# 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 *Alien Hunter* 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 :

1) Processor Dual Core 1,80 Ghz
2) RAM 1 GB
3) VGA 512 MB
4) Harddisk 500 GB
5) Mouse dan Keyboard

### 4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *Alien Hunter* adalah :

1. Sistem operasi minimal Windows Xp
2. Graphic card intel HD
3. DirectX 11

### 4.1.3 Implementasi Instalasi Aplikasi *Game*

Untuk dapat memainkan *game Alien Hunter*, pemain terlebih dahulu harus menginstall aplikasi *game*. Untuk mulai memainkan *game*, pemain hanya perlu menjalankan *file Alien Hunter.exe* yang telah ada dalam perangkat keras.

### 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 *game Alien Hunter* :

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 Alien Hunter*. Terdapat 4 (tiga) sub menu di dalamnya, yaitu menu *play* yang berfungsi untuk memulai permainan, menu *HowtoPlay* yang berfungsi untuk melihat *controller* yang digunakan didalam *game*, menu *Credits* yang berfungsi untuk melihat pengembang *game* dan *Exit* yang berfungsi untuk keluar dari aplikasi.

2. Antarmuka Menu *Play*

Saat pengguna memilih menu *play*, maka antarmuka yang ditampilkan oleh sistem adalah; Pertama, antarmuka *introscene* berupa vidio cerita awal dari *game*. Selanjutnya akan masuk ke antarmuka petunjuk misi dan kemudian akan

masuk pada antarmuka permainan. Jika seluruh misi telah terselesaikan maka akan masuk ke antarmuka *endingscene* beupa vidio akhir cerita *game*. Berikut ini merupakan tampilan dari antarmuka menu *play* :



Gambar 4.2 Antarmuka *Introscene*

Gambar 4.2 merupakan antarmuka *introscene video* yang terdapat pada *game Alien Hunter*.




Gambar 4.3 Antarmuka Petunjuk misi

Gambar 4.3 merupakan antarmuka pentunjuk misi yang terdapat pada *game Alien Hunter*.




Gambar 4.4 Antarmuka Permainan

Gambar 4.4 merupakan antarmuka permainan yang terdapat pada *game Alien Hunter*.



Gambar 4.5 Antarmuka *endingscene*

Gambar 4.5 merupakan antarmuka *endingscene video* yang terdapat pada *game Alien Hunter*.

3. Antarmuka Menu *HowtoPlay*

Saat pengguna memilih menu *HowtoPlay*, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka *controller* yang digunakan dalam *game Alien Hunter*. Gambar 4.6 merupakan tampilan dari antarmuka menu *HowtoPlay* :




Gambar 4.6 Antarmuka Menu *HowtoPlay*

4. Antarmuka Menu *Credits*

Saat pengguna memilih menu *Credits*, maka antarmuka yang ditampilkan oleh sistem adalah informasi mengenai pengembang dari *game Alien Hunter*. Gambar 4.7 merupakan tampilan dari antarmuka menu *Credits* :




Gambar 4.7 Antarmuka Menu *Credits*

## 4.2 Pengujian

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 *Alien Hunter* digunakan pengujian *black box testing*. Cara pengujian dilakukan melalui dua tahap, yaitu pengujian *alpha* (fungsional) dan pengujian *beta*.

### 4.2.1 Pengujian *Alpha*

Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi 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 Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game Alien Hunter*. Skenario pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada table 4.1 :

Tabel 4.1 Skenario Pengujian Aplikasi

| No | Komponen yang di uji | Skenario Pengujian | Jenis Pengujian |
|---|---|---|---|
| 1. | Menu Utama | Memilih menu *Play* | *Black box* |
| | | Memilih menu *HowtoPlay* | *Black box* |
| | | Memilih menu *Credits* | *Black box* |
| | | Memilih menu *Exit* | *Black box* |
| 2. | Menu *Play* | Memilih menu *Play* | *Black box* |
| 3. | Menu *HowtoPlay* | Memilih menu *HowtoPlay* | *Black box* |
| | | Memilih menu *Back* | *Black box* |
| 4. | Menu *Credits* | Memilih menu *Credits* | *Black box* |
| | | Memilih menu *Back* | *Black box* |
| 5. | *Menu Exit* | Memilih menu *Exit* | *Black box* |
| 6. | Permainan | Tekan tombol W *keyboard* | *Black box* |
| | | Tekan tombol A *keyboard* | *Black box* |
| | | Tekan tombol D *keyboard* | *Black box* |
| | | Tekan tombol S *keyboard* | *Black box* |
| | | Memilih kiri *mouse* | *Black box* |

| No | Komponen yang di uji | Skenario Pengujian | Jenis Pengujian |
|---|---|---|---|
| | | Tekan tombol P *keyboard* | *Black box* |
| | | Tekan tombol 1 *keyboard* | *Black box* |
| | | Tekan tombol 2 *keyboard* | *Black box* |
| 7. | Algoritma | A* (A *Star*) | *White box* |
| | | *Fuzzy Logic* | *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 | Memilih Menu *Play* | Sistem menampilkan halaman *Intro scene video* | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Memilih Menu *HowtoPlay* | Sistem menampilkan halaman menu *HowtoPlay* | [√] Berhasil<br>[ ] Tidak Berhasil |

| No | Kasus yang akan di uji | Skenario dan Hasil Uji | | |
|---|---|---|---|---|
| | | **Skenario Uji** | **Hasil yang Diharapkan** | **Kesimpulan** |
| | | Memilih Menu *Credits* | Sistem menampilkan halaman menu *Credits* | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Klik Menu *Exit* | Menutup atau Keluar dari aplikasi | [√] Berhasil<br>[ ] Tidak Berhasil |
| 2. | Menu *Play* | Memilih Menu *Play* | Sistem menampilkan halaman *Introscene video* | [√] Berhasil<br>[ ] Tidak Berhasil |
| 3. | Menu *HowtoPlay* | Memilih Menu *HowtoPlay* | Sistem menampilkan petunjuk *controller* yang digunakan dalam *game* | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Memilih *Back* | Sistem menampilkan menu utama | [√] Berhasil<br>[ ] Tidak Berhasil |
| 4. | Menu *Credits* | Memilih Menu *Credits* | Sistem menampilkan keterangan pengembang *game* | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | | Sistem | [√] Berhasil |

| No | Kasus yang akan di uji | Skenario dan Hasil Uji | | |
|---|---|---|---|---|
| | | **Skenario Uji** | **Hasil yang Diharapkan** | **Kesimpulan** |
| | | Memilih *Back* | menampilkan menu utama | [ ] Tidak Berhasil |
| 5. | *Menu Exit* | Memilih Menu *Exit* | Menutup atau Keluar dari aplikasi | [√] Berhasil<br>[ ] Tidak Berhasil |
| 6. | *Permainan* | Tekan tombol W *keyboard* | Karakter pemain bergerak kedepan | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol A *keyboard* | Karakter pemain bergerak kekiri | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol S *keyboard* | Karakter pemain bergerak kebelakang | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol D *keyboard* | Karakter pemain bergerak kekanan | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol W *keyboard* | Karakter pemain bergerak kedepan | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Memilih kiri *mouse* | Karakter pemain menembak | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol P *keyboard* | Menampilkan *Pause Screen* | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol 1 *keyboard* | Karakter memakai senjata *Machinegun* | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Tekan tombol 2 | Karakter | [√] Berhasil |

| No | Kasus yang akan di uji | Skenario dan Hasil Uji | | |
|---|---|---|---|---|
| | | **Skenario Uji** | **Hasil yang Diharapkan** | **Kesimpulan** |
| | | *keyboard* | memakai senjata *Rocketlouncher* | [ ] Tidak Berhasil |

#### 4.2.1.3 Kasus dan Hasil Pengujian (*White Box Testing*)

**1. Pengujian *White Box* Algoritma A***

Berikut ini merupakan algoritma A* yang implementasinya menggunakan bahasa pemrograman C#:

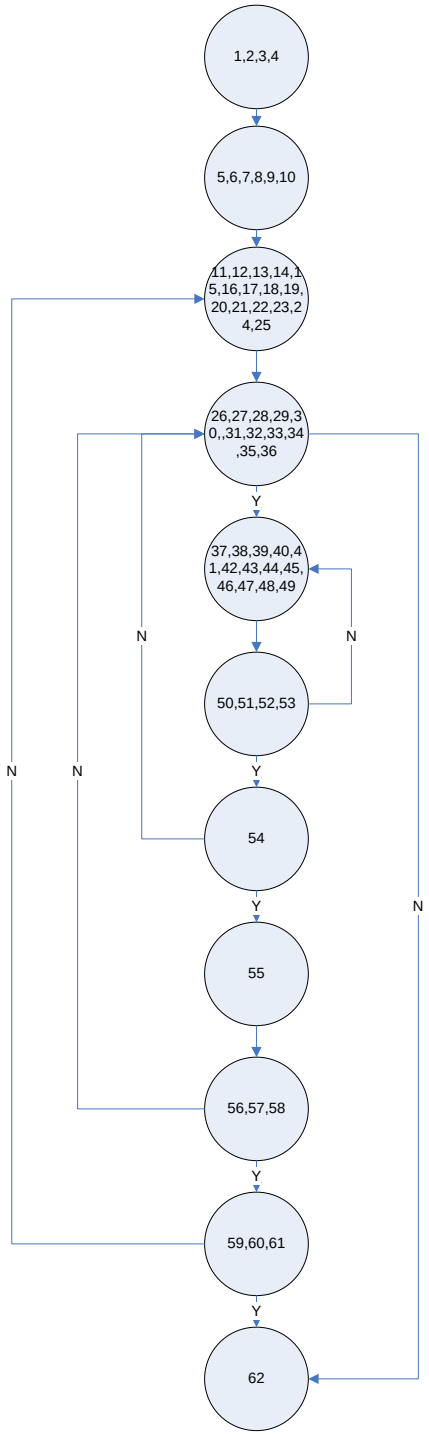
```
1.  public System.Type[] graphTypes{
2.  get {
3.  return astarData.graphTypes;}
4.  }
5.  public Matrix translate (float _X, float _Y, float _Z){
6.  m[12] = _X;
7.  m[13] = _Y;
8.  m[14] = _Z;
9.  return this;
10. }
11. public virtual int[] InitialOpen (BinaryHeapM open,
    Int3
    targetPosition, Int3 position, Path path, bool doOpen)
    {
12. return BaseInitialOpen
13. (open,targetPosition,position,path,doOpen);
14. }
15. public void Close () {
16. if (readerStream != null) {
17. readerStream.Close ();
18. }
19. if (writerStream != null) {
20. if (anchors != null) {
```

```
21.  SerializeAnchors ();
22.  }
23.  writerStream.Close ();
24.  }
25.  }
26.  if (doOpen) {
27.  connectionCosts = costs;
28.  }
29.  if (connectionCosts == null) {
30.  return null;
31.  }
32.  int[] costs = connectionCosts;
33.  connectionCosts = new int[connectionCosts.Length];
34.  for (int i=0;i<connectionCosts.Length;i++) {
35.  connectionCosts[i] = (connections[i].position-
     position).costMagnitude;
36.  }
37.  if (!doOpen) {
38.  for (int i=0;i<connectionCosts.Length;i++) {
39.  Node other = connections[i];
40.  if (other.connections != null) {
41.  for (int q = 0;q < other.connections.Length;q++) {
42.  if (other.connections[q] == this) {
43.  other.connectionCosts[q] = connectionCosts[i];
44.  break;
45.  }
46.  }
47.  }
48.  }
49.  }
50.  if (connections[i].walkable && connections[i].area !=
     area) {
51.  stack.Push (connections[i]);
52.  connections[i].area = area;
53.  }
54.  if (doOpen) {
55.  nodeR.g = nodeR.parent.g+nodeR.cost+penalty +
     nodeRunData.path.GetTagPenalty(tags);
```

```
56.  public virtual void UpdateAllG (NodeRun nodeR,
     NodeRunData nodeRunData) {
57.  BaseUpdateAllG (nodeR, nodeRunData);
58.  }
59.  if (Open == null){
60.  return null;
61.  }
62.  return path;
```

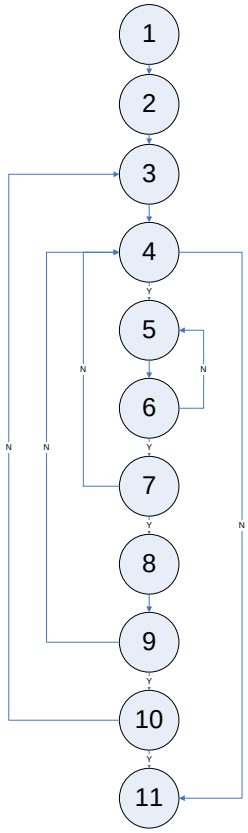
a. *Flow Graph* Algoritma A*

   Berdasarkan *sourcecode* di atas maka *flow graph* algoritma A* adalah dapat dilihat pada gambar 4.8:




Gambar 4.8 *Flow Graph* Algoritma A*

Atau disederhanakan berdasarkan kondisi atau simpul seperti pada gambar 4.9:




Gambar 4.9 Penyederhanaan *Flow Graph* Algoritma A*

Keterangan:



= Menggambarkan kondisi



= Menggambarkan aksi

b. *Cyclomatic Complexity* V(G)

E = 15, N = 11

V(G) = E – N + 2

V(G) = 15 – 11 + 2

V(G) = 4 + 2 = 6

Keterangan

E = Jumlah aksi

N = Jumlah Kondisi

c. *Independent Path*

*Path* 1 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62

*Path* 2 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-62

*Path* 3 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62

*Path* 4 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62

*Path* 5 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62

*Path* 6 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62

d. *Graph Matriks* Algoritma A*

Tabel 4.3 *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 | | | | | | | | 1-1=0 |
| 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 | | | 1-1=0 |
| 9 | | | | 1 | | | | | | 1 | | 2-1=1 |
| 10 | | | 1 | | | | | | | | 1 | 2-1=1 |
| 11 | | | | | | | | | | | | 1-1=0 |
| | | | | | | | | | | | Sum(E)+1 | 5+1=6 |

Keterangan:

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

e. Hasil Uji Algoritma A*

Tabel 4.4 Hasil Uji Algoritma A*

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| 1 | Rute = null, rute tidak ada | 1(*void*)<br>2(*void*)<br>3(*void*) | Rute tidak ditemukan | *Path* dikerjakan |

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| | | 4(*void*)<br>5(*void*)<br>6(*void*)<br>7(*void)*<br>8(*void*)<br>9(*void*)<br>10(*void*)<br>11(*void*) | | |
| 2 | Current Node = Goal. Rute ditemukan | 1,(*void*)<br>2(*void*)<br>3(*void*)<br>4(*void*)<br>11(void) | Rute ditemukan | *Path* dikerjakan |
| 3 | Node tidak dapat dilalui, pengujian collision atau unwalkable node | 1(*void*)<br>2(*void*)<br>3(*void*)<br>4(*void*)<br>5(*void*) | Node menjadi unwalkable | *Path* dikerjakan |

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| | | 6(*void*)<br>5(*void*)<br>6(*void*)<br>7(*void*)<br>8(*void*)<br>9(*void*)<br>10(*void*)<br>11(*void*) | | |
| 4 | Masukkan semua *neighbor node* ke dalam open dan hitung nilai F,G,H. Set *current node* | 1(*void*)<br>2(*void*)<br>3(*void*)<br>4(*void*)<br>5(*void*)<br>6(*void*)<br>7(*void*)<br>4(*void*)<br>5(*void*)<br>6(*void*) | *Current node* dan nilai F,G,H masuk ke dalam open | *Path* dikerjakan |

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| | | 7(*void)*<br>8(*void*)<br>9(*void*)<br>10(*void*)<br>11(*void*) | | |
| 5 | Nilai G yang didapat lebih kecil dari nilai G sebelumnya. Kalkulasi ulang nilai F,G,H | 1(*void*)<br>2(*void*)<br>3(*void*)<br>4(*void*)<br>5(*void*)<br>6(*void*)<br>7(*void)*<br>8(*void*)<br>9(*void*)<br>4(*void*)<br>5(*void*)<br>6(*void*)<br>7(*void)* | Nilai F,G,H, dikalkulasi ulang | *Path* dikerjakan |

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| | | 8(*void*)<br>9(*void*)<br>10(*void*)<br>11(*void*) | | |
| 6 | Open = null, tidak ada *node* di dalam open | 1(*void*)<br>2(*void*)<br>3(*void*)<br>4(*void*)<br>5(*void*)<br>6(*void*)<br>7(*void)*<br>8(*void*)<br>9(*void*)<br>10(*void*)<br>3(*void*)<br>4(*void*)<br>5(*void*)<br>6(*void*) | Tidak ada *node* di dalam open | *Path* dikerjakan |

| Kasus dan Hasil Uji | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| | | 7(*void)*<br>8(*void*)<br>9(*void*)<br>10(*void*)<br>11(*void*) | | |

f. Kesimpulan

Dari hasil pengujian algoritma A* yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak terhingga.

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

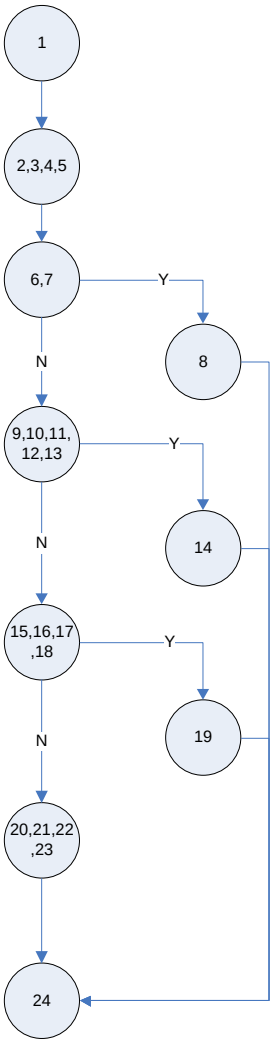
Berikut ini merupakan algoritma *fuzzy logic* yang penerapannya menggunakan bahasa pemrograman C#:

```
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((balikKeMarkas.position -
    myTransform.position).magnitude < 1.0){
7.  StartCoroutine(tambahDarah());
8.  }
9.  else if(def.hasilDefuzzyfikasi <= 10.0f) {
10. balikKeMarkas();
```

```
11. myTransform.position += myTransform.forward * moveSpeed
    * Time.deltaTime;
12. anim.animation.CrossFade(walkAnim);
13. playWalkSounds();
14. }
15. else if(def.hasilDefuzzyfikasi > 10.0f &&
    def.hasilDefuzzyfikasi <= 20.0f){
16. bertahan();
17. anim.animation.CrossFade(attackAnim2);
18. audio.Stop();
19. }
20. else if(def.hasilDefuzzyfikasi > 20.0f &&
    def.hasilDefuzzyfikasi <= 30.0f){
21. menyerang();
22. damage = Random.Range(20, 30);
23. anim.animation.CrossFade(walkAnim);
24. }
```

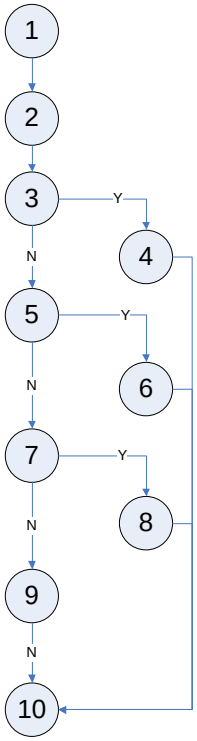
a. *Flow Graph* Algoritma *Fuzzy Logic*

Berdasarkan *pseudocode* di atas maka *flow graph* algoritma *fuzzy logic* dapat dilihat pada gambar 4.10:




Gambar 4.10 *Flow Graph* Algoritma *Fuzzy Logic*

Atau disederhanakan berdasarkan kondisi atau simpul seperti gambar 4.11:




Gambar 4.11 Penyederhanaan *Flow Graph* Algoritma *Fuzzy Logic*

Keterangan:

= Menggambarkan kondisi

= Menggambarkan aksi

b. *Cyclomatic Complexity* V(G)

E = 12, N = 10

V(G) = E – N + 2

V(G) = 12 – 10 + 2

V(G) = 2 + 2 = 4

Keterangan

E = Jumlah aksi

N = Jumlah Kondisi

c. *Independent Path*

*Path* 1 = 1-2-3-4-5-6-7-9-10-11-12-13-15-16-17-18-20-21-22-23-24

*Path* 2 = 1-2-3-4-5-6-7-8-24

*Path* 3 = 1-2-3-4-5-6-7-9-10-11-12-13-14-24

*Path* 4 = 1-2-3-4-5-6-7-9-10-11-12-13-15-16-17-18-19-24

d. *Graph Matriks* Algoritma *Fuzzy Logic*

Tabel 4.5 *Graph Matriks* Algoritma *Fuzzy Logic*

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

Keterangan:

1. Baris dan kolom merepresentasikan simpul

3. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

e. Hasil Uji Algoritma *Fuzzy Logic*

Tabel 4.6 Hasil Uji Algoritma *Fuzzy Logic*

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| 1 | NPC menambah *hitpoints* saat kembali ke markas | 1,(*void*)<br>2(*void*)<br>3(*void*)<br>4(*void*)<br>12(*void*) | NPC menambah *hitpoints* | *Path* dikerjakan |
| 2 | Hasil defuzzy <= 10 NPC melarikan diri ke markas | 1,(*void*)<br>2(*void*)<br>3(*void*)<br>5(*void*)<br>6(*void*)<br>12(*void*) | NPC melarikan diri ke markas | *Path* dikerjakan |
| 3 | Hasil defuzzy > 10 dan <= 20 NPC bertahan | 1,(*void*)<br>2(*void*)<br>3(*void*)<br>5(*void*)<br>7(*void*)<br>8(*void*) | NPC bertahan | *Path* dikerjakan |

| Kasus dan Hasil Uji | | | | |
|---|---|---|---|---|
| **No. *Path*** | **Butir uji algoritma *fuzzy logic*** | ***Node* (n), input kondisi** | **Pengamatan** | **Kesimpulan** |
| | | 12(*void*) | | |
| 4 | Hasil defuzzy > 20 dan <= 30 NPC menyerang | 1,(*void*)<br>2(*void*)<br>3(*void*)<br>5(*void*)<br>7(*void*)<br>9(*void*)<br>10(*void*)<br>12(*void*) | NPC menyerang | *Path* dikerjakan |

f. Kesimpulan

Dari hasil pengujian algoritma *fuzzy logic* yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak terhingga.

#### 4.2.1.4 Kesimpulan Pengujian *Alpha*

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa sistem sudah dapat berjalan sesuai dengan yang diharapkan beserta proses penanganan kesalahannya. Jadi, secara fungsional sistem sudah dapat menghasilkan keluaran yang diharapkan.

### 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 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 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 8 pernyataan dengan menggunakan skala 1 sampai 4 (skala likert).

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

$$P = \frac{S}{skorideal} x100\%$$

Keterangan :

P = Nilai persentase yang dicari

S = Jumlah frekuensi dikalikan dengan nilai yang di tetapkan jawaban

skorideal = Nilai tertinggi dikalikan dengan jumlah sampel

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

1. Apakah anda setuju cara mengoperasikan *game* dan *controller* menggunakan *mouse* dan *keyboard* pada *game* ini mudah digunakan?

Tabel 4.7 Perhitungan persentase pertanyaan ke-2

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 1 | Sangat Setuju | 4 | 20 | 80 |
| | Setuju | 3 | 7 | 21 |
| | Ragu-ragu | 2 | 3 | 6 |
| | Tidak Setuju | 1 | - | - |
| **Jumlah** | | | **30** | **107** |

$$P = \frac{107}{120} x100 = 89.9\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap cara pengoprasian dan controller pada *game Alien Hunter* mudah digunakan adalah 89.9% dari yang diharapkan 100%, atau dapat dikategorikan mudah.

2. Apakah anda setuju *interface* (antarmuka) *game* ini menarik??

Tabel 4.8 Perhitungan persentase pertanyaan ke-4

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

$$P = \frac{103}{120} x100 = 85.8\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap antarmuka *game Alien Hunter* adalah 85.8% dari yang diharapkan 100%, atau dapat dikategorikan menarik.

3. Apakah anda setuju pergerakan musuh dalam menemukan *player* dan kemampuan musuh memilih keputusan pergerakan sudah terlihat didalam *game*?

Tabel 4.9 Tabel perhitungan persentase pertanyaan ke-3

| **Pertanyaan** | **Keterangan** | **Skor** | **Frekuensi** | **S** |
|---|---|---|---|---|
| 3 | Sangat Setuju | 4 | 13 | 52 |
| | Setuju | 3 | 12 | 36 |
| | Ragu-ragu | 2 | 5 | 10 |
| | Tidak Setuju | 1 | - | - |
| **Jumlah** | | | **30** | **98** |

$$P = \frac{98}{120} x100 = 81.6\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap pergerakan musuh menemukan target dan kemampuan musuh memilih keputusan pergerakan pada *game Alien Hunter* adalah 81,6% dari yang diharapkan 100%, atau dapat dikategorikan terlihat didalam *game*.

4. Apakah anda setuju *abilites* (kemampuan) musuh untuk mengambil opsi keputusan membuat *game* ini lebih menarik dan menantang untuk dimainkan?

Tabel 4.10 Perhitungan persentase pertanyaan ke-4

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 4 | Sangat Setuju | 4 | 13 | 52 |
| | Setuju | 3 | 10 | 30 |
| | Ragu-ragu | 2 | 7 | 14 |
| | Tidak Setuju | 1 | - | - |
| **Jumlah** | | | **30** | **96** |

$$P = \frac{96}{120} x100 = 80\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap kemampuan musuh mengambil opsi keputusan didalam *game Alien Hunter* adalah 80% dari yang diharapkan 100%, atau dapat dikategorikan sebagai menarik dan menantang untuk dimainkan .

5. Apakah anda setuju *pathfinding* (pencarian jalan) pada musuh untuk bergerak ke arah *player* dengan cepat sudah berjalan dengan baik?

Tabel 4.11 Perhitungan persentase pertanyaan ke-5

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 5 | Sangat Setuju | 4 | 13 | 52 |
| | Setuju | 3 | 8 | 24 |
| | Ragu-ragu | 2 | 9 | 18 |
| | Tidak Setuju | 1 | - | - |
| **Jumlah** | | | **30** | **94** |

$$P = \frac{94}{120} x100 = 78.3\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap *pathfinding* pada *game Alien Hunter* adalah 78,3% dari yang diharapkan 100%, atau dapat dikategorikan berjalan dengan baik.

6. Apakah anda setuju *game* ini sudah memiliki karakter dan grafik 3D yang bagus dan membuat *game* lebih menarik?

Tabel 4.12 Perhitungan persentase pertanyaan ke-8

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 6 | Sangat Setuju | 4 | 16 | 64 |
| | Setuju | 3 | 10 | 30 |
| | Ragu-ragu | 2 | 4 | 8 |
| | Tidak Setuju | 1 | - | - |
| **Jumlah** | | | **30** | **102** |

$$P = \frac{102}{120} x100 = 85\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap karakter dan grafik 3D *game Alien Hunter* adalah 85% dari yang diharapkan 100%, atau dapat dikategorikan bagus dan membuat game lebih menarik.

7. Apakah anda setuju info petunjuk disetiap level sudah cukup jelas dan membantu dalam memainkan *gam*e ini?

Tabel 4.13 Perhitungan persentase pertanyaan ke-7

| **Pertanyaan** | **Keterangan** | **Skor** | **Frekuensi** | **S** |
|---|---|---|---|---|
| 7 | Sangat Setuju | 4 | 17 | 68 |
| | Setuju | 3 | 10 | 30 |
| | Ragu-ragu | 2 | 2 | 4 |
| | Tidak Setuju | 1 | 1 | 1 |
| **Jumlah** | | | **30** | **103** |

$$P = \frac{103}{120} x100 = 85.8\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap info petunjuk disetiap level pada *game Alien Hunter* adalah 85.8% dari yang diharapkan 100%, atau dapat dikategorikan sudah jelas dan membantu dalam memainkan *game*.