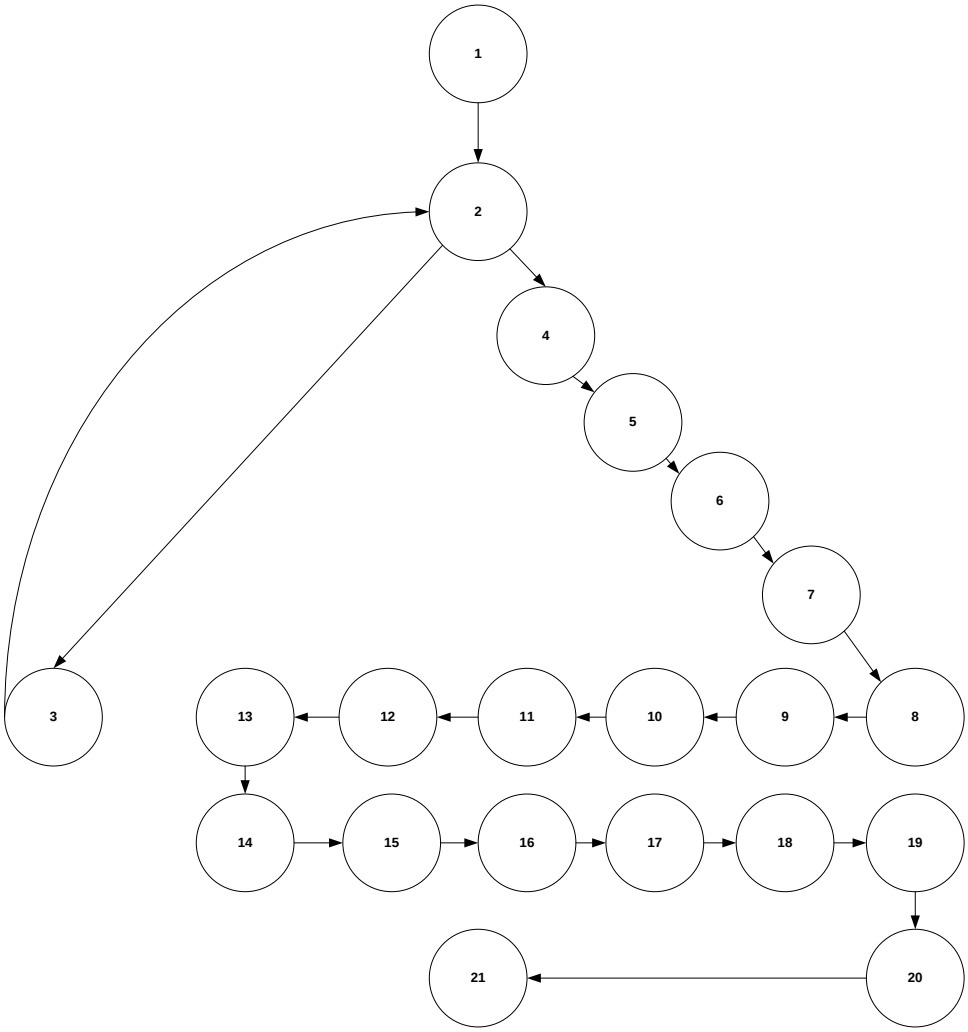
4.2.3.1 *Flow Graph*

Berikut ini merupakan *flow graph* logika *fuzzy* berdasarkan *psedocode* pada Tabel 4.2.



Gambar 4. 2 *Flow Graph* Logika *Fuzzy*

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut.



Gambar 4. 3 Penyederhanaan *Flow Graph* Logika *Fuzzy*

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

4.2.3.2 *Cyclomatic Complexity V(G)*

Dari *flow graph* di atas, dapat dihitung *cyclomatic complexity* sebagai berikut.

E = 21 N = 21

V(G) = E – N + 2

V(G) = 21 *edge* – 21 *node* + 2

V(G) = 0 + 2 = 2

Dimana:

E = jumlah *edge* pada diagram alir

N = jumlah *node* pada diagram alir

4.2.3.3 *Independent Path*

Dari hasil perhitungan *cyclomatic complexity* terdapat 4 *independent path* yaitu:

Path 1 = 1-2-3-2

Path 2 = 1-2-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21

4.2.3.4 *Graph Matric*

Penomoran ulang *flowgraph* atau penyederhanaan berdasarkan kondisi atau simpul dapat dilihat pada tabel 4.3

Tabel 4. 6 *Graph Matric* Logika ***Fuzzy***

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	N(E)-1
1		1																				1-1 = 0
2			1	1																		2-1 = 1
3		1																				1-1 = 0
4					1																	1-1 = 0
5						1																1-1 = 0
6							1															1-1 = 0
7								1														1-1 = 0
8									1													1-1 = 0
9										1												1-1 = 0
10											1											1-1 = 0
11												1										1-1 = 0
12													1									1-1 = 0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	N(E)-1
13														1								1-1 = 0
14															1							1-1 = 0
15																1						1-1 = 0
16																	1					1-1 = 0
17																		1				1-1 = 0
18																			1			1-1 = 0
19																				1		1-1 = 0
20																					1	1-1 = 0
21																						0
																	Sum(E)		+1			1+1 = 2

Keterangan:

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

4.2.3.5 Hasil Pengujian Logika *Fuzzy*

Berikut ini adalah hasil dari pengujian *whitebox* penelusuran logika *Fuzzy*

Tabel 4. 7 Hasil Pengujian Logika *Fuzzy*

No	**Path**	**Yang Diharapkan**	**Hasil Yang Dicapai**	**Kesimpulan**
1	1-2-3-2	Melakukan proses *fuzzy*fikasi apabila unit musuh dalam jarak tembak NPC *Tower*	NPC *Tower* mencari unit musuh sesuai jarak tembak sebelum masuk tahap *fuzzy*fikasi	[√] Berhasil [ ] Tidak Berhasil
2	1-2-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21	Mendapatkan nilai target *priority* dari proses *fuzzy*fikasi sesuai input berupa jarak, kecepatan dan hp dari unit musuh	Nilai target *priority* didapat berdasarkan input berupa jarak, kecepatan dan hp dari unit musuh	[√] Berhasil [ ] Tidak Berhasil

4.3 Percobaan NPC *Tower* Melawan NPC Musuh

Percobaan ini dilakukan untuk mengetahui pola perilaku NPC Tower dalam penentuan target tembak berdasarkan variabel input atau masukan menggunakan logika *fuzzy*.

4.3.1 Skenario Percobaan

Berikut ini adalah skenario percobaan NPC *Tower* melawan NPC Musuh yang akan dilakukan sebanyak tiga kali percobaan dengan jumlah unit musuh yang berbeda dari setiap percobaan.

Tabel 4. 8 Tabel Skenario Percobaan NPC Tower Melawan NPC Musuh

Percobaan	Jumlah Unit Musuh
1	2
2	3
3	4

4.3.2 Hasil Percobaan

Pada percobaan ini nilai jarak diambil dari posisi unit musuh ketika muncul di arena permainan. Nilai kecepatan dan HP merupakan hasil dari input manual user.Adapun hasil dari percobaan yang telah dilakukan dapat dilihat pada tabel dibawah ini.

Tabel 4. 9 Percobaan 1 NPC *Tower* Melawan NPC Musuh

Percobaan	Unit Musuh	Nilai Target Priority	Yang Diharapkan	Yang Dicapai	Kesimpulan
1	**Musuh 1** Nilai Jarak = 280 Input Kecepatan = 5 Input HP = 600	**50.1**	NPC *Tower* Menembak Musuh 2 karena Nilai Target *Priority* Yang Dihasilkan Lebih Tinggi Daripada Musuh 1	NPC *Tower* Menembak Musuh 2	[√] Berhasil [ ] Tidak Berhasil
	Musuh 2 Nilai Jarak = 180 Input Kecepatan = 2 Input HP = 800	**57.6**			

Tabel 4. 10 Percobaan 2 NPC *Tower* Melawan NPC Musuh

Percobaan	Unit Musuh	Nilai Target Priority	Yang Diharapkan	Yang Dicapai	Kesimpulan
2	**Musuh 1** Nilai Jarak = 180 Input Kecepatan = 3 Input HP = 300	**55.9**	NPC *Tower* Menembak Musuh 3 Karena Nilai Target *Priority* Yang Dihasilkan Lebih Tinggi Daripada Musuh 1 dan Musuh 2	NPC *Tower* Menembak Musuh 3	[√] Berhasil [ ] Tidak Berhasil
	Musuh 2 Nilai Jarak = 240 Input Kecepatan = 9 Input HP = 600	**58.9**			
	Musuh 3 Nilai Jarak = 80 Input Kecepatan = 8 Input HP = 850	**92.6**			

Tabel 4. 11 Percobaan 3 NPC *Tower* Melawan NPC Musuh

Percobaan	Unit Musuh	Nilai Target Priority	Yang Diharapkan	Yang Dicapai	Kesimpulan
3	**Musuh 1** Nilai Jarak = 250 Input Kecepatan = 5 Input HP = 640	**55.3**	NPC *Tower* Menembak Musuh 2 Karena Nilai Target *Priority* Yang Dihasilkan Lebih Tinggi Daripada Musuh 1, 3 dan 4	NPC *Tower* Menembak Musuh 2	[√] Berhasil [ ] Tidak Berhasil
	Musuh 2 Nilai Jarak = 140 Input Kecepatan = 2 Input HP = 960	**65.3**			
	Musuh 3 Nilai Jarak = 260 Input Kecepatan = 8 Input HP = 80	**41.9**			
	Musuh 4 Nilai Jarak = 120 Input Kecepatan = 2 Input HP = 540	**57.2**			

Apabila dilihat pada tabel hasil percobaan di atas. Penerapan logika *Fuzzy* pada NPC *Tower* sudah sesuai dengan hasil yang diharapkan, dimana NPC *Tower* mampu mengklasifikasikan tingkat bahaya dari setiap unit musuh sesuai nilai dari variabel masukan (jarak, kecepatan dan HP). Sehingga pemilihan target tembak atau serang pada NPC *Tower* menjadi lebih optimal.