BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan *Game*, tahap ini merupakan tahap dimana *Game* siap untuk dioperasikan. *Game* ini diimplementasikan dengan nama *building of house* dan *file extension*-nya *.xml* (*application package file*).

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang dapat digunakan untuk menjalankan *Game* ini tidak harus memiliki spesifikasi yang tinggi, seperti *processor Core* I5 dan *VGA Card* 1GB. Sebagai ilustrasi sistem minimum agar dapat menjalankan *Game building of house* adalah sebagai berikut :

1. *Processor athlon*.
2. *RAM* dengan kapasitas minimum 256*MB*.
3. *VGA Card* dengan kapasitas minimum 256 *MB*.

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan *Game building of house* adalah *flash player* dan *mozzila firefox*.

4.1.3 Implementasi Instalasi *Game*

Sebelum dapat memainkan *Game building of house*, koneksi internet harus dalam keadaan online. Petunjuk dalam instalasi *Game* ini adalah sebagai berikut :

1. Buka *mozzila firefox*.
2. Klik file buildingofhouse dengan extension file .html
3. Setelah itu, *Game* tersebut akan menampilkan permainan.

4.1.4 Implementasi Antarmuka

Adapun antarmuka yang ada pada *Game Building of House* :

1. Antarmuka Menu Utama

Saat pengguna membuka aplikasi pertama kali, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka menu utama. Berikut ini merupakan tampilan dari antarmuka menu utama :



Gambar 4. 1 Antarmuka Menu Utama

Gambar 4.1 merupakan menu utama pada *Game building of house*. Terdapat 2 sub menu di dalamnya, yaitu menu Mulai yang berfungsi untuk memulai permainan, menu Tentang yang berfungsi untuk melihat informasi permainan.

2. Antarmuka Intro

Saat pengguna memilih menu Mulai, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka Intro. Berikut ini merupakan tampilan dari antarmuka Intro :

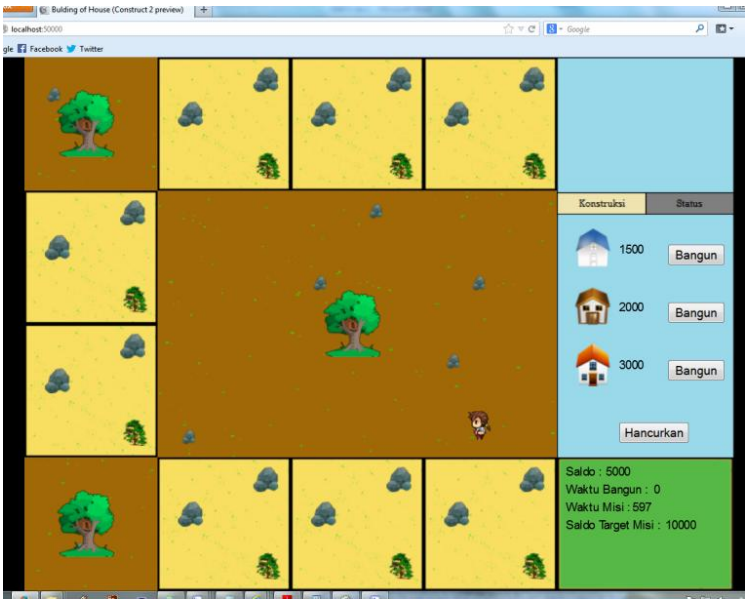


Gambar 4. 2 Antarmuka *Intro*

Gambar 4.2 merupakan antarmuka Intro yang terdapat pada *Game Building of House*.

3. Antarmuka Arena Permainan

Saat pengguna menekan tombol lanjut pada antarmuka Intro, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka Arena Permainan. Berikut ini merupakan tampilan dari antarmuka Arena Permainan :



Gambar 4. 3 Antarmuka Arena Permainan

Gambar 4.3 merupakan antarmuka Arena Permainan yang terdapat pada *Game Building of House*.

4.2 Pengujian Sistem

Pada umunya ada 2 (dua) macam teknik pengujian yaitu *white box testing* dan *black box testing*. Pengujian yang menggunakan teknik *white box testing* lebih menitik beratkan 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 testing* tidak terlalu memperhatikan struktur internal (*source code*) dari program, tetapi lebih kepada menemukan kesalahan program. Cara pengujian dilakukan melalui dua tahap, yaitu pengujian *alpha* (fungsional) dan pengujian *beta*.

4.2.1 Pengujian *Alpha*

Pengujian dilakukan terhadap *Game* untuk memastikan bahwa *Game* dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Pengujian *alpha* berfokus pada persyaratan fungsional perangkat lunak.

4.2.1.1 *Skenario Pengujian Game*

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada *Game building of house.* Skenario pengujian yang akan dilakukan pada ini selengkapnya dapat dilihat pada table 4.1 :

Tabel 4. 1Skenario Pengujian *Game*

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1.	Menu Utama	Klik menu Mulai	*Black box*
		Klik menu Tentang	*Black box*
2.	Menu Mulai	Klik *Mulai*	*Black box*
		Permainan *level* 1	*White box*
		Klik lahan	*Black box*
		Klik bangun	*Black box*
		Klik terima	*Balck box*
		Klik tolak	*Black box*
		Klik pesan	*Black box*
		Permainan *level* 2	*White box*
		Klik lahan	*Black box*
		Klik bangun	*Black box*
		Klik terima	*Black box*
		Klik tolak	*Black box*
		Klik pesan	*Black box*
		Permainan *level* 3	*White box*
		Klik lahan	*Black box*
		Klik bangun	*Black box*
		Klik terima	*Black box*
		Klik tolak	*Black box*
		Klik pesan	*Black box*
3.	Menu Tentang	Klik Menu Tentang	*Black box*
		Klik tombol kembali	*Black box*
4.	Algoritma *fuzzy*	Algoritma *fuzzy* pada level 1	*White box*
		Algoritma *fuzzy* pada level 2	*White box*
		Algoritma *fuzzy* pada level 3	*White box*

4.2.1.2 Kasus dan Hasil Pengujian *(Black Box Testing)*

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 4.1 yang sudah dijabarkan. Hasil pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada tabel 4.2 :

Tabel 4. 2 Pengujian *Black Box*

No	Kasus yang akan di uji	Skenario dan Hasil Uji		
		Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
1.	Menu Utama	Klik menu Mulai	Sistem menampilkan halaman permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik menu Tentang	Sistem menampilkan halaman menu Tentang	[√] Berhasil [ ] Tidak Berhasil
2.	Menu Mulai	Klik menu Mulai	Sistem menampilkan halaman permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan dunia *level* 1	[√] Berhasil [ ] Tidak Berhasil
		Permainan *Level* 1	Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)	
		Klik lahan	Sistem menampilkan gambar lahan	[√] Berhasil [ ] Tidak Berhasil
		Klik bangun	Sistem menampilkan bangunan rumah	[√] Berhasil [ ] Tidak Berhasil
		Klik terima	Sistem memunculkan status negosiasi	[√] Berhasil [ ] Tidak Berhasil
		Klik tolak	Sistem memunculkan status negosiasi	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan dunia *level* 2	[√] Berhasil [ ] Tidak Berhasil
		Permainan *Level* 2	Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box*	

			Testing)	
		Klik lahan	Sistem menampilkan gambar lahan	[√] Berhasil [ ] Tidak Berhasil
		Klik bangun	Sistem menampilkan bangunan rumah	[√] Berhasil [ ] Tidak Berhasil
		Klik terima	Sistem memunculkan status negosiasi	[√] Berhasil [ ] Tidak Berhasil
		Klik tolak	Sistem mememunculkan status negosiasi	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan dunia *level* 3	[√] Berhasil [ ] Tidak Berhasil
		Permainan *Level* 3	Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)	
		Klik lahan	Sistem menampilkan gambar lahan	[√] Berhasil [ ] Tidak Berhasil
		Klik bangun	Sistem menampilkan bangunan rumah	[√] Berhasil [ ] Tidak Berhasil
		Klik terima	Sistem memunculkan status negosiasi	[√] Berhasil [ ] Tidak Berhasil
		Klik tolak	Sistem memunculkan status negosiasi	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan permainan telah berakhir	[√] Berhasil [ ] Tidak Berhasil
3.	Menu Tentang	Klik Menu Tentang	Sistem menampilkan informasi permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik kembali	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil

4.2.1.3 Kasus dan Hasil Pengujian *(White box Testing)*

Berikut ini merupakan kasus menu Mulai untuk menguji permainan *level* 1, permainan *level* 2, dan permainan *level* 3 di dalam perangkat lunak yang sudah dibangun dengan menggunakan metode *white box testing* berdasarkan skenario pengujian *Game* yang terdapat pada tabel 4.1 yang sudah dijabarkan.

1. Pengujian Permainan *level* 1 (*white box*)

   Pengujian permainan *level* 1 merupakan proses dimana akan diuji algoritma *fuzzy* terhadap pelanggan dalam melakukan pencarian rumah. Berikut ini merupakan pengujian algoritma *fuzzy* yang di terapkan pada permainan *level* 1 dengan menggunakan metode *white box*.

   a. Algoritma Yang Akan Di Uji

   Berikut ini merupakan algoritma *fuzzy* untuk *level* 1 :

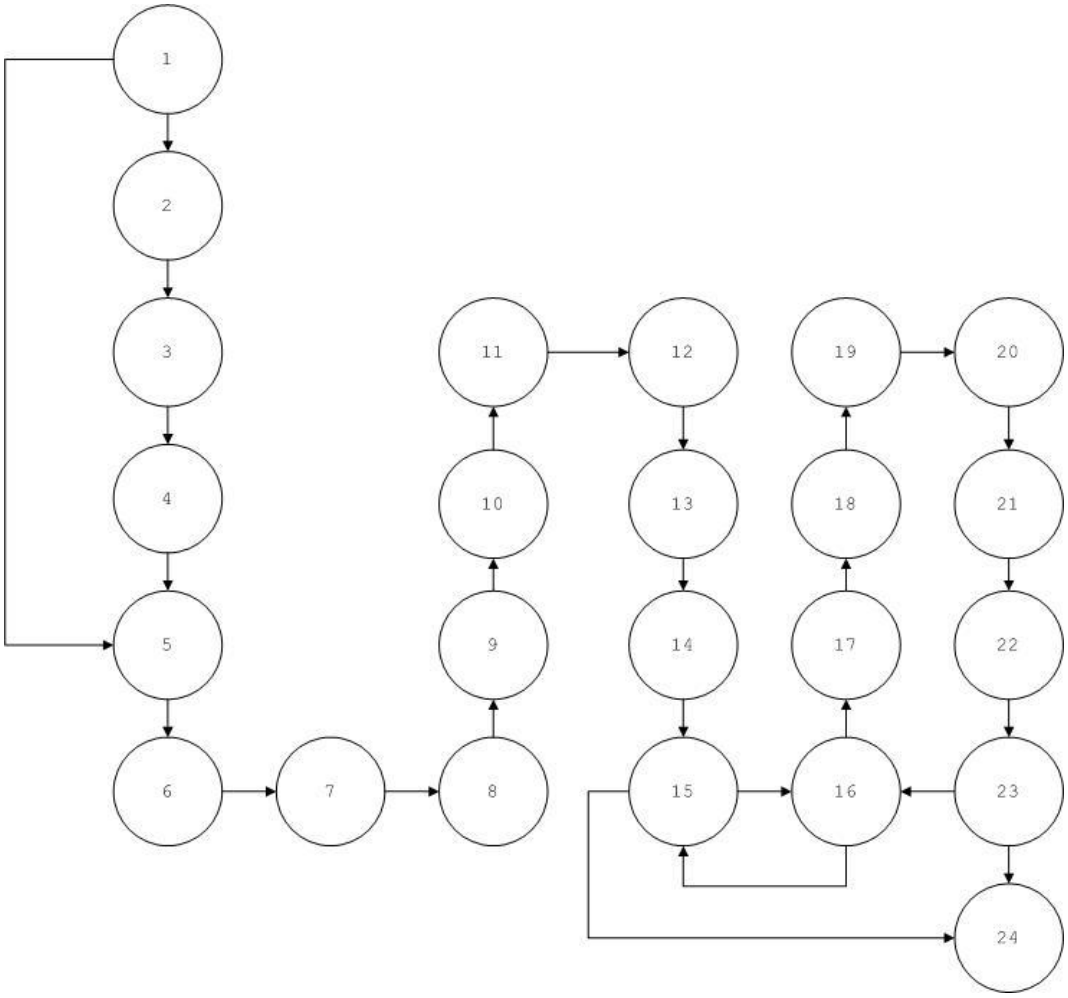
```
1   for(int j=0; j <= 525; j += 100)
    {
2   if(j==150)
    {
    harga1 = hargafuzzy();
    ukuran1 = ukuranfuzzy();
    K_NPC1 = harga1 + ukuran1;
3   } else if(j==275)
    {
    harga2 = hargafuzzy();
    ukuran2 = ukuranfuzzy();
    K_NPC2 = harga2 + ukuran2;
4   } else if(j==375)
    {
    harga3 = hargafuzzy();
    ukuran3 = ukuranfuzzy();
    K_NPC3 = harga3 + ukuran3;
    }
5   }
```

```
6   if(K_NPC1 == 1 && K_NPC1 <= 150)
    {
    house[1] = bagus;
7   } else if(K_NPC1 <= 2 && K_NPC1 <= 350)
    {
    house[2] = sedang;
8   } else if(K_NPC1 <= 3 && K_NPC1 <= 550)
    {
    house[3] = mahal;
    }
9   if(K_NPC2 == 1 && K_NPC2 <= 150)
    {
    house[1] = bagus;
10  } else if(K_NPC2 <= 2 && K_NPC2 <= 350)
    {
    house[2] = sedang;
11  } else if(K_NPC2 <= 3 && K_NPC2 <= 550)
    {
    house[3] = mahal;
    }
12  if(K_NPC3 == 1 && K_NPC3 <= 150)
    {
    house[1] = bagus;
13  } else if(K_NPC3 <= 2 && K_NPC3 <= 350)
    {
    house[2] = sedang;
14  } else if(K_NPC3 <= 3 && K_NPC3 <= 550)
    {
    house[3] = mahal;
    }
15  for(int i=0; i<=3; i++)
```

```
    {
16  if(house[i] != null && house[i] != this.Tree())
    {
17  if(house[i] == bagus)
    {
    this.EAI();
18  } else if(house[i] != bagus)
    {
    this.Move();
    }
19  if(house[i] == sedang)
    {
    this.EAI();
20  } else if(house[i] != sedang)
    {
    this.Move();
    }
21  if(house[i] == mahal)
    {
    this.EAI();
22  } else if(house[i] != mahal)
    {
    this.Move();
    }
23  }
24  }
```

b. *Flow Graph* Permainan *Level* 1

Berikut ini merupakan *flow graph* permainan *level* 1 yang dapat dilihat pada gambar 4.4 :



Gambar 4. 4 *Flow Graph* Permainan *Level* 1

Keterangan :

= Menggambarkan Kondisi

→ = Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 27, N = 24

V(G) = E – N + 2

V(G) = 27 – 24 + 2

V(G) = 3 + 2 = 5

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1),(2),(3),(4),(5)

Path 2 = (1),(6),(7),(8),(9),(10),(11)

Path 3 = (1),(9),(10),(11),(12),(13),(14)

Path 4 = (1) ,(15),(16),(17),(18),(19),(20)

Path 5 = (1),(19),(20),(21),(22),(23),(24)

e. *Graph Matriks* Permainan *Level* 1

Tabel 4. 3 *Graph Matriks* Permainan *Level* 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N =(E-1)
1	1				1																				1
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								1	1
1															1		1								1

6																							
17																	1						0
18																		1					0
19																			1				0
20																				1			0
21																					1		0
22																						1	0
23															1								1	1
24																								0
SUM(E)+1																								4+1=

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Kesimpulan

Dari hasil pengujian permainan *level* 1 yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *path* telah dikerjakan, serta tidak terjadi pengulangan.

2. Pengujian Permainan *level* 2 (*white box*)

Pengujian permainan *level* 2 merupakan proses dimana akan diuji algoritma *fuzzy* terhadap musuh dalam melakukan pencarian rute terpendek. Berikut ini merupakan pengujian algoritma *fuzzy* yang di terapkan pada permainan *level* 2 dengan menggunakan metode *white box*.

a. Algoritma Yang Akan Di Uji

Berikut ini merupakan algoritma *fuzzy* untuk *level* 2 :

```
1   for(int j=0; j <= 525; j += 100)
    {
```

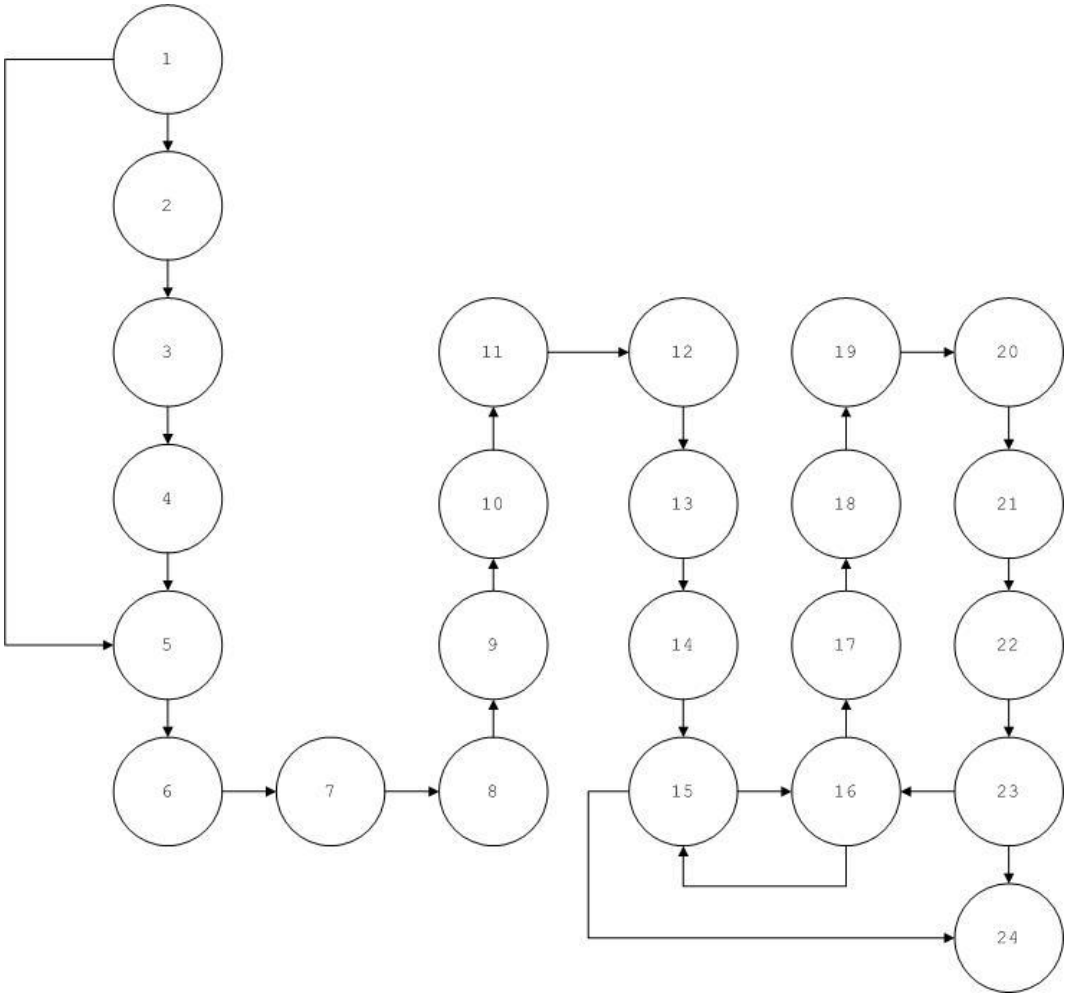
```
2   if(j==150)
    {
    harga1 = hargafuzzy();
    ukuran1 = ukuranfuzzy();
    K_NPC1 = harga1 + ukuran1;
3   } else if(j==275)
    {
    harga2 = hargafuzzy();
    ukuran2 = ukuranfuzzy();
    K_NPC2 = harga2 + ukuran2;
4   } else if(j==375)
    {
    harga3 = hargafuzzy();
    ukuran3 = ukuranfuzzy();
    K_NPC3 = harga3 + ukuran3;
    }
5   }
6   if(K_NPC1 == 1 && K_NPC1 <= 150)
    {
    house[1] = bagus;
7   } else if(K_NPC1 <= 2 && K_NPC1 <= 350)
    {
    house[2] = sedang;
8   } else if(K_NPC1 <= 3 && K_NPC1 <= 550)
    {
    house[3] = mahal;
    }
9   if(K_NPC2 == 1 && K_NPC2 <= 150)
    {
    house[1] = bagus;
10  } else if(K_NPC2 <= 2 && K_NPC2 <= 350)
```

```
{
house[2] = sedang;
} else if(K_NPC2 <= 3 && K_NPC2 <= 550)
{
house[3] = mahal;
}
if(K_NPC3 == 1 && K_NPC3 <= 150)
{
house[1] = bagus;
} else if(K_NPC3 <= 2 && K_NPC3 <= 350)
{
house[2] = sedang;
} else if(K_NPC3 <= 3 && K_NPC3 <= 550)
{
house[3] = mahal;
}
for(int i=0; i<=3; i++)
{
if(house[i] != null && house[i] != this.Tree())
{
if(house[i] == bagus)
{
this.EAI();
} else if(house[i] != bagus)
{
this.Move();
}
if(house[i] == sedang)
{
this.EAI();
} else if(house[i] != sedang)
```

```
{
this.Move();
}
21  if(house[i] == mahal)
{
this.EAI();
22  } else if(house[i] != mahal)
{
this.Move();
}
23  }
24  }
```

b. *Flow Graph* Permainan *Level* 2

Berikut ini merupakan *flow graph* permainan *level* 2 yang dapat dilihat pada gambar 4.5 :



Gambar 4. 5 *Flow Graph* Permainan *Level* 2

Keterangan :

= Menggambarkan Kondisi

→ = Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 27, N = 24

V(G) = E – N + 2

V(G) = 27 – 24 + 2

V(G) = 3 + 2 = 5

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1),(2),(3),(4),(5)

Path 2 = (1),(6),(7),(8),(9),(10),(11)

Path 3 = (1),(9),(10),(11),(12),(13),(14)

Path 4 = (1) ,(15),(16),(17),(18),(19),(20)

Path 5 = (1),(19),(20),(21),(22),(23),(24)

e. *Graph Matriks* Permainan *Level* 2

Tabel 4. 4 *Graph Matriks* Permainan *Level* 2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N =(E-1)
1	1				1																				1
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								1	1
1															1		1								1

6																							
17																	1						0
18																		1					0
19																			1				0
20																				1			0
21																					1		0
22																						1	0
23															1								1	1
24																								0
SUM(E)+1																								4+1=

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Kesimpulan

Dari hasil pengujian permainan *level* 2 yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *path* telah dikerjakan, serta tidak terjadi pengulangan.

3. Pengujian Permainan *level* 3 (*white box*)

Pengujian permainan *level* 3 merupakan proses dimana akan diuji algoritma *fuzzy* terhadap musuh dalam melakukan pencarian rute terpendek. Berikut ini merupakan pengujian algoritma *fuzzy* yang di terapkan pada permainan *level* 3 dengan menggunakan metode *white box*.

a. Algoritma Yang Akan Di Uji

Berikut ini merupakan algoritma *fuzzy* untuk *level* 3 :

```
1   for(int j=0; j <= 525; j += 100)
```

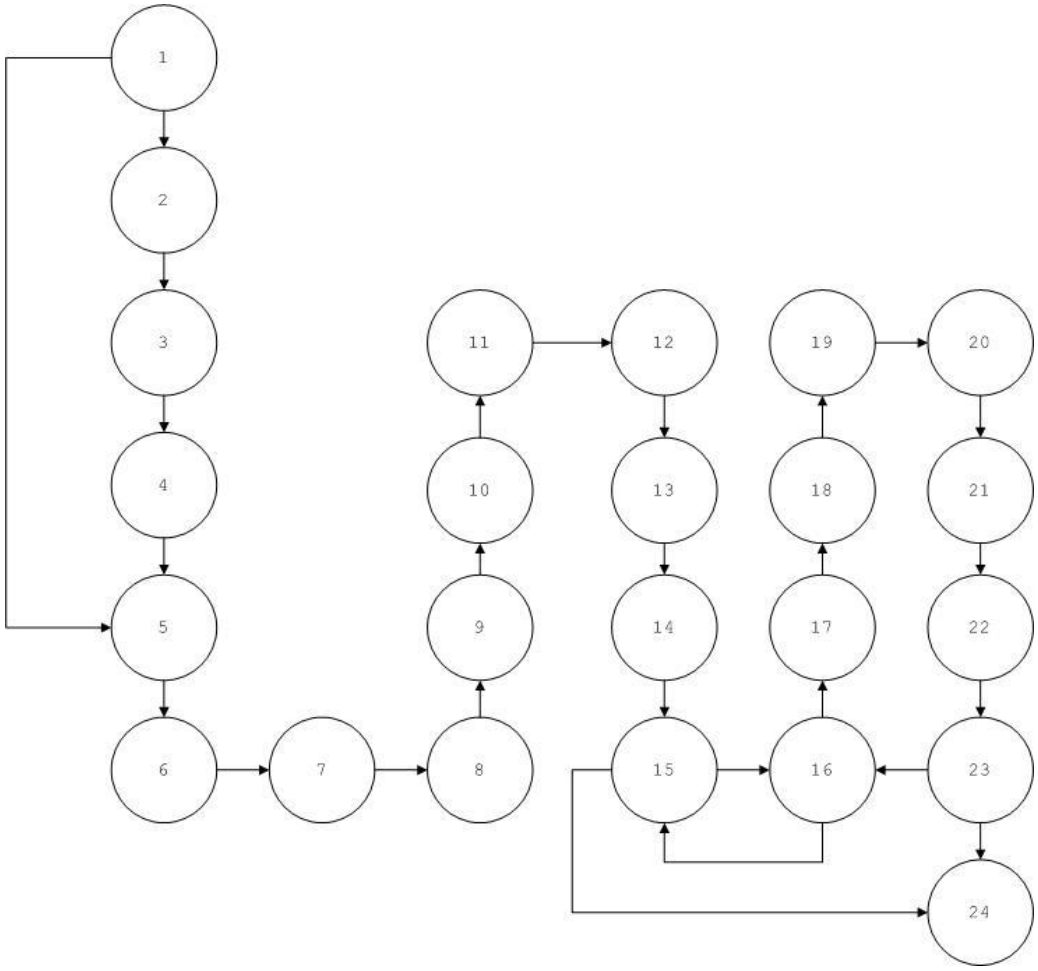
```
    {
2   if(j==150)
    {
    harga1 = hargafuzzy();
    ukuran1 = ukuranfuzzy();
    K_NPC1 = harga1 + ukuran1;
3   } else if(j==275)
    {
    harga2 = hargafuzzy();
    ukuran2 = ukuranfuzzy();
    K_NPC2 = harga2 + ukuran2;
4   } else if(j==375)
    {
    harga3 = hargafuzzy();
    ukuran3 = ukuranfuzzy();
    K_NPC3 = harga3 + ukuran3;
    }
5   }
6   if(K_NPC1 == 1 && K_NPC1 <= 150)
    {
    house[1] = bagus;
7   } else if(K_NPC1 <= 2 && K_NPC1 <= 350)
    {
    house[2] = sedang;
8   } else if(K_NPC1 <= 3 && K_NPC1 <= 550)
    {
    house[3] = mahal;
    }
9   if(K_NPC2 == 1 && K_NPC2 <= 150)
    {
    house[1] = bagus;
```

```
10  } else if(K_NPC2 <= 2 && K_NPC2 <= 350)
    {
    house[2] = sedang;
11  } else if(K_NPC2 <= 3 && K_NPC2 <= 550)
    {
    house[3] = mahal;
    }
12  if(K_NPC3 == 1 && K_NPC3 <= 150)
    {
    house[1] = bagus;
13  } else if(K_NPC3 <= 2 && K_NPC3 <= 350)
    {
    house[2] = sedang;
14  } else if(K_NPC3 <= 3 && K_NPC3 <= 550)
    {
    house[3] = mahal;
    }
15  for(int i=0; i<=3; i++)
    {
16  if(house[i] != null && house[i] != this.Tree())
    {
17  if(house[i] == bagus)
    {
    this.EAI();
18  } else if(house[i] != bagus)
    {
    this.Move();
    }
19  if(house[i] == sedang)
    {
    this.EAI();
```

```
20  } else if(house[i] != sedang)
    {
    this.Move();
    }
21  if(house[i] == mahal)
    {
    this.EAI();
22  } else if(house[i] != mahal)
    {
    this.Move();
    }
23  }
24  }
```

b. *Flow Graph* Permainan *Level* 3

Berikut ini merupakan *flow graph* permainan *level* 3 yang dapat dilihat pada gambar 4.6 :



Gambar 4. 6 *Flow Graph* Permainan *Level* 3

Keterangan :

= Menggambarkan Kondisi

→ = Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 27, N = 24

V(G) = E – N + 2

V(G) = 27 – 24 + 2

V(G) = 3 + 2 = 5

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1),(2),(3),(4),(5)

Path 2 = (1),(6),(7),(8),(9),(10),(11)

Path 3 = (1),(9),(10),(11),(12),(13),(14)

Path 4 = (1) ,(15),(16),(17),(18),(19),(20)

Path 5 = (1),(19),(20),(21),(22),(23),(24)

e. *Graph Matriks* Permainan *Level* 3

Tabel 4. 5 *Graph Matriks* Permainan *Level* 3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	N =(E-1)
1	1				1																				1
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								1	1
16															1		1								1
1																		1							0

7																							
18																		1					0
19																			1				0
20																				1			0
21																					1		0
22																						1	0
23															1								1	1
24																								0
SUM(E)+1																								4+1=

Keterangan :

3. Baris dan kolom merepresentasikan simpul
4. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Kesimpulan

Dari hasil pengujian permainan *level* 3 yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *path* telah dikerjakan, serta tidak terjadi pengulangan.

4.2.2 Pengujian *Beta*

Pengujian *beta* merupakan pengujian yang dilakukan secara objektif, dimana dilakukan pengujian secara langsung terhadap pengguna dengan menggunakan kuesioner mengenai kepuasan pengguna atas aplikasi yang telah dibangun. Adapun metode penilaian pengujian yang digunakan adalah metode kuantitatif berdasarkan data dari pengguna.

4.2.2.1 *Kuesioner*

Kuesioner disebarkan menggunakan teknik *Simple Random Sampling* dimana anggota *sample* dari populasi dilakukan secara acak tanpa memperhatikan strata yang ada dalam populasi. Kuesioner disebarkan kepada 30 orang dari umur

9 tahun ke atas, dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian penerapan sistem yang telah dibangun. Kuesioner terdiri dari 6 pernyataan dengan menggunakan skala 1 sampai 5 (skala likert).

Untuk mencari nilai persentase dari masing-masing jawaban kuesioner digunakan rumus skala likert sebagai berikut :

$$P = \frac{S}{skor\ ideal} \times 100$$

Keterangan :

P = Nilai persentase yang dicari

S = Jumlah frekuensi dikalikan dengan nilai yang di tetapkan jawaban

Skor ideal = Nilai tertinggi dikalikan dengan jumlah sampel

Berikut ini adalah hasil persentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan rumus diatas.

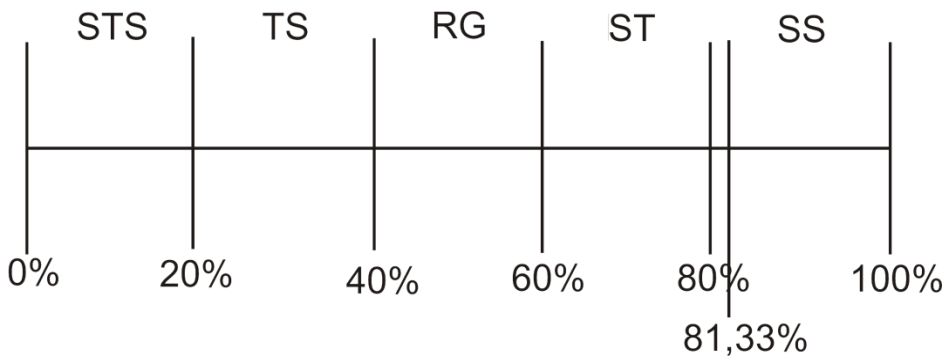
1. *Gameplay* pada *Game* ini unik.

Tabel 4. 6 Perhitungan persentase pertanyaan ke-1

Pertanyaan	Keterangan	Skor	Frekuensi	S
1	Sangat Setuju	5	10	50
	Setuju	4	18	72
	Ragu-ragu	3	2	6
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**122**

$$P = \frac{122}{150} \times 100\% = 81.33\%$$

Dapat disimpulkan bahwa penilaian terhadap *gameplay* yang unik pada permainan *Building of House* adalah 81,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.7.



Gambar 4. 7 Interpretasi Skor Pertanyaan ke-1

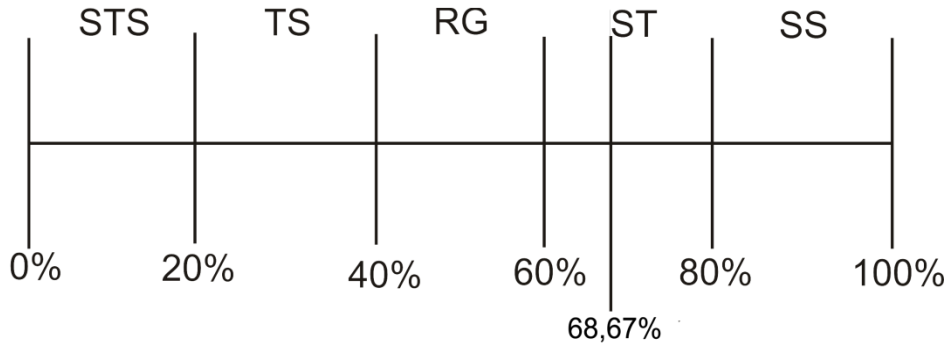
2. Sulit untuk menyelesaikan *Game* ini.

Tabel 4. 7 Perhitungan persentase pertanyaan ke-2

Pertanyaan	Keterangan	Skor	Frekuensi	S
2	Sangat Setuju	5	0	0
	Setuju	4	16	64
	Ragu-ragu	3	11	33
	Tidak Setuju	2	3	6
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**103**

$$P = \frac{103}{150} \times 100\% = 68.67\%$$

Dapat disimpulkan bahwa penilaian terhadap kecerdasan musuh pada permainan *Building of House* adalah 68,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.8.



Gambar 4. 8 Interpretasi Skor Pertanyaan ke-2

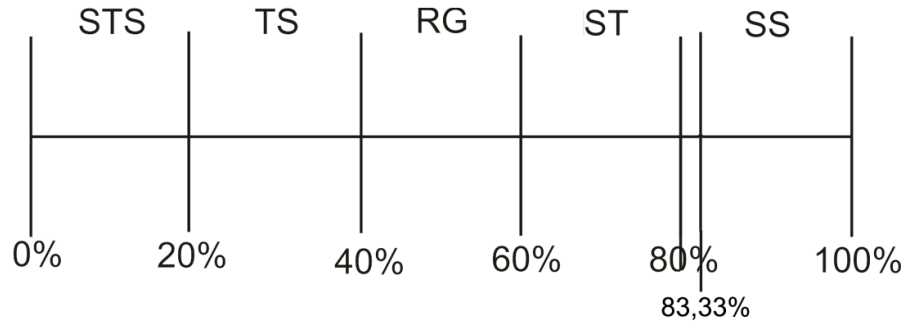
3. *Game* ini dapat menggambarkan simulasi pembuatan rumah.

Tabel 4. 8 Perhitungan persentase pertanyaan ke-3

Pertanyaan	Keterangan	Skor	Frekuensi	S
3	Sangat Setuju	5	9	45
	Setuju	4	17	68
	Ragu-ragu	3	4	12
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**125**

$$P = \frac{125}{150} \times 100\% = 83.33\%$$

Dapat disimpulkan bahwa penilaian terhadap simulasi pembuatan rumah pada permainan *Building of House* adalah 83,33% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.9.



Gambar 4. 9 Interpretasi Skor Pertanyaan ke-3

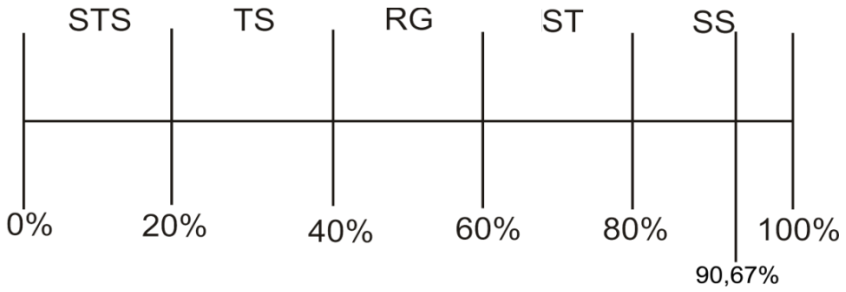
4. Skenario permainan yang di buat sudah menarik untuk di mainkan.

Tabel 4. 9 Perhitungan persentase pertanyaan ke-4

Pertanyaan	Keterangan	Skor	Frekuensi	S
4	Sangat Setuju	5	19	95
	Setuju	4	9	36
	Ragu-ragu	3	1	3
	Tidak Setuju	2	1	2
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**136**

$$P = \frac{136}{150} \times 100\% = 90.67\%$$

Dapat disimpulkan bahwa penilaian terhadap skenario *game* pada permainan *Building of House* adalah 90,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.10.



Gambar 4. 10 Interpretasi Skor Pertanyaan ke-4

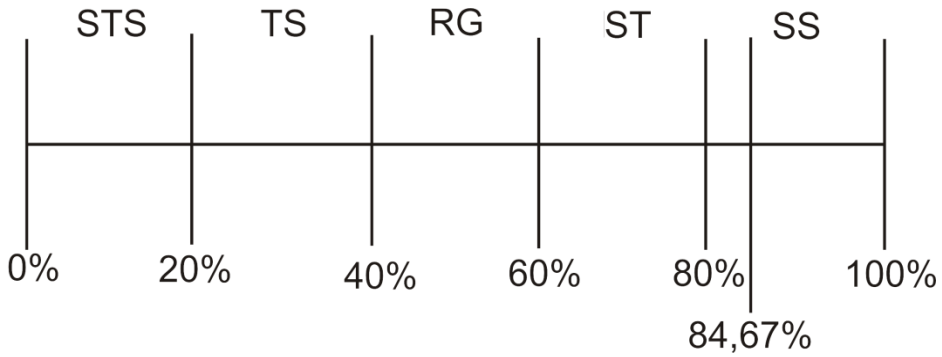
5. *Game* ini dapat menggambarkan simulasi penyewaan rumah terhadap pelanggan.

Tabel 4. 10 Perhitungan persentase pertanyaan ke-5

Pertanyaan	Keterangan	Skor	Frekuensi	S
5	Sangat Setuju	5	10	50
	Setuju	4	17	68
	Ragu-ragu	3	3	9
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**127**

$$P = \frac{127}{150} \times 100\% = 84.67\%$$

Dapat disimpulkan bahwa penilaian terhadap simulasi penyewaan rumah oleh pelanggan pada permainan *Building of House* adalah 84,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.11.



Gambar 4. 11 Interpretasi Skor Pertanyaan ke-5

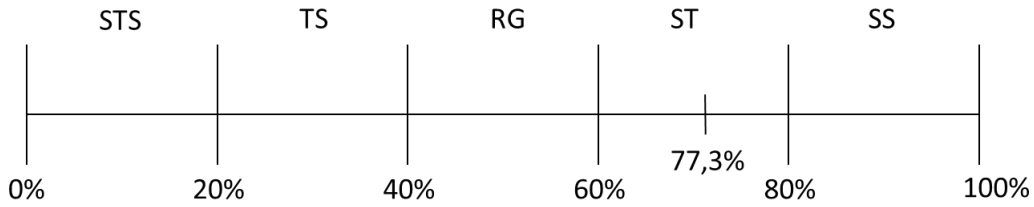
6. *Game* ini dapat dijadikan media pembelajaran.

Tabel 4. 11 Perhitungan persentase pertanyaan ke-6

Pertanyaan	Keterangan	Skor	Frekuensi	S
6	Sangat Setuju	5	6	30
	Setuju	4	14	56
	Ragu-ragu	3	10	30
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**116**

$$P = \frac{116}{150} \times 100\% = 77{,}3\%$$

Dapat disimpulkan bahwa penilaian terhadap sarana media pembelajaran pada permainan *Building of House* adalah 77,3% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.12.



Gambar 4. 12 Interpretasi Skor Pertanyaan ke-6