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. Aplikasi ini diimplementasikan dengan nama *The Tracker* dengan *file extension*-nya *.exe.*

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	Spesifikasi
Processor	Dula Core 1,80 Ghz
Ram	1Gb
VGA	512
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 7 Ultimate 32bit
Tools	Unity 4.0

4.1.3 Implementasi Antarmuka

Antarmuka dapat membuat pengguna merasa nyaman dan bertahan lama berada dalam aplikasi atau sebaliknya. Berikut ini merupakan antarmuka yang ada pada aplikasi *game The Tracker* :

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 The Tracker*. Terdapat 3 (tiga) sub menu di dalamnya, yaitu menu mulai yang berfungsi untuk memulai permainan, menu petunjuk yang berfungsi untuk melihat *controller* yang digunakan didalam *game*, menu keluar yang berfungsi untuk yang berfungsi untuk keluar dari aplikasi.

2. Antarmuka Menu Mulai

Saat pengguna memilih menu mulai, maka antarmuka yang ditampilkan oleh sistem adalah; Pertama, antarmuka *introscene* berupa *vidio* cerita awal dari *game*. Selanjutnya akan masuk ke antarmuka informasi misi dan kemudian akan masuk pada antarmuka permainan. Jika seluruh misi telah terselesaikan maka akan masuk ke antarmuka *endingscene* berupa vidio akhir cerita *game*. Berikut ini merupakan tampilan dari antarmuka menu mulai :



Gambar 4.2 Antarmuka *Introscene level* pertama

Gambar 4.2 merupakan antarmuka *introscene video level* pertama yang terdapat pada *game The Tracker*.



Gambar 4.3 Antarmuka Informasi misi *Level* pertama

Gambar 4.3 merupakan antarmuka informasi misi *level* pertama yang terdapat pada *game The Tracker*.



Gambar 4.4 Antarmuka Permainan *level* pertama

Gambar 4.4 merupakan antarmuka permainan *level* pertama yang terdapat pada *game The Tracker*.



Gambar 4.5 Antarmuka *Introscene level* kedua

Gambar 4.5 merupakan antarmuka *introscene video level* kedua yang terdapat pada *game The Tracker*.



Gambar 4.6 Antarmuka Informasi misi *level* kedua

Gambar 4.6 merupakan antarmuka Informasi misi *level* kedua yang terdapat pada *game The Tracker.*



Gambar 4.7 merupakan antarmuka permainan *level* kedua

Gambar 4.7 merupakan antarmuka permainan *level* kedua yang terdapat pada *game The Tracker.*



Gambar 4.8 merupakan antarmuka *Introscene video level* ketiga

Gambar 4.8 merupakan antarmuka *Introscene video level* ketiga yang terdapat pada *game The Tracker*.



Gambar 4.9 merupakan antarmuka inforamsi misi *level* ketiga

Gambar 4.9 merupakan antarmuka informasi misi *level* ketiga yang terdapat pada *game The Tracker*.



Gambar 4.10 merupakan antarmuka permainan *level* ketiga

Gambar 4.10 merupakan antarmuka permainan *level* ketiga yang terdapat pada *game The Tracker.*

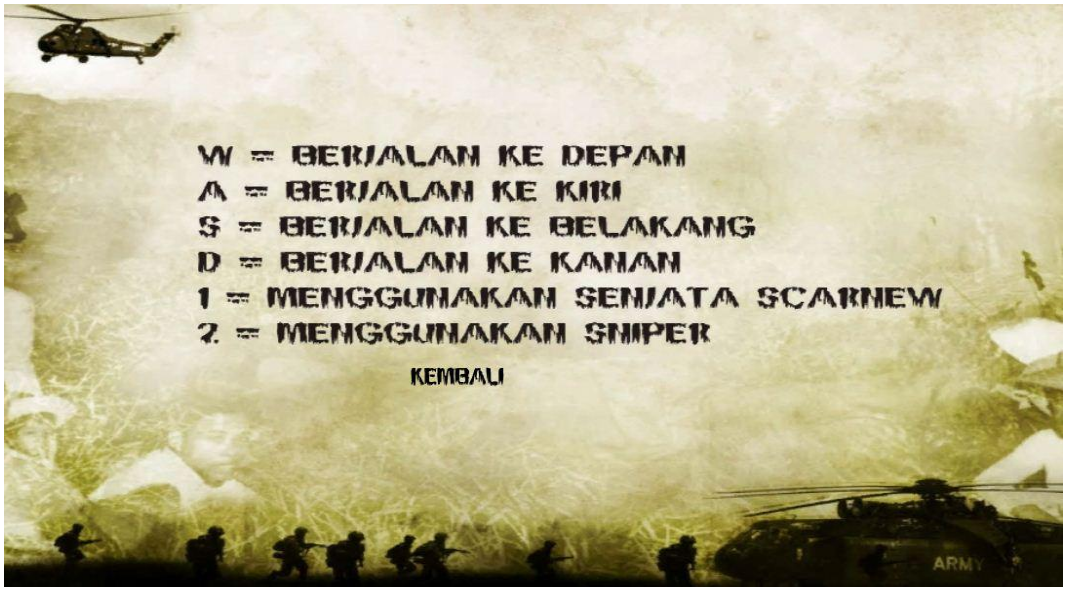


Gambar 4.11 Antarmuka *endingscene*

Gambar 4.11 merupakan antarmuka *endingscene video* yang terdapat pada *game The Tracker*.

3. Antarmuka Menu Petunjuk

Saat pengguna memilih menu Petunjuk, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka *controller* yang digunakan dalam *game The Tracker*. Gambar 4.12 merupakan tampilan dari antarmuka menu Petunjuk :



Gambar 4.12 Antarmuka Menu Petunjuk

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 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 testing* tidak terlalu memperhatikan struktur internal (*source code*) dari program, tetapi lebih kepada menemukan kesalahan program. Pada *game The Tracker* digunakan pengujian *black box testing*. Cara pengujian dilakukan melalui dua tahap, yaitu pengujian *alpha* (fungsional) dan pengujian *beta*.

4.2.1 Pengujian *Alpha*

Pengujian *Alpha* merupakan pengujian fungsional yang digunakan untuk menguji sistem yang telah dibuat dengan metode pengujian. Pengujian *alpha* dilakukan dengan menggunakan metode *white box* dan *black box*. Pengujian *black*

box berfokus pada persyaratan *fungsional* perangkat lunak. Skenario pengujian dari *game The Tracker* dapat dilihat pada *table* 4.3.

Tabel 4. 3 Skenario pengujian *Alpha*

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1.	Menu Utama	Memilih menu Mulai	*Black box*
		Memilih menu Petunjuk	*Black box*
		Memilih menu Keluar	*Black box*
2.	*Scene video*	Tekan tombol spasi	*Black box*
3.	Informasi misi	Memilih tombol mainkan	*Black box*
4.	*Game Over*	Memilih tombol kembali	*Black box*
5.	*Mission success*	Memilih tombol kembali	*Black box*
6.	Gerakan karakter	Tekan tombol W *keyboard*	*Black box*
		Tekan tombol A *keyboard*	*Black box*
		Tekan tombol S *keyboard*	*Black box*
		Tekan tombol D *keyboard*	*Black box*
		Tekan tombol spasi	*Black box*
		Tekan tombol W+Shift *keyboard*	*Black box*
		Tekan tombol A+Shift *keyboard*	*Black box*
		Tekan tombol D+Shift *keyboard*	*Black box*
		Memilih kiri *mouse*	*Black box*
		Tekan tombol 1 *keyboard*	*Black box*
		Tekan tombol 2 *keyboard*	*Black box*
		Tekan tombol esc *keyboard*	*Black box*
7.	Algoritma	A* (A *Star*)	*White box*
		Fuzzy Logic	*White 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 menu utama dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada table 4.2.

Tabel 4. 4 Pengujian menu utama

Kasus yang akan di uji	Skenario dan Hasil Uji		
	Skenario Uji	Hasil yang Diharapkan	Kesimpulan
Menu Utama	Memilih tombol menu mulai	Sistem menampilkan halaman *Intro scene video* dan informasi misi	[√] Berhasil [ ] Tidak Berhasil
	Memilih Menu Petunjuk	Sistem menampilkan halaman menu Petunjuk	[√] Berhasil [ ] Tidak Berhasil
	Memilih Menu Keluar	Menutup atau Keluar dari aplikasi	[√] Berhasil [ ] Tidak Berhasil

2. Pengujian *Scene video* dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada table 4.3.

Tabel 4. 5 Pengujian *Scene video*

Kasus yang akan di uji	Skenario dan Hasil Uji		
	Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
Tombol spasi	Menekan tombol spasi pada *scene video*	Menghentikan *scene video* dan masuk ke Informasi misi permainan	[√] Berhasil [ ] Tidak Berhasil

3. Pengujian Informasi misi dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada table 4.4.

Tabel 4. 6 Pengujian Informasi misi

Kasus yang akan di uji	Skenario dan Hasil Uji		
	Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
Tombol mainkan	Menekan tombol mainkan pada informasi misi	Menjalankan permainan	[√] Berhasil [ ] Tidak Berhasil

4. Pengujian *Game over* dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada table 4.5.

Tabel 4. 7 Pengujian *Game over*

Kasus yang akan di uji	Skenario dan Hasil Uji		
	Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
tombol kembali	Menekan tombol kembali pada *game over*	Menampilkan kembali halaman *main menu*	[√] Berhasil [ ] Tidak Berhasil

5. Pengujian *mission success* dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada table 4.6.

Tabel 4. 8 Pengujian *mission success*

Kasus yang akan di uji	Skenario dan Hasil Uji		
	Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
tombol kembali	Menekan tombol kembali pada *mission success*	Menampilkan kembali halaman *main menu*	[√] Berhasil [ ] Tidak Berhasil

6. Pengujian Gerakan karakter dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada table 4.7.

Tabel 4. 9 Pengujian Gerakan karakter

Kasus yang akan di uji	Skenario dan Hasil Uji		
	Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
Permainan	Tekan tombol W *keyboard*	Karakter pemain berjalan kedepan	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol A *keyboard*	Karakter pemain berjalan kekiri	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol S *keyboard*	Karakter pemain mundur kebelakang	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol D *keyboard*	Karakter pemain berjalan kekanan	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol W+Shift *keyboard*	Karakter pemain berlari kedepan	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol A+Shift *keyboard*	Karakter pemain berlari kekiri	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol D+Shift *keyboard*	Karakter pemain berlari kekanan	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol spasi	Karakter pemain melompat	[√] Berhasil [ ] Tidak Berhasil
	Memilih kiri *mouse*	Karakter pemain menembak	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol 1 *keyboard*	Karakter memakai senjata *Scarnew*	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol 2 *keyboard*	Karakter memakai senjata *Sniper*	[√] Berhasil [ ] Tidak Berhasil
	Tekan tombol esc *keyboard*	memberhentikan permainan	[√] Berhasil [ ] Tidak Berhasil

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

1. Pengujian *White Box* Algoritma A*

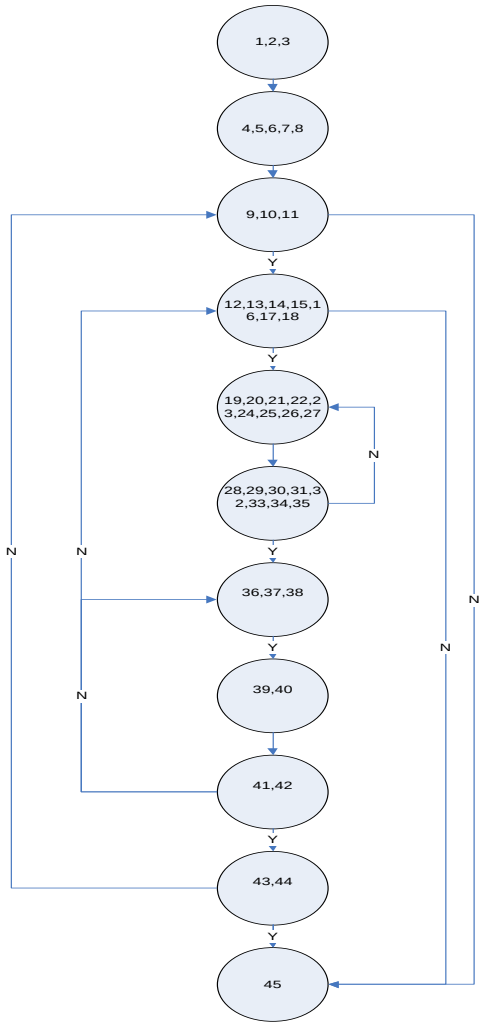
Berikut adalah *sourcecode* algoritma A* pada *game The Tracker* yang penerapannya menggunakan bahasa C#:

```
1.  public System.Type[] graphTypes
2.  get {
3.  return astarData.graphTypes;}
4.  public Matrix translate (float _X, float _Y, float _Z){
5.  m[12] = _X;
6.  m[13] = _Y;
7.  m[14] = _Z;
8.  return this;}                    ///class astarmath
9.  public virtual int[] InitialOpen (BinaryHeapM open,
    Int3
    targetPosition, Int3 position, Path path, bool doOpen){
10. return BaseInitialOpen
11. (open,targetPosition,position,path,doOpen);} clas node
12. public void Close () {
13. if (readerStream != null) {
14. readerStream.Close ();}
15. if (writerStream != null) {
16. if (anchors != null) {
```

```
17. SerializeAnchors ();}
18. writerStream.Close ();}}       ///clas astar serialize
19. public int[]public int[] BaseInitialOpen (BinaryHeapM
    open, Int3 targetPosition, Int3 position, Path path,
    bool doOpen) {
20. if (doOpen) {
21. connectionCosts = costs;}
22. if (connectionCosts == null) {
23. return null;}
24. int[] costs = connectionCosts;
25. connectionCosts = new int[connectionCosts.Length];
26. for (int i=0;i<connectionCosts.Length;i++) {
27. connectionCosts[i] = (connections[i].position-
    position).costMagnitude;}}
28. if (!doOpen) {
29. for (int i=0;i<connectionCosts.Length;i++) {
30. Node other = connections[i];
31. if (other.connections != null) {
32. for (int q = 0;q < other.connections.Length;q++) {
33. if (other.connections[q] == this) {
34. other.connectionCosts[q] = connectionCosts[i];
35. break;}}}}}
36. if (connections[i].walkable && connections[i].area !=
    area) {
37. stack.Push (connections[i]);
38. connections[i].area = area;}
39. if (doOpen) {
40. nodeR.g = nodeR.parent.g+nodeR.cost+penalty +
    nodeRunData.path.GetTagPenalty(tags);}
41. public virtual void UpdateAllG (NodeRun nodeR,
    NodeRunData nodeRunData) {
42. BaseUpdateAllG (nodeR, nodeRunData);} ////class node
43. if (Open == null){
44. return null;}
45. return path;
```

a. *Flow Graph* Algoritma A*

Berdasarkan *sourcecode* di atas maka *flow graph* algoritma A* adalah dapat dilihat pada gambar 4.13 :



Gambar 4.13 *Flow Graph* Algoritma A*

Keterangan:

= Menggambarkan jumlah edges.

= Menggambarkan jumlah node/simpul.

Dari Gambar 4.13 dapat dihitung *cyclomatic complexity* sebagai berikut :

1) V(G) = E – N + 2
2) V(G) = 16 – 11 + 2
3) V(G) = 5 + 2 = 7

Keterangan : E = jumlah edges/panah, N = jumlah node/simpul

Jadi, *cyclomatic complexity* untuk gambar 4.13 adalah 5. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 5 *path* yang terdiri dari :

1) *Path* 1 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45

   untuk melakukan pencarian rute ketika open list tidak kosong, dan menemukan tujuan.
2) *Path* 2 = 1-2-3-4-5-6-7-8-9-10-11-45

   Untuk menentukan rute tujuan yang berada di openlist.
3) *Path* 3 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-45

   Mendapatkan rute yang di cari dengan current node=goal.
4) *Path* 4 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45

   Pencarian rute selanjutnya tidak dapat dilalui collison atau unwalkable, buka kembali openlist.
5) *Path* 5 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45

   Masukan kembali semua neighbor node ke dalam open dan hitung kembali F,G,H selanjutnya tentukan current node dan nilai F,G,H masuk kedalam open.
6) *Path* 6 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45

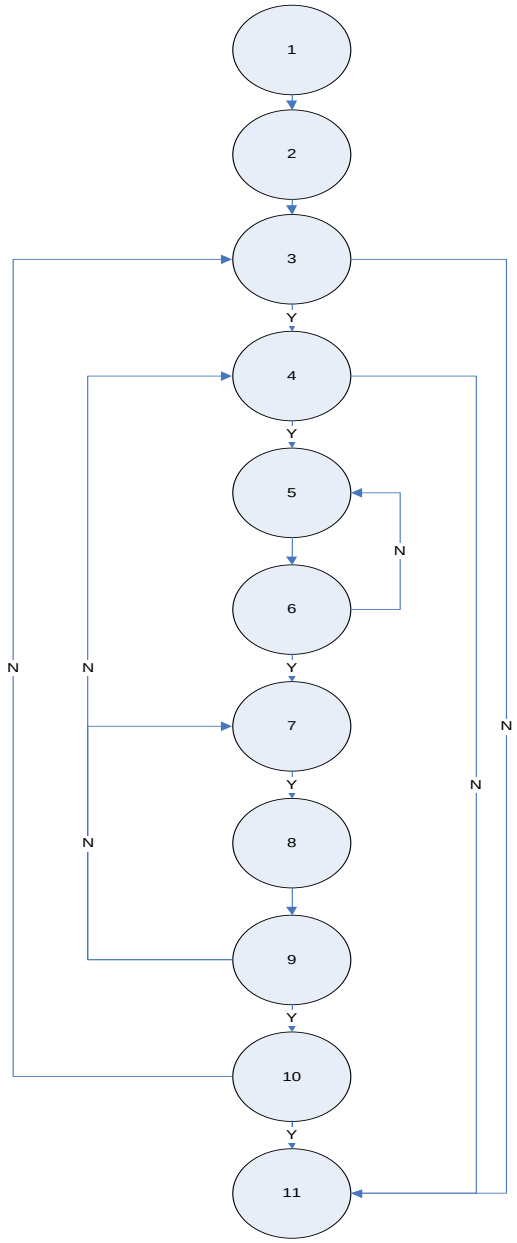
   Mendapatkan nilai F yang lebih kecil dari F sebelumnya cek ulang nilai F,G,H.
7) *Path* 7 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-

44-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45

Tidak ada node di dalam open cek kembali.

b. *Graph Matriks* Algoritma A*

Penomoran ulang *flow graph* atau penyederhanaan berdasarkan kondisi atau simpul sebagai berikut.



Gambar 4.14 Penomoran ulang *Flow Graph* A*

c. *Graph Matriks* Algoritma A*

Tabel 4. 10 *Graph Matriks* Algoritma A*

N	1	2	3	4	5	6	7	8	9	10	11	N(E)-1
1		1										1-1=0
2			1									1-1=0
3				1							1	2-1=1
4					1						1	2-1=1
5						1						1-1=0
6					1		1					2-1=1
7				1				1				2-1=1
8									1			2-1=1
9				1						1		2-1=1
10			1								1	2-1=1
11												1-1=0
Sum(E)+1												6+1=7

Keterangan:

1. Baris dan kolom merepresentasikan simpul.
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul. yang dibuat sesuai alur logika dari program tersebut. penggunaan kondisi memengaruhi jumlah node dan edge dari *flow graph* tersebut.
    a) Baris 1 dan kolom 2 = merepresentasikan adanya keterhubungan antar node/simpul 1 dan node/simpul 2 dari *flow graph*.
    b) Baris 2 dan kolom 3 = merepresentasikan adanya keterhubungan antar node/simpul 2 dan node/simpul 3 dari *flow graph*.
    c) Baris 3 dan kolom 4 dan 11 = merepresentasikan adanya keterhubungan antar node/simpul 3 dan node/simpul 4 dan 11 dari *flow graph*.
    d) Baris 4 dan kolom 5 dan 11 = merepresentasikan adanya keterhubungan antar node/simpul 4 dan node/simpul 5 dan 11 dari *flow graph*.
    e) Baris 5 dan kolom 6 = merepresentasikan adanya keterhubungan antar node/simpul 5 dan node/simpul 6 dari *flow graph*.
    f) Baris 6 dan kolom 5 dan 7 = merepresentasikan adanya keterhubungan antar node/simpul 6 dan node/simpul 5 dan 7 dari *flow graph*.
    g) Baris 7 dan kolom 4 dan 8 = merepresentasikan adanya keterhubungan antar node/simpul 7 dan node/simpul 4 dan 8 dari *flow graph*.

h) Baris 8 dan kolom 9 = merepresentasikan adanya keterhubungan antar node/simpul 8 dan node/simpul 9 dari *flow graph*.

i) Baris 9 dan kolom 4 dan 10 = merepresentasikan adanya keterhubungan antar node/simpul 9 dan node/simpul 4 dan 10 dari *flow graph*.

j) Baris 10 dan kolom 3 dan 11 = merepresentasikan adanya keterhubungan antar node/simpul 10 dan node/simpul 33 dan 11 dari *flow graph*.

e. Hasil Uji Algoritma A*

Tabel 4. 11 Hasil Uji Algoritma A*

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

1	-public System.Type[] graphTypes
	-get {
	-return astarData.graphTypes;}
	-public Matrix translate (float _X, float _Y, float _Z){
	-m[12] = _X;
	-m[13] = _Y;
	-m[14] = _Z;
	-return this;}
	astarmath
	-public virtual int[] InitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen){
	-return BaseInitialOpen -(open,targetPosition,position,path,doOpen);}
	clas node
	-public void Close ()
	{
	-if (readerStream != null) {
	-readerStream.Close ();}
	-if (writerStream != null) {
	-if (anchors != null) {
	SerializeAnchors ();}
	-writerStream.Close
		Untuk melakukan pencarian rute ketika open list tidak kosong, dan menemukan tujuan	Mendapatkan rute tujuan	[□]dikerjakan
				[ ] tidak dikerjakan

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan
	();}}clas astar serialize -public int[]public int[] BaseInitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen) { -if (doOpen) { -connectionCosts = costs;} -if (connectionCosts == null) { -return null;} -int[] costs = connectionCosts; -connectionCosts = new int[connectionCosts.Length]; -for (int i=0;i<connectionCosts.Length;i++) { -connectionCosts[i] = (connections[i].position-position).costMagnitude;}} -if (!doOpen) { -for (int i=0;i<connectionCosts.Length;i++) { -Node other = connections[i]; -if (other.connections != null) { -for (int q = 0;q < other.connections.Length;q++) { -if (other.connections[q] == this) { -other.connectionCosts[q] = connectionCosts[i]; -break;}}}}} -if (connections[i].walkable && connections[i].area != area) { -stack.Push (connections[i]); -connections[i].area =			

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	area;}
	-if (doOpen) {
	-nodeR.g =
	nodeR.parent.g+nodeR.c
	ost+penalty +
	nodeRunData.path.GetTa
	gPenalty(tags);}
	-public virtual void
	UpdateAllG (NodeRun
	nodeR, NodeRunData
	nodeRunData) {
	-BaseUpdateAllG
	(nodeR, nodeRunData);}
	class node
	-if (Open == null){
	-return null;}
	-return path;			

2	-public System.Type[]
	graphTypes
	-get {
	-return
	astarData.graphTypes;}
	-public Matrix
	translate (float _X,
	float _Y, float _Z){
	-m[12] = _X;
	-m[13] = _Y;
	-m[14] = _Z;
	-return this;}
	astarmath
	-public virtual int[]
	InitialOpen
	(BinaryHeapM open, Int3
	targetPosition, Int3
	position,
	Path path,
	bool doOpen){
	-return BaseInitialOpen
	-if (readerStream !=
	null) {
	-readerStream.Close
	();}
	-if (writerStream !=
	null) {
	-if (anchors != null) {
	-SerializeAnchors ();}
	
	-writerStream.Close
	();}}clas astar
	serialize
	- return path;
		Untuk menentukan rute tujuan yang berada di openlist	Node n' ditemukann dari openList	[□]dikerjakan
				[ ] tidak dikerjakan

3	-public System.Type[] graphTypes	Mendapatkan rute yang dicari	Mendapatkan rute dng pareent	[□]dikerjakan

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	-get {
	-return
	astarData.graphTypes;}
	-public Matrix
	translate (float _X,
	float _Y, float _Z){
	-m[12] = _X;
	-m[13] = _Y;
	-m[14] = _Z;
	-return this;}
	astarmath
	-public virtual int[]
	InitialOpen
	(BinaryHeapM open, Int3
	targetPosition, Int3
	position,
	Path path,
	bool doOpen){
	- return path;
		dengan current node=goal	goal.	[ ] Tidak dikerjakan

4	-public System.Type[]
	graphTypes
	-get {
	-return
	astarData.graphTypes;}
	-public Matrix
	translate (float _X,
	float _Y, float _Z){
	-m[12] = _X;
	-m[13] = _Y;
	-m[14] = _Z;
	-return this;}
	astarmath
	-public virtual int[]
	InitialOpen
	(BinaryHeapM open, Int3
	targetPositio
	n, Int3
	position,
	Path path,
	bool doOpen){
	-return BaseInitialOpen
	-
	(open,targetPosition,po
	sition,path,doOpen);}
	clas node
	-public void Close () {
	-if (readerStream !=
	null) {
	-readerStream.Close
	();}
	-if (writerStream !=
	null) {
	-if (anchors != null) {
		Pencarian rule selanjutnyatidak dapat dilalui collison atau unwalkable. Kembali ke open list.	Pencarian di ulang kembali ke openlist	[□]dikerjakan
				[ ] Tidak dikerjakan

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	-SerializeAnchors ();}
	
	-writerStream.Close ();}}clas astar serialize
	-public int[]public int[] BaseInitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen) {
	-if (doOpen) {
	-connectionCosts = costs;}
	-if (connectionCosts == null) {
	-return null;}
	-int[] costs = connectionCosts;
	-connectionCosts = new int[connectionCosts.Length];
	-for (int i=0;i<connectionCosts.Length;i++) {
	-connectionCosts[i] = (connections[i].position-position).costMagnitude;}}
	-if (!doOpen) {
	-for (int i=0;i<connectionCosts.Length;i++) {
	-Node other = connections[i];
	-if (other.connections != null) {
	-for (int q = 0;q < ther.connections.Length;q++) {
	-if (other.connections[q] == this) {
	-other.connectionCosts[q] = connectionCosts[i];
	-break;}}}}}
	-public int[]public int[] BaseInitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen) {			

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	-if (doOpen) {
	-connectionCosts = costs;}
	-if (connectionCosts == null) {
	-return null;}
	-int[] costs = connectionCosts;
	-connectionCosts = new int[connectionCosts.Length];
	-for (int i=0;i<connectionCosts.Length;i++) {
	-connectionCosts[i] = (connections[i].position-position).costMagnitude;}}
	-if (!doOpen) {
	-for (int i=0;i<connectionCosts.Length;i++) {
	-Node other = connections[i];
	-if (other.connections != null) {
	-for (int q = 0;q < other.connections.Length;q++) {
	-if (other.connections[q] == this) {
	-other.connectionCosts[q] = connectionCosts[i];
	-break;}}}}}
	-if (connections[i].walkable && connections[i].area != area) {
	-stack.Push (connections[i]);
	-connections[i].area = area;}
	-if (doOpen) {
	-nodeR.g = nodeR.parent.g+nodeR.cost+penalty + nodeRunData.path.GetTagPenalty(tags);}
	-public virtual void UpdateAllG (NodeRun			

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan
	nodeR, NodeRunData nodeRunData) { -BaseUpdateAllG (nodeR, nodeRunData);} class node -if (Open == null){ -return null;} return path;			
5	-public System.Type[] graphTypes -get { -return astarData.graphTypes;} -public Matrix translate (float _X, float _Y, float _Z){ -m[12] = _X; -m[13] = _Y; -m[14] = _Z; -return this;} astarmath -public virtual int[] InitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen){ -return BaseInitialOpen - (open,targetPosition,position,path,doOpen);} clas node -public void Close () { -if (readerStream != null) { -readerStream.Close ();} -if (writerStream != null) { -if (anchors != null) { -SerializeAnchors ();} -writerStream.Close ();}}clas astar serialize -public int[]public int[] BaseInitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path,	Masukan kembali semua nighbor node ke dalam open dan hitung kembali F,G,H selanjutnya tentukan current node dan nilai F,G,H masukan ke dalam open.	Memasukan kembali semua nighbor node ke dalam open dan menghitung kembali F,G,H	[ ]dikerjakan [ ] Tidak dikerjakan

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	bool doOpen) {
	-if (doOpen) {
	-connectionCosts = costs;}
	-if (connectionCosts == null) {
	-return null;}
	-int[] costs = connectionCosts;
	-connectionCosts = new int[connectionCosts.Length];
	-for (int i=0;i<connectionCosts.Length;i++) {
	-connectionCosts[i] = (connections[i].position-position).costMagnitude;}}
	-if (!doOpen) {
	-for (int i=0;i<connectionCosts.Length;i++) {
	-Node other = connections[i];
	-if (other.connections != null) {
	-for (int q = 0;q < ther.connections.Length;q++) {
	-if (other.connections[q] == this) {
	-other.connectionCosts[q] = connectionCosts[i];
	-break;}}}}}
	-public int[]public int[] BaseInitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen) {
	-if (doOpen) {
	-connectionCosts = costs;}
	-if (connectionCosts == null) {
	-return null;}
	-int[] costs = connectionCosts;
	-connectionCosts = new int[connectionCosts.Len			

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	gth];
	-for (int i=0;i<connectionCosts.Length;i++) {
	-connectionCosts[i] = (connections[i].position-position).costMagnitude;}}
	-if (!doOpen) {
	-for (int i=0;i<connectionCosts.Length;i++) {
	-Node other = connections[i];
	-if (other.connections != null) {
	-for (int q = 0;q < other.connections.Length;q++) {
	-if (other.connections[q] == this) {
	-other.connectionCosts[q] = connectionCosts[i];
	-break;}}}}}
	-if (connections[i].walkable && connections[i].area != area) {
	-stack.Push (connections[i]);
	-connections[i].area = area;}
	-if (doOpen) {
	-nodeR.g = nodeR.parent.g+nodeR.cost+penalty + nodeRunData.path.GetTagPenalty(tags);}			

6.	-public System.Type[] graphTypes
	-get {
	-return astarData.graphTypes;}
	-public Matrix translate (float _X, float _Y, float _Z){
	-m[12] = _X;
	-m[13] = _Y;
	-m[14] = _Z;
	-return this;}
		Mendapatkan nilai F yang lebih kecil dan F sebelumnya cek ulang nilai F,G,H	Mendapatkan nilai F yang lebih kecil	

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	astarmath
	-public virtual int[] InitialOpen
	(BinaryHeapM open, Int3 targetPositio n, Int3 position, Path path, bool doOpen){
	-return BaseInitialOpen -
	(open,targetPosition,po sition,path,doOpen);}
	clas node
	-public void Close () {
	-if (readerStream != null) {
	-readerStream.Close ();}
	-if (writerStream != null) {
	-if (anchors != null) {
	-SerializeAnchors ();}
	
	-writerStream.Close ();}}clas astar serialize
	-public int[]public int[] BaseInitialOpen (BinaryHeapM open, Int3 targetPosition, Int3 position, Path path, bool doOpen) {
	1. if (doOpen) {
	-connectionCosts = costs;}
	-if (connectionCosts == null) {
	-return null;}
	-int[] costs = connectionCosts;
	-connectionCosts = new int[connectionCosts.Len gth];
	-for (int i=0;i<connectionCosts.L ength;i++) {
	-connectionCosts[i] = (connections[i].positio n-
	position).costMagnitude ;}}
	-if (!doOpen) {
	-for (int			

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	i=0;i<connectionCosts.Length;i++) {
	-Node other =
	-connections[i];
	-if (other.connections != null) {
	-for (int q = 0;q < other.connections.Length;q++) {
	-if (other.connections[q] == this) {
	-other.connectionCosts[q] = connectionCosts[i];
	-break;}}}}}
	-if (connections[i].walkable && connections[i].area != area) {
	-stack.Push (connections[i]);
	-connections[i].area = area;}
	-if (doOpen) {
	-nodeR.g = nodeR.parent.g+nodeR.cost+penalty + nodeRunData.path.GetTagPenalty(tags);}
	-public virtual void UpdateAllG (NodeRun nodeR, NodeRunData nodeRunData) {
	-BaseUpdateAllG (nodeR, nodeRunData);} class node
	- return path;			

7.	-public System.Type[] graphTypes
	-get {
	-return astarData.graphTypes;}
	-public Matrix translate (float _X, float _Y, float _Z){
	-m[12] = _X;
	-m[13] = _Y;
	-m[14] = _Z;
	-return this;}
	astarmath
	-public virtual int[]
		Tidak ada lagi node di dalam open, node di temukan.	Node ditemukan	

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan

	InitialOpen
	(BinaryHeapM open, Int3
	targetPositio
	n, Int3
	position,
	Path path,
	bool doOpen){
	-return BaseInitialOpen
	-
	(open,targetPosition,po
	sition,path,doOpen);}
	clas node
	-public void Close () {
	-if (readerStream !=
	null) {
	-readerStream.Close
	();}
	-if (writerStream !=
	null) {
	-if (anchors != null) {
	-SerializeAnchors ();}
	
	-writerStream.Close
	();}}clas astar
	serialize
	-public int[]public
	int[] BaseInitialOpen
	(BinaryHeapM open, Int3
	targetPosition, Int3
	position, Path path,
	bool doOpen) {
	1. if (doOpen) {
	-connectionCosts =
	costs;}
	-if (connectionCosts ==
	null) {
	-return null;}
	-int[] costs =
	connectionCosts;
	-connectionCosts = new
	int[connectionCosts.Len
	gth];
	-for (int
	i=0;i<connectionCosts.L
	ength;i++) {
	-connectionCosts[i] =
	(connections[i].positio
	n-
	position).costMagnitude
	;}}
	-if (!doOpen) {
	-for (int
	i=0;i<connectionCosts.L
	ength;i++) {			

No	Kasus uji	Pengamatan	Yang diharapkan	Keterangan
	-Node other = -connections[i]; -if (other.connections != null) { -for (int q = 0;q < other.connections.Length;q++) { -if (other.connections[q] == this) { - other.connectionCosts[q] = connectionCosts[i]; -break;}}}}} -if (connections[i].walkable && connections[i].area != area) { -stack.Push (connections[i]); -connections[i].area = area;} -if (doOpen) { -nodeR.g = nodeR.parent.g+nodeR.cost+penalty + nodeRunData.path.GetTagPenalty(tags);} -public virtual void UpdateAllG (NodeRun nodeR, NodeRunData nodeRunData) { -BaseUpdateAllG (nodeR, nodeRunData);} class node - if (Open == null){ 1. - return path; return null;}			

2. Pengujian *White Box* Algoritma *Fuzzy Logic*

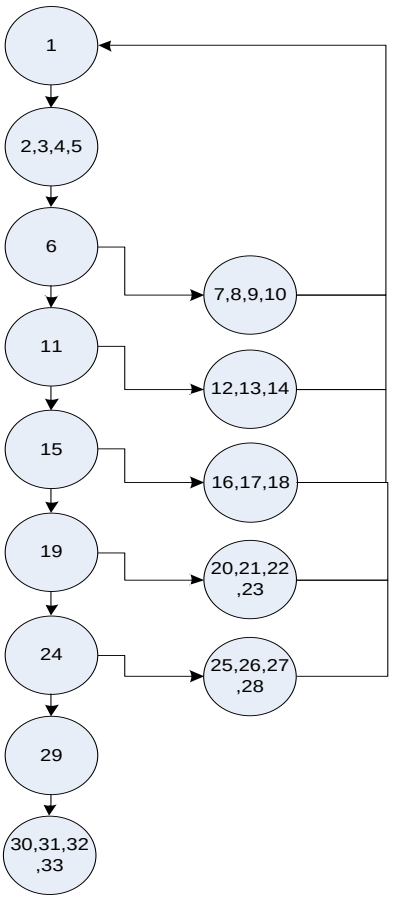
Berikut ini merupakan algoritma *fuzzy logic* yang penerapannya menggunakan bahasa pemrograman C#:

```
1.  jarak = (target.position -
    myTransform.position).magnitude;
2.  rules rule = new rules(jarak,hitPoints);
3.  defuzzyfikasi def = new defuzzyfikasi();
4.  rule.periksaRule();
5.  def. hasilDefuzzy();
```

```
6.   if(def.hasilDefuzzyfikasi >80.of &&
     def.hasilDefuzzyfikasi <= 96.0f) {
7.   Melarikandiri();
8.   myTransform.position += myTransform.forward * moveSpeed
     * Time.deltaTime;
9.   anim.animation.CrossFade(runAnim);
10.  }
11.  else if(def.hasilDefuzzyfikasi > 64.0f &&
     def.hasilDefuzzyfikasi <= 80.0f){
12.  Menambah darah();
13.  anim.animation.CrossFade(runAnim);
14.  }
15.  else if(def.hasilDefuzzyfikasi > 32.0f &&
     def.hasilDefuzzyfikasi <= 48.0f){
16.  bertahan();
17.  anim.animation.CrossFade(idleAnim);
18.  }
19.  else if(def.hasilDefuzzyfikasi > 48.0f &&
     def.hasilDefuzzyfikasi <= 64.0f){
20.  menghindar();
21.  float trans = Time.deltaTime * 2;
22.  myTransform.Translate(-trans, 0, 0);
23.  }
24.  else if (def.hasilDefuzzyfikasi > 16.0f &&
     Def.hasillDefuzzifikasi <=32.0f){
25.  Menyerang();
26.  checkInDelay();
27.  attackTime = Time.time _attackRepeatTime;
28.  }
29.  else if (def.hasilDefuzzyfikasi > 0.0f &&
     Def.hasillDefuzzifikasi <=16.0f){
30.  menyerang brutal() {
31.  moveSpeed=4;
32.  Damage=30;
33.
```

a. *Flow Graph* Algoritma *Fuzzy Logic*

Berdasarkan *sourcecode* di atas maka *flow graph* algoritma *fuzzy logic* dapat dilihat pada gambar 4.15 :



Gambar 4.15 ***Flow Graph Fuzzy Logic***

Keterangan :

= Menggambarkan *node*/simpul

= *edges*/panah

Dari Gambar 4.15 dapat dihitung *cyclomatic complexity* sebagai berikut *Cyclomatic Complexity* V(G) :

1). E = 18, N = 14

2). V(G) = E – N + 2

3). V(G) = 18 – 14 + 2

4). V(G) = 4 + 2 = 6

Keterangan : E = jumlah *edges*/panah, N = jumlah *node*/simpul

Jadi, *cyclomatic complexity* untuk gambar 4.15 adalah 6. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 6 *path* yang terdiri dari :

1). *Path* 1 = 1-2-3-4-5-6-7-8-9-11-12-13-15-16-17-19-20-21-22-24-25-26-27-29-30-31-32-33

Untuk menentukan jarak dan kekuatan dari setiap variabel linguistik dan dibandingkan sesuai dengan rule, yang menghasilkan keputusan melarikan diri.

2). *Path* 2 = 1-2-3-4-5-6-7-8-9-10-33

Jika rule > 64 dan <= 80 maka menghasilkan keputusan menambah darah.

3). *Path* 3 = 1-2-3-4-5-6-7-8-9-11-12-13-14-33

Jika rule > 32 dan <= 48 maka menghasilkan keputusan bertahan.

4). *Path* 4 = 1-2-3-4-5-6-7-8-9-11-12-13-15-16-17-18-33

Jika rule > 48 dan <= 64 maka menghasilkan keputusan menghindar.

5). *Path* 5 = 1-2-3-4-5-6-7-8-9-11-12-13-15-16-17-19-20-21-22-23-33

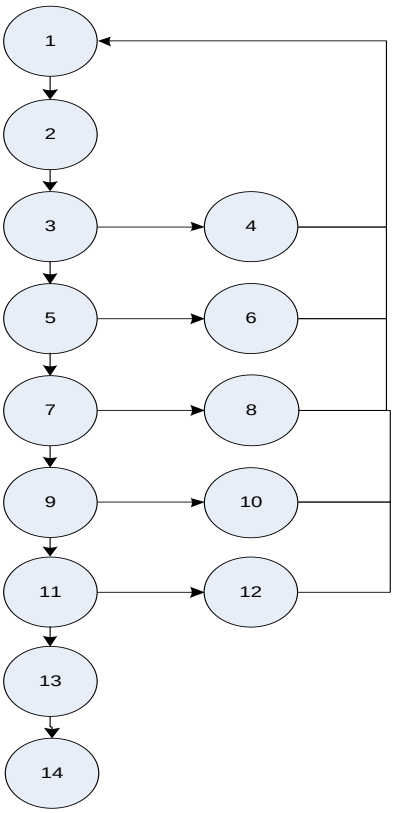
Jika rule > 16 dan <= 32 maka menghasilkan keputusan menyerang.

6). *Path* 5 = 1-2-3-4-5-6-7-8-9-11-12-13-15-16-17-19-20-21-22-24-25-26-27-28-33

Jika rule > 0 dan <= 16 maka menghasilkan keputusan menyerang brutal.

b. *Graph Matriks* Algoritma *Fuzzy Logic*

Penomoran ulang *flow graph* atau penyederhanaan berdasarkan kondisi atau simpul sebagai berikut :



Gambar 4.16 Penomoran ulang *Flow Graph Fuzzy Logic*

Tabel 4. 12 *Graph Matriks* Algoritma *Fuzzy Logic*

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

Keterangan:

1. Baris dan kolom merepresentasikan simpul.
3. 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.
    a) Baris 1 dan kolom 2 = merepresentasikan adanya keterhubungan antar node/simpul 1 dan node/simpul 2 dari *flow graph*.
    b) Baris 2 dan kolom 3 = merepresentasikan adanya keterhubungan antar node/simpul 2 dan node/simpul 3 dari *flow graph*.
    c) Baris 3 dan kolom 4 dan 5 = merepresentasikan adanya keterhubungan antar node/simpul 3 dan node/simpul 4 dan 5 dari *flow graph*.
    d) Baris 4 dan kolom 14 = merepresentasikan adanya keterhubungan antar node/simpul 4 dan node/simpul 14 dari *flow graph*.
    e) Baris 5 dan kolom 6 dan 7 = merepresentasikan adanya keterhubungan antar node/simpul 5 dan node/simpul 6 dan 7 dari *flow graph*.
    f) Baris 6 dan kolom 14 = merepresentasikan adanya keterhubungan antar node/simpul 6 dan node/simpul 14 dari *flow graph*.
    g) Baris 7 dan kolom 8 dan 9 = merepresentasikan adanya keterhubungan antar node/simpul 7 dan node/simpul 8 dan 9 dari *flow graph*.
    h) Baris 8 dan kolom 14 = merepresentasikan adanya keterhubungan antar node/simpul 8 dan node/simpul 14 dari *flow graph*.
    i) Baris 9 dan kolom 10 dan 11 = merepresentasikan adanya keterhubungan antar node/simpul 9 dan node/simpul 10 dan 11 dari *flow graph*.
    j) Baris 10 dan kolom 12 dan 13 = merepresentasikan adanya keterhubungan antar node/simpul 10 dan node/simpul 12 dan 13 dari *flow graph*.
    k) Baris 11 dan kolom 12 dan 13 = merepresentasikan adanya keterhubungan antar node/simpul 11 dan node/simpul 12 dan 13 dari *flow graph*.

l) Baris 12 dan kolom 14 = merepresentasikan adanya keterhubungan antar node/simpul 12 dan node/simpul 14 dari *flow graph*.

c. Hasil Uji Algoritma *Fuzzy Logic*.

Tabel 4. 13 Hasil Uji Algoritma *Fuzzy Logic*

No	Kasus uji	Pengamatan	Yang diharapkan	Kesimpulan
1	- jarak = (target.position - myTransform.position).magnitude; - rules rule = new rules(jarak,hitPoints) - defuzzyfikasi def = new defuzzyfikasi(); - rule.periksaRule(); - def. hasilDefuzzy(); - if(def.hasilDefuzzyfikasi >80.of && def.hasilDefuzzyfikasi <= 96.0f){ - Melarikandiri(); - myTransform.position += myTransform.forward * moveSpeed * Time.deltaTime; - anim.animation.CrossFade(runAnim); - }	NPC melarikan diri	Hasil defuzzy > 80 dan <= 96 NPC melarikan diri	[□]Dikerjakan []tidak dikerjakan
2	- else if(def.hasilDefuzzyfikasi > 64.0f && def.hasilDefuzzyfikasi <= 80.0f){anim.animation.CrossFade(runAnim);} - Menambah darah(); - anim.animation.CrossFade(runAnim); - }	NPC Menambah darah	Hasil defuzzy > 64 dan <= 80 NPC Menambah darah	[□]Dikerjakan [ ]tidak dikerjakan
3	- else if(def.hasilDefuzzyfikasi > 32.0f && def.hasilDefuzzyfikasi <= 48.0f){ - bertahan(); - anim.animation.CrossFade(idleAnim); - }	NPC bertahan	Hasil defuzzy > 32 dan <= 48 NPC Bertahan	[□]Dikerjakan [□]tidak dikerjakan
4	- else if(def.hasilDefuzzyfikasi > 48.0f && def.hasilDefuzzyfikasi <= 64.0f){	NPC Menghindar	Hasil defuzzy > 48 dan <= 64 NPC Menghindar	[□]Dikerjakan [□]tidak dikerjakan

No	Kasus uji	Pengamatan	Yang diharapkan	Kesimpulan
	- menghindar(); - float trans = Time.deltaTime * 2; - myTransform.Translate(-trans, 0, 0); - }			
5	- else if(def.hasilDefuzzyfikasi > 16.0f && def.hasilDefuzzyfikasi <= 32.0f){ - menyerang(); - checkInDelay(); - attackTime = Time.time _attackRepeatTime; - }	NPC menyerang	Hasil defuzzy > 16 dan <= 32 NPC menyerang	[□]Dikerjakan [□]tidak dikerjakan

6	- else if(def.hasilDefuzzyfikasi > 0.0f && def.hasilDefuzzyfikasi <= 16.0f){
	- menyerang brutal() {
	- moveSpeed=4;
	- Damage=30;
		NPC menyerang brutal	Hasil defuzzy > 0 dan <= 16 NPC menyerang brutal	[□]Dikerjakan
				[□]tidak dikerjakan

4.2.1.3 Kesimpulan Pengujian *Alpha*

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa sistem sudah berjalan seperti yang diharapkan dan secara fungsional sudah dapat menghasilkan keluaran yang diharapkan, akan tetapi pada prosesnya masih memungkinkan untuk terjadinya kesalahan, namun frekuensi kesalahan masih relatif kecil.

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.

Kuesioner disebarkan secara acak tanpa memperhatikan strata yang ada dalam populasi. Kuesioner disebarkan kepada 18 orang dari umur 13 tahun ke atas, dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian penerapan sistem yang telah dibangun. Kuesioner

terdiri dari 5 pernyataan dengan menggunakan skala 1 sampai 5 (skala likert). Untuk mencari nilai persentase dari masing-masing jawaban kuesioner digunakan rumus skala likert sebagai berikut :

$$Y=P/Q*100\% \quad (4.1)$$

Keterangan :

1). Y= Nilai persentase yang dicari.
2). P= Bobot setiap pilihan jawaban.
3). Q= Nilai tertinggi dikalikan dengan jumlah sampel.

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

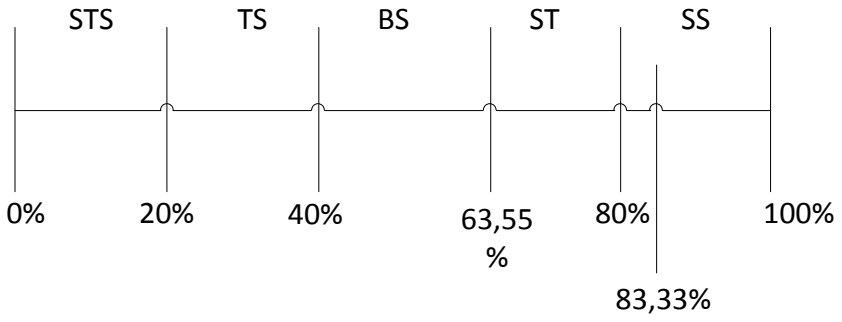
1. Apakah desain grafis 3D di dalam game The Tracker sudah menarik dibandingkan dengan Alien Hunter?

Tabel 4. 14 Perhitungan persentase pertanyaan pertama

Pertanyaan	Keterangan	Skor	Frekuensi	S
1	Sangat Setuju	5	7	35
	Setuju	4	7	28
	Biasa saja	3	4	12
	Tidak Setuju	2	-	-
	Sangat tidak setuju	1	-	-
Jumlah			**18**	**75**

$$Y = \frac{75}{90} x100 = 83{,}33\%$$

Berikut kriteria interpretasi skor:



Gambar 4.17 Interpretasi Skor Pertanyaan No 1

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap antarmuka *game The Tracker* adalah 83,33% dari yang diharapkan 100%, atau dapat dikategorikan setuju menarik.

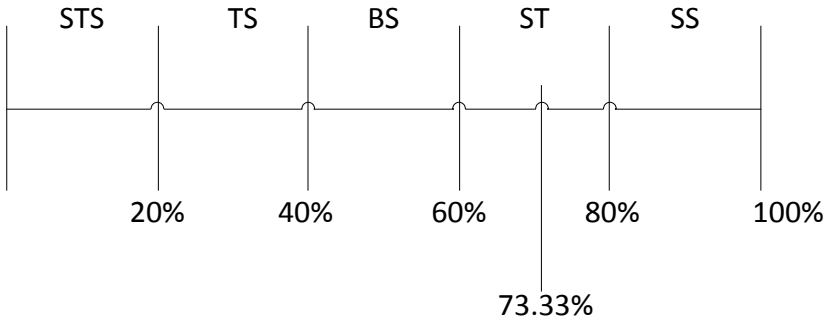
2. Apakah musuh di dalam *game The Tracker* sulit untuk dikalahkan dibandingkan *game Alien Hunter*?

Tabel 4. 15 Perhitungan persentase pertanyaan Kedua

Pertanyaan	Keterangan	Skor	Frekuensi	S
2	Sangat Setuju	5	3	15
	Setuju	4	6	24
	Biasa saja	3	9	27
	Tidak Setuju	2	-	-
	Sangat tidak setuju	1	-	-
Jumlah			**18**	**66**

$$Y = \frac{66}{90} x100 = 73{,}33\%$$

Berikut kriteria interpretasi skor:



Gambar 4.18 Interpretasi Skor Pertanyaan No 2

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap *game The Tracker* sulit untuk dikalahkan adalah 73.33% dari yang diharapkan 100%, atau dapat dikategorikan setuju sudah terlihat.

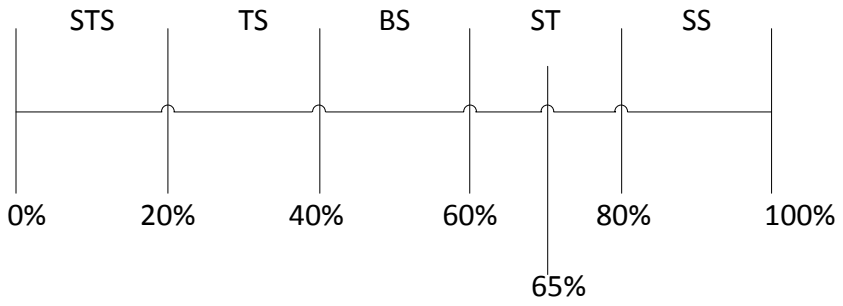
3. Apakah anda merasa tertantang menyelesaikan *game The Tracker* sampai *level* akhir dibandingkan dengan *game Alien Hunter*?

Tabel 4. 16 Perhitungan persentase pertanyaan Ketiga

Pertanyaan	Keterangan	Skor	Frekuensi	S
3	Sangat Setuju	5	2	10
	Setuju	4	7	28
	Biasa saja	3	9	27
	Tidak Setuju	2	-	-
	Sangat tidak setuju	1	-	-
Jumlah			**18**	**72**

$$Y = \frac{72}{90} x100 = 65\%$$

Berikut kriteria interpretasi skor:



Gambar 4.19 Interpretasi Skor Pertanyaan No 3

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap tantangan sebesar 65% bisa dikatakan setuju.

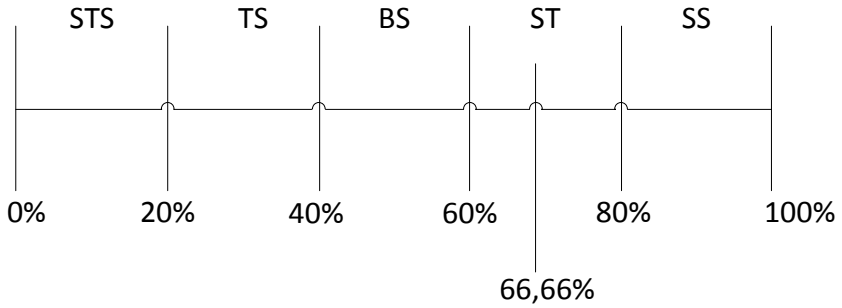
4. Apakah musuh di dalam *game The Tracker* sudah bervariasi dibandingkan dengan *game Alien Hunter*?

Tabel 4. 17 Perhitungan persentase pertanyaan Keempat

Pertanyaan	Keterangan	Skor	Frekuensi	S
4	Sangat Setuju	5	3	15
	Setuju	4	3	12
	Biasa saja	3	11	33
	Tidak Setuju	2		
	Sangat tidak setuju	1	-	-
Jumlah			**18**	**60**

$$Y = \frac{60}{90} x100 = 66{,}66\%$$

Berikut kriteria interpretasi skor:



Gambar 4.20 Interpretasi Skor Pertanyaan No 4

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap Variasi musuh pada *game The Tracker* sampai *level* akhir adalah 66,66% dari yang diharapkan 100%, atau dapat dikategorikan setuju.

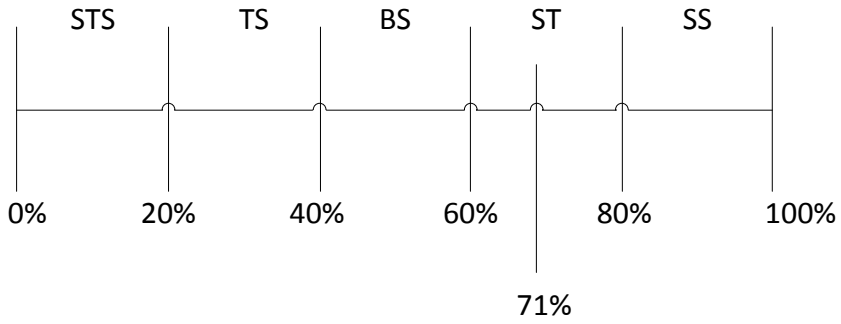
5. Secara keseluruhan.Apakah *game The Tracker* sudah menantang untuk dimainkan, dibandingkan *game alien hunter*?

Tabel 4. 18 Perhitungan persentase pertanyaan Kelima

Pertanyaan	Keterangan	Skor	Frekuensi	S
5	Sangat Setuju	5	2	10
	Setuju	4	6	24
	Biasa saja	3	10	30
	Tidak Setuju	2		
	Sangat tidak setuju	1	-	-
Jumlah			**18**	**93**

$$Y = \frac{93}{90} x100 = 71{,}11\%$$

Berikut kriteria interpretasi skor:



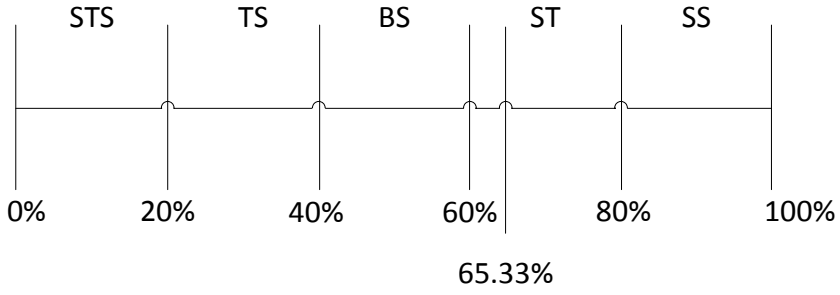
Gambar 4.21 Interpretasi Skor Pertanyaan No 5

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap info petunjuk disetiap *level* pada *game The Tracker* adalah 71,11% dari yang diharapkan 100%, atau dapat dikategorikan setuju dan *game the tracker* menantang untuk dimainka.

4.2.2.1 Kesimpulan keseluruhan Pengujian *Beta*

Berdasarkan dari jawaban yang diberikan oleh responden, bisa ditarik kesimpulan bahwa "*Game First Person Shooter* 3D *The Tracker*" dinyatakan Setuju memiliki desain grafis 3D yang menarik dengan nilai 83,33, untuk musuh sulit dikalahkan memiliki nilai 77,33, responden dalam memainkan *game The Tracker* sudah setuju merasa tertantang untuk menyelesaikan *game* sampai *level* terakhir dengan nilai 65%, variasi musuh di dalam *game the tracker* sudah bervariasi dengan nilai 66,66% dan secara keseluruhan *game The Tracker* setuju menantang untuk dimainkan dengan nilai 71,11%.

Game First Person Shooter 3D *The Tracker* menghasilkan nilai rata-rata 60,57 % atau bisa dikatakan setuju dari 5 pertanyataan yang di jawab oleh responden terhadap *game* yang dibangun. Berikut kriteria interpretasi skor nilai rata-rata sebagai berikut :



Gambar 4.22 Interpretasi Skor nilai rata-rata