# BAB 4
# IMPLEMENTASI DAN PENGUJIAN

## 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, baik itu lingkungan perangkat keras, maupun lingkungan perangkat lunak, serta implementasi program.

### 4.1.1 Implementasi Perangkat Keras

Perangkat keras yang dapat digunakan untuk menjalankan aplikasi *game* ini harus memiliki spesifikasi yang cukup tinggi. Spesifikasi perangkat keras yang digunakan untuk implementasi sebagai berikut :.

**Tabel 4.1 Implementasi Perangkat Keras**

| Nama Perangkat Keras | Spesifikasi |
|---|---|
| *Processor* | Core i3 2,30 GHz |
| RAM | 2GB |
| VGA | 512MB |
| *Mouse* | Standar |
| *Keyboard* | Standar |
| *Speaker* | Standar |

### 4.1.2 Implementasi Perangkat Lunak

Perangkat lunak yang dapat digunakan untuk menjalankan aplikasi *game* ini harus memiliki spesifikasi sebagai berikut :.

**Tabel 4.2 Implementasi Perangkat Lunak**

| Nama Perangkat Lunak | Spesifikasi |
|---|---|
| Sistem Operasi | Microsoft Windows 8 32bit |
| Bahasa Pemrograman | Bahasa C# |

### 4.1.3 Implementasi Antarmuka

Implementasi antarmuka ini agar dapat mengetahui performansi dari algoritma logika *fuzzy*. Berikut ini merupakan antarmuka yang ada pada aplikasi *game Rescue Flood Victim* :

1. Antarmuka Menu Utama

   Saat pengguna membuka aplikasi pertama kali, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka menu utama. Berikut ini merupakan gambar dari antarmuka menu utama yang dapat dilihat pada gambar 4.1 :




**Gambar 4.1 Antarmuka Menu Utama**

Gambar 4.1 merupakan menu utama pada *game Rescue Flood Victim*. Terdapat 3 (dua) sub menu di dalamnya, yaitu menu mulai yang berfungsi untuk memulai permainan, menu petunjuk yang berfungsi untuk melihat petunjuk permainan dan menu keluar yang berfungsi untuk keluar dari permainan.

2. Antarmuka Menu Mulai

Saat pengguna memilih menu mulai, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka dari implementasi *game Rescue Flood Victim.* Berikut ini adalah gambar antarmuka menu mulai dilihat pada gambar 4.2 :




**Gambar 4.2 Antarmuka Menu Mulai**

Gambar 4.2 merupakan menu mulai pada *game Rescue Flood Victim*. Terdapat informasi jarak, tinggi air dan stamina korban yang merupakan implementasi dari algoritma logika *fuzzy*. Lalu terdapat pula informasi kouta perahu, waktu dan misi penyelamatan korban yang berada didalam *game.*

3. Antarmuka Menu Petunjuk

Saat pengguna memilih menu petunjuk, maka antarmuka yang ditampilkan oleh sistem adalah informasi mengenai petunjuk menggerakan perahu dan penyelamatan korban yang didapatkan dari *game Rescue Flood Victim*. Berikut ini merupakan gambar pada antarmuka menu petunjuk dilihat dari gambar 4.3 :




**Gambar 4.3 Antarmuka Menu Petunjuk**

## 4.2 Pengujian Sistem

Pengujian sistem memiliki 3 (tiga) macam teknik pengujian yaitu pengujian *white box,* pengujian *black box* dan pengujian performansi . Pengujian yang menggunakan teknik *white box* lebih menitikberatkan pada struktur internal (*source code*) program, di mana setiap baris kode yang akan di *compile* dicek satu persatu untuk mengetahui apakah masih terjadi *error*. Pengujian pada *black box* tidak terlalu memperhatikan struktur internal (*source code*) dari program, tetapi lebih kepada menemukan kesalahan pada program. Pada perancangan *fungsional* digunakan pengujian *black box*. Pengujian performansi dapat memverifikasi bahwa sistem memenuhi spesifikasi diklaim oleh produsen atau vendor. Proses ini dapat membandingkan dua atau lebih perangkat atau program dalam hal parameter.

### 4.2.1 Pengujian *Black Box*

Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Pengujian *black box* berfokus pada persyaratan *fungsional* perangkat lunak. Berikut ini pengujian dari *game Rescue Flood Victim* dapat dilihat pada tabel 4.3 :

**Tabel 4.3 Pengujian *Black Box***

| No | Komponen yang di uji | Jenis Pengujian |
|---|---|---|
| 1. | Mulai Permainan | *Black Box* |
| 2. | Permainan Berjalan | *Black Box* |
| 3. | Menjeda Permainan | *Black Box* |
| 4. | Permainan Selesai | *Black Box* |
| 5. | Petunjuk Permainan | *Black Box* |
| 6. | Keluar Permainan | *Black Box* |

#### 4.2.1.1 Kasus dan Hasil Pengujian *Black Box*

Berikut ini adalah hasil dari pengujian perangkat lunak yang sudah dibangun menggunakan metode *black box* berdasarkan pengamatan yang sudah dilakukan dan menghasilkan kesimpulan dari pengujian tersebut.

1. Pengujian Memulai Permainan

   Pengujian memulai permainan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.4 :

**Tabel 4.4 Pengujian Memulai Permainan**

| Kasus yang akan di uji | Hasil Uji | |
|---|---|---|
| | Hasil yang diharapkan | Kesimpulan |
| Memulai Permainan | Sistem menampilkan layar permainan | [✓] Berhasil<br>[ ] Tidak Berhasil |

2. Pengujian Permainan Berjalan

   Pengujian permainan berjalan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.5 :

**Tabel 4.5 Pengujian Permainan Berjalan**

| Kasus yang akan di uji | Hasil Uji | |
|---|---|---|
| | Hasil yang diharapkan | Kesimpulan |
| Permainan Berjalan | Sistem menjalankan permainan dan memproses logika *fuzzy* | [✓] Berhasil<br>[ ] Tidak Berhasil |

3. Pengujian Menjeda Permainan

   Pengujian menjeda permainan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.6 :

**Tabel 4.6 Pengujian Menjeda Permainan**

| Kasus yang akan di uji | Hasil Uji | |
|---|---|---|
| | Hasil yang diharapkan | Kesimpulan |
| Menjeda Permainan | Sistem menampilkan jeda permainan | [✓] Berhasil<br>[ ] Tidak Berhasil |

4. Pengujian Permainan Selesai

   Pengujian permainan selesai dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.7 :

**Tabel 4.7 Pengujian Permainan Selesai**

| Kasus yang akan di uji | Hasil Uji | |
|---|---|---|
| | Hasil yang diharapkan | Kesimpulan |
| Permainan Selesai | Sistem menampilkan permainan selesai | [✓] Berhasil<br>[ ] Tidak Berhasil |

5. Pengujian Petunjuk Permainan

   Pengujian petunjuk permainan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.8 :

**Tabel 4.8 Pengujian Petunjuk Permainan**

| Kasus yang akan di uji | Hasil Uji | |
|---|---|---|
| | Hasil yang diharapkan | Kesimpulan |
| Petunjuk Permainan | Sistem menampilkan petunjuk permainan | [✓] Berhasil<br>[ ] Tidak Berhasil |

6. Pengujian Keluar Permainan

   Pengujian keluar permainan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.9 :

**Tabel 4.9 Pengujian Keluar Permainan**

| Kasus yang akan di uji | Hasil Uji | |
|---|---|---|
| | Hasil yang diharapkan | Kesimpulan |
| Keluar Permainan | Sistem keluar dari permainan | [✓] Berhasil<br>[ ] Tidak Berhasil |

#### 4.2.1.2 Pengujian *White Box*

Pengujian *white box* merupakan pengujian yang didasarkan pada pengecekan terhadap detail perancangan dengan menggunakan stuktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Tujuan dari pengujian *white box* untuk mengetahui cara kerja suatu aplikasi atau perangkat lunak secara internal dan untuk menjamin operasi-operasi internal sesuai dengan spesifikasi yang telah ditetapkan dengan menggunakan struktur kontrol dari prosedur yang dirancang. Berikut adalah *pseudocode* untuk pembentukan himpunan *fuzzy* yang terdapat pada tabel 4.10 :

**Tabel 4.10 *Pseudocode* Himpunan *Fuzzy***

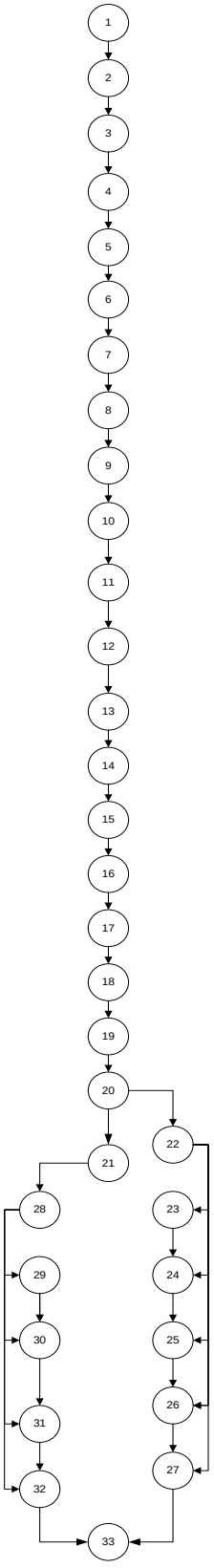
| | |
|---|---|
| | //variabel diagram jarak |
| 1 | public float hasil_jarak_dekat; |
| 2 | float[] jarak_dekat = new float[3] {0,30,40}; |
| 3 | public float hasil_jarak_sedang; |

```
4   float[] jarak_sedang = new float[4] {30,40,60,70};
5   public float hasil_jarak_jauh;
6   float[] jarak_jauh = new float[3] {60,70,100};
    //variabel diagram stamina
7   public float hasil_stamina_lemah;
8   float[] stamina_lemah = new float[3] {0,30,40};
9   public float hasil_stamina_sedang;
10  float[] stamina_sedang = new float[4] {30,40,60,70};
11  public float hasil_stamina_kuat;
12  float[] stamina_kuat = new float[3] {60,70,100};
    //variabel diagram air
13  public float hasil_air_rendah;
14  float[] air_rendah = new float[3] {0,70,80};
15  public float hasil_air_sedang;
16  float[] air_sedang = new float[4] {70,80,130,140};
17  public float hasil_air_tinggi;
18  float[] air_tinggi = new float[3] {130,140,200};
19  keputusan=0;
20  hasil_menyelamatkan_diri=0;
21  hasil_bertahan=0;
22  for (int i=0; i<menyelamatkan_diri.Length; i++) {
23  if (menyelamatkan_diri[i]>0.0f) {
24  if (hasil_menyelamatkan_diri<=0.0f) {
25  hasil_menyelamatkan_diri = menyelamatkan_diri[i];
26  } else if (menyelamatkan_diri[i]>=hasil_menyelamatkan_diri)
    {hasil_menyelamatkan_diri = menyelamatkan_diri[i]; } } }
27  for (int i=0; i<bertahan.Length; i++) {
28  if (bertahan[i]>0.0f)
29  { if (hasil_bertahan<=0.0f)
30  { hasil_bertahan = bertahan[i]; }else
31  if(bertahan[i]>=hasil_bertahan) {
```

| | |
|---|---|
| 32 | hasil_bertahan = bertahan[i]; } } }<br>//defuzzyfikasi hasil keputusan |
| 33 | keputusan =<br>(hasil_menyelamatkan_diri*100+hasil_bertahan*50) /<br>(hasil_menyelamatkan_diri+hasil_bertahan); |

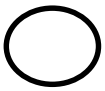
a. *Flow Graph* logika *fuzzy*

Berikut ini merupakan *flow graph* dari logika *fuzzy* yang dapat dilihat dari gambar 4.4 :




**Gambar 4.4 *Flow Graph* Logika *Fuzzy***

Keterangan :

 = Menggambarkan Kondisi

 = Menggambarkan Aksi

b. *Cyclomatic Complexity* V(G)

E = 33 , N = 26

V(G) = E – N + 2

V(G) = 33 – 26 + 2

V(G) = 9

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

c. *Independent Path*

*Path 1* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-28-29-30-31-32-33

*Path 2* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-28-30-31-32-33

*Path 3* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-28-31-32-33

*Path 4* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-28-32-33

*Path 5* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-23-24-25-26-27-33

*Path 6* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-24-25-26-27-33

*Path 7* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-25-26-27-33

*Path 8* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-26-27-33

*Path 9* = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-22-27-33

d. *Graph Matriks Logika Fuzzy*

**Tabel 4.11 *Graph Matriks* Logika *Fuzzy***

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | N(E-1) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 2 | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 3 | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 4 | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 5 | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 6 | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 7 | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 8 | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 9 | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 10 | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | 0 |
| 11 | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 0 |
| 12 | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 0 |
| 13 | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | 0 |
| 14 | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | 0 |
| 15 | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | 0 |
| 16 | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | 0 |
| 17 | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | 0 |
| 18 | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | 0 |
| 19 | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | 0 |
| 20 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | 0 |
| 21 | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | | | | | 1 |
| 22 | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | | | | 1 |
| 23 | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | | | 1 |
| 24 | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | | 1 |
| 25 | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | 1 |
| 26 | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | 1 |
| 27 | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 1 | 1 |
| 28 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | 0 |
| 29 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | 0 |
| 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | 0 |
| 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | 0 |
| 32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 |
| 33 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SUM(E+1) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 8 |

Keterangan :

1. Baris dan kolom merepresentasikan simpul.
2. Nilai 1 merepresentasikan adanya keterhubungan antar node/simpul. yang dibuat sesuai alur logika dari program tersebut. penggunaan kondisi memengaruhi jumlah node dan edge dari *flow graph* tersebut.
    - Baris 1 dan kolom 2 = merepresentasikan adanya keterhubungan antar node/simpul 1 dan node/simpul 2 dari *flow graph*.

- Baris 2 dan kolom 3 = merepresentasikan adanya keterhubungan antar node/simpul 2 dan node/simpul 3 dari *flow graph*.
- Baris 3 dan kolom 4 = merepresentasikan adanya keterhubungan antar node/simpul 3 dan node/simpul 4 dari *flow graph*.
- Baris 4 dan kolom 5 = merepresentasikan adanya keterhubungan antar node/simpul 4 dan node/simpul 5 dari *flow graph*.
- Baris 5 dan kolom 6 = merepresentasikan adanya keterhubungan antar node/simpul 5 dan node/simpul 6 dari *flow graph*.
- Baris 6 dan kolom 7 = merepresentasikan adanya keterhubungan antar node/simpul 6 dan node/simpul 7 dari *flow graph*.
- Baris 7 dan kolom 8 = merepresentasikan adanya keterhubungan antar node/simpul 7 dan node/simpul 8 dari *flow graph*.
- Baris 8 dan kolom 9 = merepresentasikan adanya keterhubungan antar node/simpul 8 dan node/simpul 9 dari *flow graph*.
- Baris 9 dan kolom 10 = merepresentasikan adanya keterhubungan antar node/simpul 9 dan node/simpul 10 dari *flow graph*.
- Baris 10 dan kolom 11 = merepresentasikan adanya keterhubungan antar node/simpul 10 dan node/simpul 11 dari *flow graph*.
- Baris 11 dan kolom 12 = merepresentasikan adanya keterhubungan antar node/simpul 11 dan node/simpul 12 dari *flow graph*.
- Baris 12 dan kolom 13 = merepresentasikan adanya keterhubungan antar node/simpul 12 dan node/simpul 13 dari *flow graph*.
- Baris 13 dan kolom 14 = merepresentasikan adanya keterhubungan antar node/simpul 13 dan node/simpul 14 dari *flow graph*.
- Baris 14 dan kolom 15 = merepresentasikan adanya keterhubungan antar node/simpul 14 dan node/simpul 15 dari *flow graph*.
- Baris 15 dan kolom 16 = merepresentasikan adanya keterhubungan antar node/simpul 15 dan node/simpul 16 dari *flow graph*.
- Baris 16 dan kolom 17 = merepresentasikan adanya keterhubungan antar node/simpul 16 dan node/simpul 17 dari *flow graph*.

- Baris 17 dan kolom 18 = merepresentasikan adanya keterhubungan antar node/simpul 17 dan node/simpul 18 dari *flow graph*.
- Baris 18 dan kolom 19 = merepresentasikan adanya keterhubungan antar node/simpul 18 dan node/simpul 19 dari *flow graph*.
- Baris 19 dan kolom 20 = merepresentasikan adanya keterhubungan antar node/simpul 19 dan node/simpul 20 dari *flow graph*.
- Baris 20 dan kolom 21 = merepresentasikan adanya keterhubungan antar node/simpul 18 dan node/simpul 19 dari *flow graph*.
- Baris 21 dan kolom 22 dan 27 = merepresentasikan adanya keterhubungan antar node/simpul 18 dan node/simpul 19 dari *flow graph*.
- Baris 22 dan kolom 23 dan 28 = merepresentasikan adanya keterhubungan antar node/simpul 22, node/simpul 23, node/simpul 28 dari *flow graph*.
- Baris 23 dan kolom 24 dan 29 = merepresentasikan adanya keterhubungan antar node/simpul 23, node/simpul 24, node/simpul 29 dari *flow graph*.
- Baris 24 dan kolom 25 dan 30 = merepresentasikan adanya keterhubungan antar node/simpul 24, node/simpul 25, node/simpul 30 dari *flow graph*.
- Baris 25 dan kolom 26 dan 31 = merepresentasikan adanya keterhubungan antar node/simpul 25, node/simpul 26, node/simpul 27 dari *flow graph*.
- Baris 26 dan kolom 27 dan 32 = merepresentasikan adanya keterhubungan antar node/simpul 26, node/simpul 27, node/simpul 32 dari *flow graph*.
- Baris 27 dan kolom 28 dan 33 = merepresentasikan adanya keterhubungan antar node/simpul 27, node/simpul 28, node/simpul 33 dari *flow graph*.
- Baris 28 dan kolom 29 = merepresentasikan adanya keterhubungan antar node/simpul 28 dan node/simpul 29 dari *flow graph*.

- Baris 29 dan kolom 30 = merepresentasikan adanya keterhubungan antar node/simpul 29 dan node/simpul 30 dari *flow graph*.
- Baris 30 dan kolom 31 = merepresentasikan adanya keterhubungan antar node/simpul 30 dan node/simpul 31 dari *flow graph*.
- Baris 31 dan kolom 32 = merepresentasikan adanya keterhubungan antar node/simpul 31 dan node/simpul 32 dari *flow graph*.
- Baris 32 dan kolom 33 = merepresentasikan adanya keterhubungan antar node/simpul 32 dan node/simpul 33 dari *flow graph*.

#### 4.2.1.3 Pengujian Performansi

*Performance Testing* atau pengujian performansi merupakan proses menentukan kecepatan atau efektivitas, jaringan program computer, perangkat lunak atau perangkat. Proses ini dapat membandingkan dua atau lebih perangkat atau program dalam hal parameter [11]. Berikut ini merupakan pengujian performansi untuk parameter logika *fuzzy* didalam *game* :

**Tabel 4.12 Pengujian Performansi Logika *Fuzzy***

| No | Uji Parameter | Fuzzifikasi | Defuzzifikasi | Keputusan |
|---|---|---|---|---|
| 1 | Input Jarak = 50<br>Input Stamina = 50<br>Input Air = 50 | Jarak Dekat(50) = 0<br>Jarak Sedang(50) = 1<br>Jarak Jauh(50) = 0<br>Stamina Lemah(50) = 0<br>Stamina Sedang(50) = 1<br>Stamina Kuat(50) = 0<br>Air Rendah(50) = 1<br>Air Sedang(50) = 0<br>Air Tinggi(50) = 0 | Z = 50 | Bertahan |
| 2 | Input Jarak = 80<br>Input Stamina = 100<br>Input Air = 100 | Jarak Dekat(80) = 0<br>Jarak Sedang(80) = 0<br>Jarak Jauh(80) = 1<br>Stamina Lemah(100) = 0 | Z = 50 | Bertahan |

| | | | | |
|---|---|---|---|---|
| | | Stamina Sedang(100) = 0<br>Stamina Kuat(100) = 1<br>Air Rendah(100) = 0<br>Air Sedang(100) = 1<br>Air Tinggi(100) = 0 | | |
| 3 | Input Jarak = 60<br>Input Stamina = 50<br>Input Air = 135 | Jarak Dekat(75) = 0<br>Jarak Sedang(75) = 1<br>Jarak Jauh(75) = 0<br>Stamina Lemah(70) = 0<br>Stamina Sedang(70) = 1<br>Stamina Kuat(70) = 0<br>Air Rendah(120) = 0<br>Air Sedang(120) = 0.5<br>Air Tinggi(120) = 0.5 | Z = 75 | Menyelamatkan diri |
| 4 | Input Jarak = 35<br>Input Stamina = 65<br>Input Air = 135 | Jarak Dekat(35) = 0.5<br>Jarak Sedang(35) = 0.5<br>Jarak Jauh(35) = 0<br>Stamina Lemah(65) = 0<br>Stamina Sedang(65) = 0.5<br>Stamina Kuat(65) = 0.5<br>Air Rendah(135) = 0<br>Air Sedang(135) = 0.5<br>Air Tinggi(135) = 0.5 | Z = 75 | Menyelamatkan diri |