BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

Pada bab ini akan dilakukan implementasi dan pengujian sistem terhadap aplikasi permainan yang dibangun, yaitu permainan *The Island Colonized*. Tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan. Setelah tahap implementasi selesai, maka akan dilakukan tahap pengujian terhadap aplikasi permainan *The Island Colonized.*

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk menjalankan aplikasi permainan *The Island Colonized* sebagai berikut:

1. Processor Dual Core 1,0 Ghz
2. RAM 1 GB
3. Micro SD 4 GB

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi permainan *The Island Colonized* adalah :

1. Sistem operasi minimal Android 4.0
2. *Graphic card*

4.1.3 Implementasi Aplikasi

The Island Colonized merupakan aplikasi permainan yang di fokuskan untuk mengusir para penjajah, mencari bantuan, serta merebut *item*.

Permainan dikatakan *game over* apabila jumlah *lifepoint* berkurang menjadi 0 dan permainan dikatakan *mission complete* apabila pemain dapat menyelesaikan misi pada setiap *level*.

Aplikasi permainan *The Island Colonized* ini berbasis *android* dengan menggunakan construct 2. Aplikasi ini hanya dapat dijalankan pada android dengan operating system 4.0 ke atas.

4.1.4 Implementasi Antarmuka

Antarmuka dapat membuat pengguna merasa nyaman dan bertahan lama berada dalam aplikasi atau sebaliknya. Berikut ini merupakan antarmuka yang ada pada aplikasi permainan.

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 di atas merupakan menu utama pada *game The Island Colonized*. Terdapat 4 sub menu di dalamnya, yaitu menu Mainkan yang berfungsi untuk memulai permainan, menu Petunjuk yang berfungsi untuk melihat informasi cara memainkan permainan, menu Tentang berfungsi untuk

menampilkan info tentang aplikasi permainan dan menu Keluar yang berfungsi untuk keluar dari aplikasi.

2. Antarmuka Menu Mainkan

Saat pengguna memilih menu mainkan, maka antarmuka yang ditampilkan oleh sistem adalah : Pertama, antarmuka *intro 1* berupa cerita kekayaan tiap pulau. Selanjutnya akan masuk ke antarmuka *intro 2* berupa cerita perompak yang hendak menguasai dan merebut sumber daya alam di tiap pulau dan kemudian akan masuk pada antarmuka permainan. Jika seluruh misi telah terselesaikan maka akan menampilkan *mission complete* dan akan muncul simbol untuk masuk ke *level* berikutnya. Berikut ini merupakan tampilan dari antarmuka menu mainkan.



Gambar 4.2 Antarmuka *Intro 1*

Gambar 4.2 merupakan antarmuka *intro 1* yang terdapat pada aplikasi permainan *The Island Colonized*.



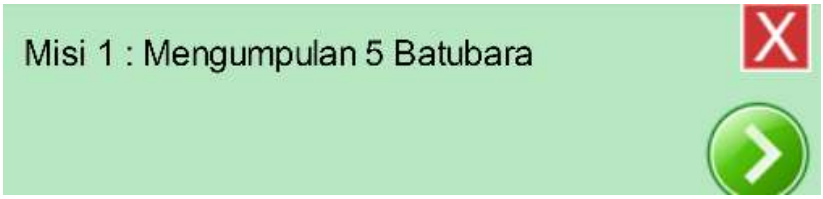
Gambar 4.3 Antarmuka *Intro 2*

Gambar 4.3 merupakan antarmuka *intro 2* yang terdapat pada aplikasi permainan *The Island Colonized*.



Gambar 4.4 Antarmuka Permainan *Level 1*

Gambar 4.4 merupakan antarmuka permainan yang terdapat pada *game The Island Colonized*. Gambar tersebut memperlihatkan posisi pemain ketika masuk ke dalam permainan.



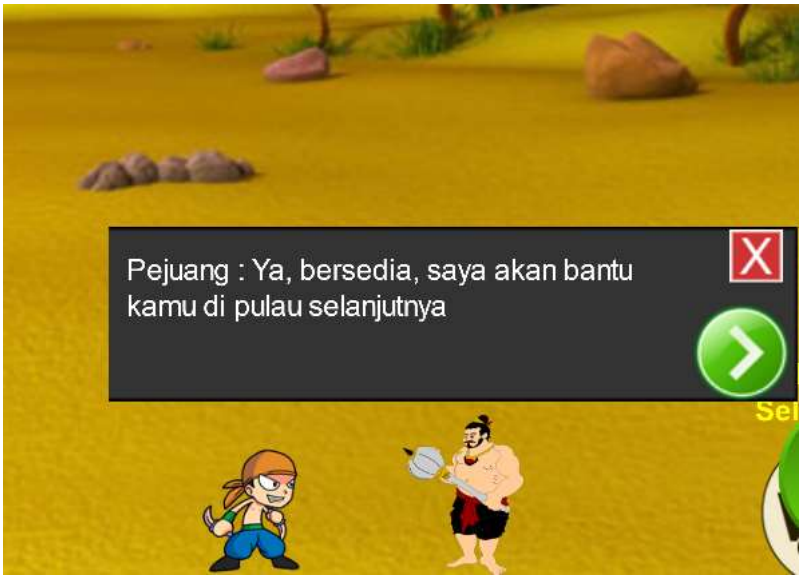
Gambar 4.5 Antarmuka Misi

Gambar 4.5 merupakan antarmuka misi yang terdapat pada *game The Island Colonized*. Gambar tersebut memperlihatkan misi yang harus diselesaikan dalam permainan.



Gambar 4.6 Antarmuka *Mission Complete*

Gambar 4.6 merupakan antarmuka *mission complete* yang terdapat pada *game The Island Colonized*. Gambar tersebut memperlihatkan misi telah selesai.



Gambar 4.7 Antarmuka Dialog Pejuang

Gambar 4.7 merupakan antarmuka ketika karakter lakon berdialog dengan karakter pejuang.



Gambar 4.8 Antarmuka Simbol *Level* Berikutnya

Gambar 4.8 merupakan antarmuka ketika karakter lakon akan melanjutkan ke *level berikutnya*.



Gambar 4.9 Antarmuka *Level 2*

Gambar 4.9 merupakan antarmuka setelah pemain menyelesaikan *level* pertama.



Gambar 4.10 Antarmuka *Level 2*

Gambar 4.10 merupakan antarmuka permainan yang terdapat pada *level 2 game The Island Colonized*. Gambar tersebut memperlihatkan pemain sudah merekrut karakter pejuang, posisi pemain ketika masuk ke dalam permainan.



Gambar 4.11 Antarmuka *Level 2*

Gambar 4.11 merupakan antarmuka setelah pemain menyelesaikan *level* kedua.



Gambar 4.12 Antarmuka *Level 3*

Gambar 4.12 merupakan antarmuka permainan yang terdapat pada *level 3 game The Island Colonized*. Gambar tersebut memperlihatkan pemain sudah merekrut karakter pejuang posisi pemain ketika masuk ke dalam permainan.



Gambar 4.13 Antarmuka *Level 3*

Gambar 4.13 merupakan antarmuka permainan yang terdapat pada *level* terakhir *game The Island Colonized*. Gambar tersebut memperlihatkan pemain saat akan menghadapi raja terakhir.

3. Antarmuka Menu Petunjuk

Saat pengguna membuka menu petunjuk maka akan ditampilkan informasi tentang bagaimana menjalankan aplikasi *The Island Colonized.* Berikut ini merupakan tampilan dari antarmuka menu Petunjuk :



Gambar 4.14 Antarmuka Petunjuk

Gambar 4.14 di atas merupakan sub menu Petunjuk pada *game The Island Colonized*.

4. Antarmuka Menu Tentang

Saat pengguna membuka menu Tentang, maka akan ditampilkan informasi untuk menampilkan info tentang aplikasi permainan. Berikut ini merupakan tampilan dari antarmuka menu Petunjuk :



Gambar 4.15 Antarmuka Tentang

4.2 Pengujian

Dalam melakukan pengujian terhadap aplikasi permainan *The Island Colonized* dilakukan dengan dua macam teknik, yaitu pengujian *black-box* dan pengujian *white-box*. Adapun pengujian *black box* berfokus pada menemukan kesalahan program, sedangkan pengujian *white box* berfokus kepada struktur internal (*source* code) program untuk mengetahui apakah masih terjadi *error*.

4.2.1 Rencana Pengujian

Rencana pengujian yang akan dilakukan pada aplikasi permainan *The Island Colonized* selengkapnya terlihat pada tabel di bawah.

Tabel 4.1 Rencana Pengujian Aplikasi Permainan *The Island Colonized*

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1.	Menu Mainkan	Klik menu Mainkan	*Black box*

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
2.	Menu Petunjuk	Klik menu Petunjuk	*Black box*
		Klik menu Kembali	*Black box*
3.	Menu *About*	Klik menu *About*	*Black box*
		Klik menu Kembali	*Black box*
4.	Menu Keluar	Klik menu Keluar	*Black box*
5.	Intro	Klik *Next* Intro 1	*Black box*
		Klik *Next* Intro 2	*Black box*
		Permainan Level 1	*Black box*
6.	Bergerak	Tekan *Button Left*	*Black box*
		Tekan *Button Right*	*Black box*
		Tekan *Button Up*	*Black box*
		Tekan *Button Down*	*Black box*
7.	Menyerang	Tekan *Button Attack*	*Black box*
8.	Mengambil Item	Mengarahkan karakter lakon hingga menyentuh item	*Black box*
9.	Merebut Item	Mengarahkan karakter lakon dengan musuh sehingga merebut item dari musuh	*Black box*
10.	Merekrut Pejuang	Mengarahkan karakter lakon dengan karakter pejuang sehingga karakter lakon bisa merekrut pejuang	*Black box*
11.	Menambah Energi	Mengarahkan karakter lakon hingga menyentuh icon energi	*Black box*
12.	Algortima A* (A *star*)	Pencarian rute atau jalan terbaik untuk NPC	*White Box*
13.	Algoritma *Fuzzy Logic*	Perilaku NPC	*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. Hasil pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada tabel 4.2.

Tabel 4.2 Skenario Pengujian *Black Box*

No	Kasus yang akan di uji	Skenario dan Hasil Uji		
		Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
1.	Menu Mainkan	Klik Menu Mainkan	Sistem menampilkan halaman *Intro*	[√] Berhasil [ ] Tidak Berhasil
2.	Menu Petunjuk	Klik Menu Petunjuk	Sistem menampilkan petunjuk berupa informasi seputar permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik Kembali	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil
3.	Menu About	Klik Menu About	Sistem menampilkan seputar tentang permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik Kembali	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil
4.	Menu Keluar	Klik Menu Keluar	Menutup atau Keluar dari aplikasi	[√] Berhasil [ ] Tidak Berhasil
5.	Intro	Klik Next Intro 1	Sistem menampilkan Intro 1	[√] Berhasil [ ] Tidak Berhasil
		Klik Next Intro 2	Sistem menampilkan Intro 2	[√] Berhasil [ ] Tidak Berhasil
		Permainan Level 1	Sistem menampilkan Permainan level 1	[√] Berhasil [ ] Tidak Berhasil
6.	Bergerak	Tekan *Button Left*	Karakter lakon bergerak kekiri	[√] Berhasil [ ] Tidak Berhasil
		Tekan *Button Right*	Karakter lakon bergerak kekanan	[√] Berhasil [ ] Tidak Berhasil
		Tekan *Button*	Karakter lakon	[√] Berhasil

No	Kasus yang akan di uji	Skenario dan Hasil Uji		
		Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
		Up	bergerak keatas	[ ] Tidak Berhasil
		Tekan *Button Down*	Karakter lakon bergerak kebawah	[√] Berhasil [ ] Tidak Berhasil
7.	Menyerang	Menekan *Button Attack*	Karaker lakon akan menyerang	[√] Berhasil [ ] Tidak Berhasil
8.	Mengambil item	Mengarahkan karakter hingga menyentuh *item*	Jumlah item pada permainan akan bertambah jika karakter lakon menyentuh item tersebut	[√] Berhasil [ ] Tidak Berhasil
9.	Merebut item	Mengarahkan karakter lakon dengan musuh sehingga merebut item dari musuh	Jumlah item pada permainan akan bertambah jika karakter lakon mengalahkan musuh dan menyentuh item tersebut	[√] Berhasil [ ] Tidak Berhasil
10.	Merekrut Pejuang	Mengarahkan karakter lakon dengan karakter pejuang sehingga karakter lakon bisa merekrut pejuang	System akan menampilkan dialog antara karakter lakon dan karakter pejuang	[√] Berhasil [ ] Tidak Berhasil
11.	Menambah Energi	Mengarahkan karakter lakon hingga menyentuh icon energi	*Bar* energi pada permainan akan bertambah jika karakter lakon menyentuh icon energi tersebut	[√] Berhasil [ ] Tidak Berhasil

4.2.1.3 Kasus dan Hasil Pengujian (White Box Testing)

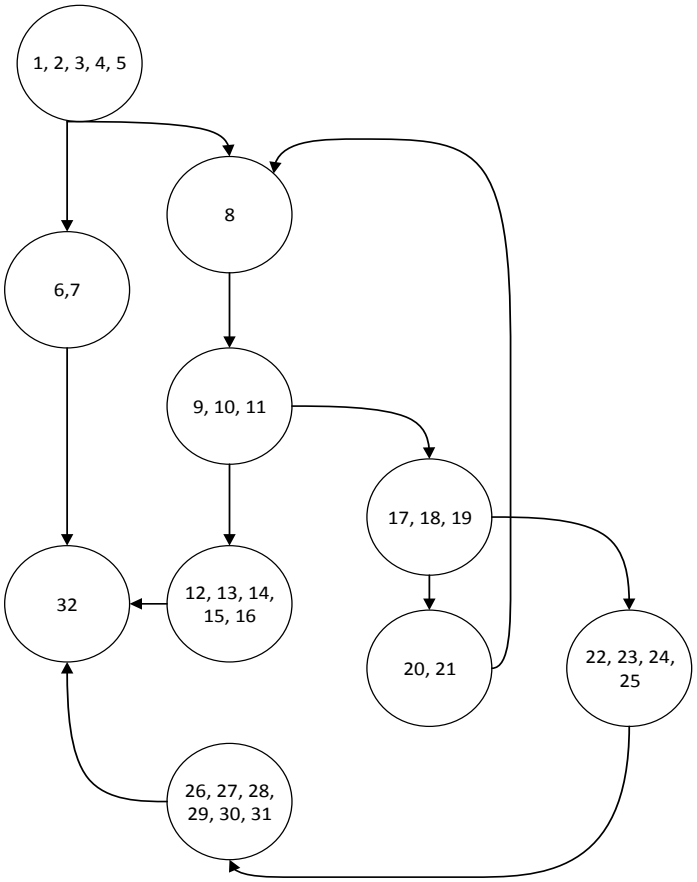
1. Pengujian A* (A *Star*)

Dibawah ini merupakan kasus algoritma A* (A *star*) untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *white box*, berdasarkan skenario pengujian aplikasi yang dijabarkan.

A. Algoritma

```
1   OpenList.Add (nodeAwal) ;
2       while(foundTarget==false) {
3                 checkingNode = GetSallestFValueNode () ;
4                 pindah checkingNode ke closelist
5              if (checkingNode == targetNode)
6                           then foundTarget = true;
7                       TraceBackPath (musuh);
8                       else for (var node : PathfindNode in
                                  checkingNode.dekat)
9    if (closedList.Contains (node) == false)
10    then if (openList.Contains (testing) ==
                        true)
11      then hitung G cost baru
12                 if (newGCoct < node.getGValue () )
13        then node.setParent (ceckingNode);
14                 node.setGValue (newGCost);
15        testing.CalculateFValue ();
16     endif
17               else testing.setParent (ceckingNode);
18                     node.setGValue (ceckingNode.getGValue ()
                                  +baseMovementCost);
19              node.CalculateFValue();
20                 if (openList.Count<limit)
21                 then AddToOpenlist (node);
22                  else var GetBiggestFValueNode : PathfindNode =
                              GetBiggestFValueNode ();
23             if (node.getFValue () <= GetBiggestFValueNode.getFValue
                                        ())
24                       then openList.Remove (GetBiggestFValueNode);
25                               AddToOpenList (node);
26                               endif
27                       end if
28              end if
29              endif
30    endfor
31     endif
32  endwhile
```

B. Flow Graph



Gambar 4.16 ***Flow Graph A Star***

Kerangan :

= Menggambarkan kondisi

= Menggambarkan aksi

C. ***Cylomatic Complexity*** **V(G)**

Cyclomatic complexity akan digunakan dalam menguji *path* yang nantinya akan digunakan untuk mencari jumlah *path* dalam satu jalur dan juga sebagai penentuan jalur atau *path* pencarian NPC terhadap *player*.

Berikut ini adalah perhitungan *Cyclomatic complexity* yang nantinya akan digunakan sebagai jumlah aksi pada *independent path.*

V(G) = E – N + 2

V(G) = 12 – 10 + 2

V(G) = 4

Di mana:

E = jumlah edge pada grafik alir

N = jumlah node pada grafik alir

D. Independent Path

Independent Path adalah jalur yang melintasi atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru. Berdasarkan *Cyclomatic complexity* maka terdapat 4 path, yang terdiri dari :

Path 1 = 1-2-3-4-5-8-9-10-11-12-13-14-15-16-32

Path 2 = 1-2-3-4-5-8-9-10-11-17-18-19-20-21-8-9-10-11-17-18-19-22-23-24-25-26-27-28-29-30-31-32

Path 3 = 1-2-3-4-5-8-9-10-11-17-18-19-22-23-24-25-26-27-28-29-30-31-32

Path 4 = 1-2-3-4-5-6-7-32

E. Graph Matrices

Berikut in adalah *graph matrices* dari algoritma A* Setelah *flow graph* diberi nomor dan *cyclomatic complexity* telah dihitung, selanjutnya dibuat tabel *graph matrics* yang dapat dilihat pada tabel 4.3.

Tabel 4.3 *Graph Matrices A**

N	1	2	3	4	5	6	7	8	9	10	N(E) -1
1		1	1								2-1=1
2										1	1-1=0
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		1-1=0
9										1	1-1=0
10											
										Sum(E)+1	3+1=4

Keterangan:

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

Setelah dilakukan pengujian, hasil yang didapat dapat dilihat pada tabel 4.4 :

Tabel 4.4 Hasil Uji *White Box* Penerapan Algoritma *A**

No Path	Node (n)	Hasil yang diharapkan	Hasil Sesuai Uji Kasus	Keterangan
1	1-2-3-4-5-8-9-10-11-12-13-14-15-16-32	Menghasilkan *rute* terpendek dari A*	Menghasilkan *rute* terpendek dari A*	[√] Alur terlewati [ ] Alur tidak terlewati
2	1-2-3-4-5-8-9-10-11-17-18-19-20-21-8-9-10-	Memeriksa apakah *node* tetangga dari	Memeriksa apakah *node* tetangga dari	[√] Alur terlewati [ ] Alur tidak terlewati

	11-17-18-19-22-23-24-25-26-27-28-29-30-31-32	*node* yang diperiksa masuk pada *closed list*	*node* yang diperiksa masuk pada *closed list*	
3	1-2-3-4-5-8-9-10-11-17-18-19-22-23-24-25-26-27-28-29-30-31-32	Memeriksa F(n) terkecil pada setiap *node* tetangga dari *node* yang diperiksa untuk dijadikan *node* selanjutnya yang akan menjadi *parent*	Memeriksa F(n) terkecil pada setiap *node* tetangga dari *node* yang diperiksa untuk dijadikan *node* selanjutnya yang akan menjadi *parent*	[√] Alur terlewati [ ] Alur tidak terlewati
4	1-2-3-4-5-6-7-32	*Node* yang diperiksa sama dengan *node* tujuan	*Node* yang diperiksa sama dengan *node* tujuan	[√] Alur terlewati [ ] Alur tidak terlewati

Kesimpulan:

Berdasarkan hasil pengujian pada tabel 4.4 maka dapat disimpulkan hasil uji kasus setiap *node* pada *independent path A** berjalan sesuai hasil yang diharapkan yaitu semua alur terlewati 1 kali.

2. Pengujian *Fuzzy Logic*

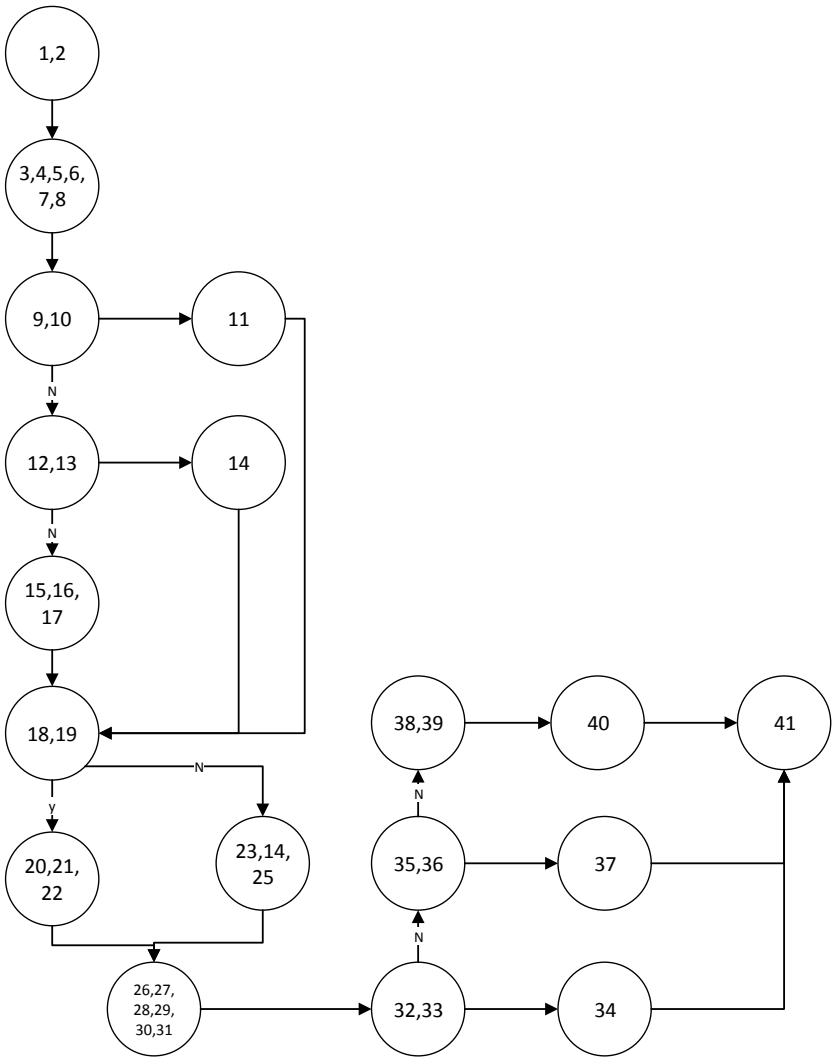
Dibawah ini merupakan kasus algoritma *Fuzzy Logic* untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *white box*, berdasarkan skenario pengujian aplikasi yang dijabarkan.

A. Algoritma

```
1  Function Fuzzy_Logic ()
2  {
3  var jarak = 0.0 ;
4  var kekuatan = 0.0 ;
5  var hitPoints = 0.0 ;
6  var posisiTarget = 0.0 ;
7  var posisiPlayer = 0.0 ;
8  var damage = 0 ;
9  if ( jarak >= 40 ) {
10                 jarak = kekuatan_lemah ;
11                 }
12         else if ( jarak >= 30 && <= 70 ) {
13                 jarak  = kekuatan_sedang ;
14                 }
15         else if ( jarak >= 60 && <= 100 ) {
16                 jarak = kekuatan_kuat ;
17                 }
18         jarak = posisiTarget - posisiPlayer ;
19         kekuatan = new kekuatan ;
20         if ( kekuatan <= 80 ) {
21                 kekuatan = kekuatan_lemah ;
22                 }
23         else ( kekuatan >= 20 && <=80 ) {
24                 kekuatan = kekuatan_kuat ;
25                 }
26         kekuatan  = hitPoints - damage ;
27         fungsi_trapesium = ( jarak, kekuatan ) ;
28         fungsi_segitiga = ( jarak, kekuatan ) ;
29         implikasi_nilai = ( jarak, kekuatan ) ;
30         periksa_rule = ( jarak, kekuatan ) ;
31         hitung_defuzzyfikasi = ( rule ) ;
32         if (hitung_defuzzyfikasi <= 10 ) {
33                 hitung_defuzzyfikasi = melarikan_diri ;
34                 }
35         else if ( hitung_defuzzyfikasi >= 10 &&  >=20 ) {
36                 hitung_defuzzyfikasi = bertahan
                   ;
37                 }
```

```
38          else if ( hitung_defuzzyfikasi >= 20 &&  <=30 ) {
39                  hitung_defuzzyfikasi = menyerang
                    ;
40          }
41  }
```

B. Flow Graph



Gambar 4.17 ***Flow Graph Fuzzy Logic***

Kerangan :

= Menggambarkan kondisi

= Menggambarkan aksi

C. *Cylomatic Complexity* V(G)

Cyclomatic complexity akan digunakan dalam menguji *path* yang nantinya akan digunakan untuk mencari jumlah *path* dalam satu jalur. Berikut ini adalah perhitungan *Cyclomatic complexity* yang diperoleh dari gambar yang nantinya akan digunakan sebagai jumlah aksi pada *independent path.*

V(G) = E – N + 2

V(G) = 22 – 18 + 2

V(G) = 6

Di mana:

E = jumlah edge pada grafik alir

N = jumlah node pada grafik alir

Jadi, *Cyclomatic complexity* untuk gambar 4.xx adalah 6

D. Independent Path

Independent Path adalah jalur yang melintasi atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru. Berdasarkan *Cyclomatic complexity* maka terdapat 6 path, yang terdiri dari :

Path 1 = 1-2-3-4-5-6-7-8-9-10-11-18-19-20-21-22-26-27-28-29-30-31-32-33-34-41

Path 2 = 1-2-3-4-5-6-7-8-9-10-11-18-19-23-24-25-26-27-28-29-30-31-32-33-35-36-38-39-40-41

Path 3 = 1-2-3-4-5-6-7-8-9-10-12-13-14-18-19-20-21-22-26-27-28-29-30-31-32-33-35-36-37-41

Path 4 = 1-2-3-4-5-6-7-8-9-10-12-13-14-18-19-23-24-25-26-27-28-29-30-31-32-33-35-36-38-39-40-41

Path 5 = 1-2-3-4-5-6-7-8-9-10-12-13-15-16-17-18-19-20-21-22-26-27-28-29-30-31-32-33-35-36-38-39-40-41

Path 6 = 1-2-3-4-5-6-7-8-9-10-12-13-15-16-17-18-19-23-24-25-26-27-28-29-30-31-32-33-34-41

E. Graph Matrices

Berikut ini adalah *graph matrices* dari algoritma *Fuzzy Logic* Setelah *flow graph* diberi nomor dan *cyclomatic complexity* telah dihitung, selanjutnya dibuat tabel *graph matrics* seperti pada tabel 4.5 berikut :

Tabel 4.5 *Graph Matrices Fuzzy Logic*

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

Keterangan:

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

Setelah dilakukan pengujian, hasil yang didapat dapat dilihat pada tabel 4.6 :

Tabel 4.6 Hasil Uji *White Box* Penerapan *Fuzzy Logic*

Masukan	Hasil yang diharapkan	Hasil Sesuai Uji Kasus	Keterangan
Hasil *defuzzy* <= 10 NPC melarikan diri menjauhi karakter lakon	NPC musuh melarikan diri menjauhi karakter lakon	NPC musuh melarikan diri menjauhi karakter lakon	[√] Berhasil [ ] Tidak Berhasil
Hasil *defuzzy* > 10 dan <= 20 NPC bertahan	NPC musuh bertahan terhadap karakter lakon	NPC musuh bertahan terhadap karakter lakon	[√] Berhasil [ ] Tidak Berhasil
Hasil *defuzzy* > 20 dan <= 30 NPC menyerang	NPC musuh menyerang karakter lakon	NPC musuh menyerang karakter lakon	[√] Berhasil [ ] Tidak Berhasil

Kesimpulan:

Berdasarkan hasil pengujian pada tabel 4.6, dapat disimpulkan bahwa sistem sudah dapat berjalan sesuai dengan yang diharapkan. Jadi, secara fungsional sistem sudah dapat menghasilkan keluaran yang diharapkan.

4.2.2 *Kuesioner*

Kuesioner dilakukan untuk mengetahui ketertarikan siswa terhadap game yang dibangun, meperkenalkan danpak global warming, bagian pohon beserta fungsinya. Pembuatan Kuesioner menggunakan metode skala likert, dimana masing-masing pertanyaan mempunyai pilihan sangat setuju, setuju, ragu-ragu,

tidak setuju, dan sangat tidak setuju. Masing-masing pilihan tersebut diberi poin (5 sampai 1). Berikut ini adalah daftar pertanyaan Kuesioner yang diberikan terhadap 50 orang responden, serta hasil yang didapat setelah nilainya dihitung :

1. Apakah tampilan permainan *The Island Colonized* secara umum sudah baik ?
   Hasil kuesioner untuk pernyataan ke-1 dapat dilihat pada tabel 4.7.

Tabel 4.7 Hasil Kuesioner Tampilan Aplikasi Permainan

Pernyataan	Jawaban	Skor	Responden	S
1	Sangat Setuju	5	35	175
	Setuju	4	15	60
	Ragu-Ragu	3	0	0
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**50**	**235**

2. Apakah alur permainan *The Island Colonized* mudah di mengerti ?
   Hasil kuesioner untuk pernyataan ke-2 dapat dilihat pada tabel 4.8.

Tabel 4.8 Hasil Kuesioner Alur Permainan

Pernyataan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	31	155
	Setuju	4	15	60
	Ragu-Ragu	3	4	12
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**50**	**227**

3. Apakah dengan menambah tingkat kesulitan (musuh mengejar karakter utama) membuat *game* lebih menantang ?

   Hasil kuesioner untuk pernyataan ke-3 dapat dilihat pada tabel 4.9.

Tabel 4.9 Hasil Kuesioner Menambah Tingkat Kesulitan

Pernyataan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	30	150
	Setuju	4	20	80
	Ragu-Ragu	3	0	0
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**50**	**230**

4. Apakah perilaku musuh yang lebih pintar ketika berhadapan karakter utama membuat musuh dalam permainan *The Island Colonized* lebih sulit untuk dikalahkan?

   Hasil kuesioner untuk pernyataan ke-4 dapat dilihat pada tabel 4.10.

Tabel 4.10 Hasil Kuesioner Perilaku Musuh yang Lebih Pintar

Pernyataan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	33	165
	Setuju	4	17	68
	Ragu-Ragu	3	0	0
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**50**	**233**

5. Apakah anda bersedia untuk memainkan kembali aplikasi permainan *The Island Colonized* ?

   Hasil kuesioner untuk pernyataan ke-5 dapat dilihat pada tabel 4.11.

Tabel 4.11 Hasil Kuesioner Memainkan Kembali Permainan

Pernyataan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	23	115
	Setuju	4	21	84
	Ragu-Ragu	3	6	18
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**50**	**217**

Setelah dikalikan, poin tersebut dijumlahkan dan kemudian dihitung persentasenya dengan membagi jumlah oleh total kemungkinan nilai terbesar. Total kemungkinan nilai terbesar adalah kemungkinan poin tertinggi dikali jumlah responden.

Tabel 4.12 Hasil Perhitungan *Kuesioner*

	Sangat Setuju	Setuju	Ragu	Tidak Setuju	Sangat Tidak Setuju	*Persentase*	Kesimpulan
Q1	175	60	0	0	0	94.00%	Sangat Kuat
Q2	155	60	12	0	0	90.80%	Sangat Kuat
Q3	150	80	0	0	0	92.00%	Sangat Kuat
Q4	165	68	0	0	0	93.20%	Sangat Kuat
Q5	115	84	18	0	0	86.80%	Sangat Kuat

Kriteria kesimpulan hasil tersebut adalah sebagai berikut :

1. 0% - 20% : Sangat Lemah

2. 21% - 40% : Lemah

3. 41% - 60% : Cukup

4. 61% - 80% : Kuat

5. 81% - 100% : Sangat Kuat

Berdasarkan hasil yang didapat, maka kesimpulannya adalah sebagai berikut :

1. Tampilan *game* sangat menarik

2. Alur *game* sangat mudah dimengerti

3. *Game* sangat menantang dengan bertambahnya tingkat kesulitan

4. Musuh dalam *game* tidak mudah dikalahkan karena perilakunya lebih pintar

5. Responden sangat tertarik untuk memainkan *game* kembali.

4.2.2.1 *Kesimpulan Pengujian Kuesioner*

Data Berdasarkan hasil pengujian *beta kuesioner* yang dibagikan pada 50 responden menyatakan 94% bahwa tampilan *game* sangat menarik, 90.8 % alur *game* sangat mudah untuk diengerti, 92 % responden menyatakan *game* sangat menantang dengan bertambahnya tingkat kesulitan, 93.2 % menyatakan musuh dalam *game* tidak mudah dikalahkan karena perilakunya lebih pintar, dan 86.8 responden sangat tertarik untuk memainkan *game The Island Colonized* kembali. Oleh karena itu dapat disimpulkan bahwa tampilan *game* sangat menarik, alur permainan sangat mudah dimengerti, *game* lebih menantang, musuh lebih sulit dikalahkan, dan responden tertarik untuk memainkan *game The Island Colonized* kembali.