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 di implementasikan dengan nama The Defend Island dengan *file extenstion*-nya.*exe*.

4.1.1 Implenetasi Perangkat Keras

Perangkat keras yang digunakan untuk menjalankan *game The Defend Island* ini tidak harus dengan komputer yang berspesifikasi tinggi. Adapun spesifikasi perangkat keras yang digunakan untuk implementasi dan pengujian aplikasi *game The Defend Island* adalah sebagai berikut :

Tabel 4.1 Implementasi Perangkat Keras

No	Nama Perangkat	Spesifikasi
1	Processor	Dual Core 1.8 GHz
2	VGA	256 MB
3	RAM	1 GB
4	Harddisk	1 GB
5	Mouse	Standar
6	Keyboard	Starndar
7	Speaker	Standar

Perangkat keras yang digunakan untuk pembangunan *game* adalah sebagai berikut.

1. Processor yang setara dengan kecepatan 2.0 GHz.
2. RAM 2 GB.
3. Harddisk dengan space kosong 5 GB.
4. VGA Card shared 256 MB.
5. Monitor dengan resolusi 1024 x 768 *pixels*.

6. *Speakers*.
7. *Keyboard* dan *mouse*

Perangkat lunak yang digunakan untuk mengimplementasikan game adalah sebagai berikut.

1. Sistem Operasi Microsoft Windows 7 Ultimate 32bit.
2. Game engine unity 3D, dengan bahasa pemrograman berupa c# dan *javascript*.
3. *Rendering* gambar 3D menggunakan *3Dstudio Max*, dan gambar 2D menggunakan *Adobe Photoshop CS3*, sedangkan untuk suara atau atau *sound effect* menggunakan *Adobe After Effect*.
4. *DirextX* 9.

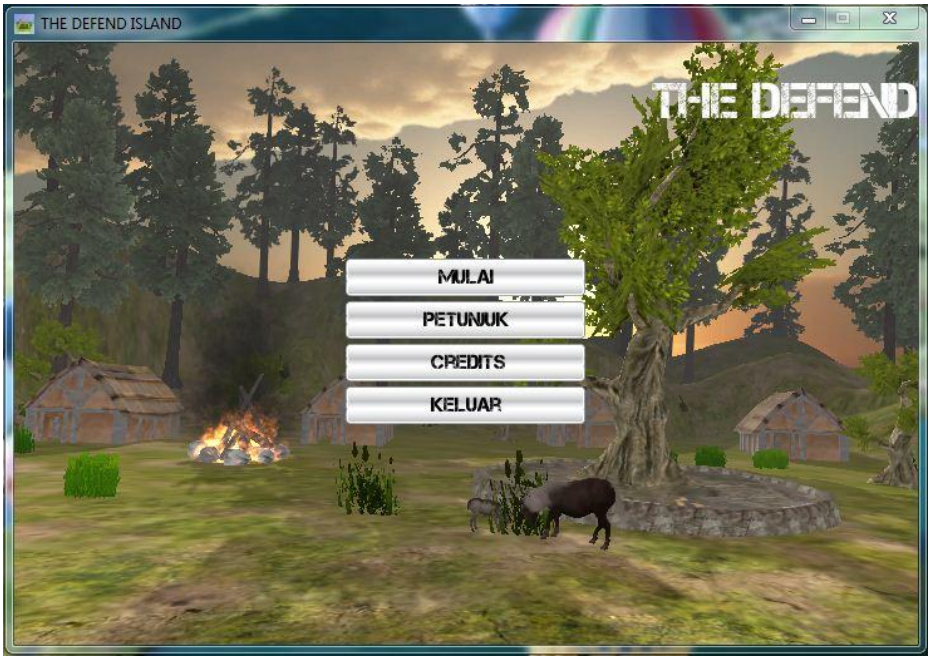
Sistem operasi harus sudah terpasang pada komputer agar game yang dibangun dapat dijalankan pada komputer tersebut. Jika pengguna ingin membuka *project*-nya maka terlebih dahulu harus memasang Unity 3D dan komponen-komponen yang mendukunya. *File TheDefendIsland.exe* dapat digunakan untuk mengimplementasikan game yang telah dibangun.

4.2 Implementasi Antarmuka

Adapun implementasi antarmuka pada aplikasi game The Defend Island adalah sebagai berikut :

1. Antarmuka Menu Utama

Antarmuka menu utama merupakan tampilan menu utama pada saat pertama kali pengguna membuka aplikasi *game*. Implementasi antarmuka untuk perancangan *play game menu* yang dibangun dapat dilihat pada gambar 4.1.



Gambar 4.1 Implementasi *Start Menu*

Di dalam *game* ini terdapat beberapa menu yang terbagi menjadi empat yaitu menu *play game*, *controls*, *credits* dan *quit*. Menu *play game* berguna untuk memulai memainkan *game*, menu *controls* berfungsi untuk melihat halaman bantuan bagi pengguna ketika memainkan *game*, menu *credits* berfungsi untuk menampilkan informasi tentang pengembang game, dan menu *quit* berfungsi ketika pengguna ingin keluar dari permainan.



Gambar 4.2 Implementasi Antarmuka Petunjuk.



Gambar 4.3 Implementasi Antarmuka *Credits*.



Gambar 4.4 Implementasi Antarmuka Keluar.

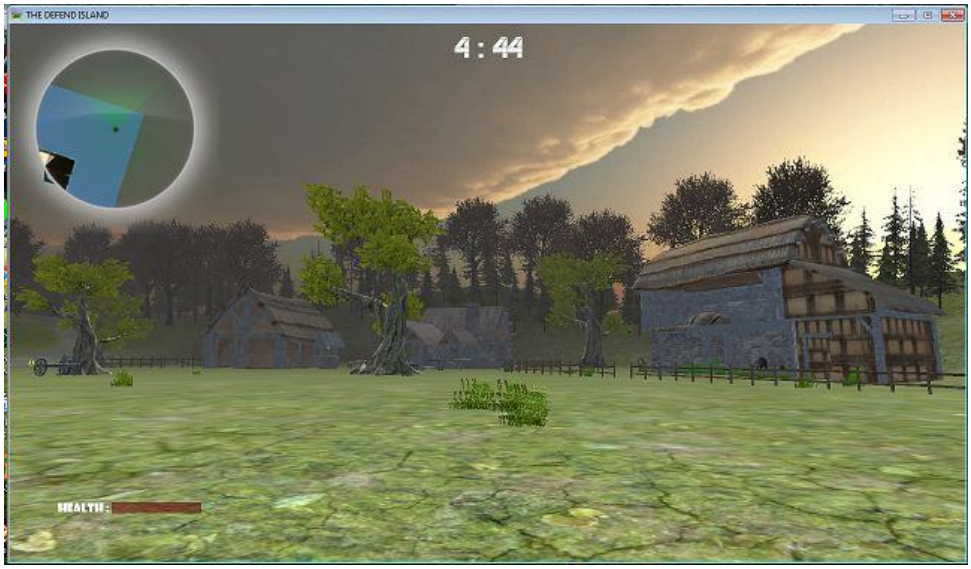
2. Antarmuka Mulai Permainan

Antarmuka mulai permainan merupakan tampilan ketika pengguna mengklik tombol *play game*. Pertama adalah tampilan dari antarmuka *scene video* awal permainan dapat dilihat pada gambar 4.5.

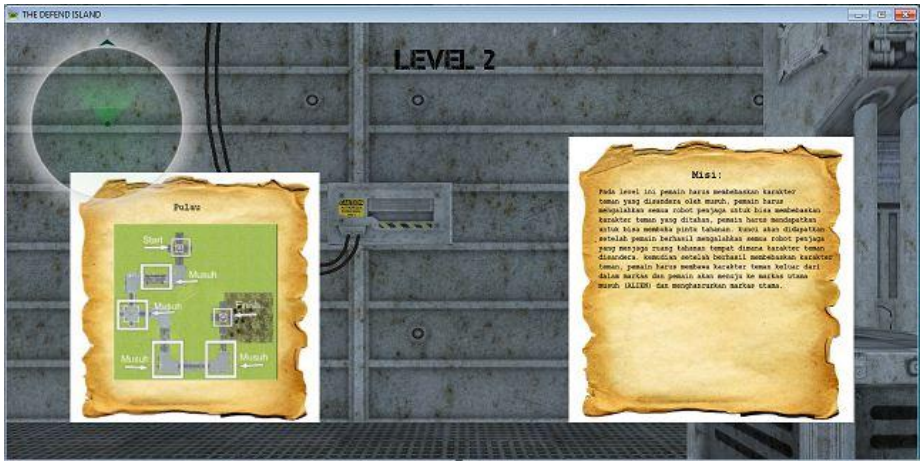


Gambar 4.5 Impelentasi *Scene Video* Level 1

Tampilan antarmuka memulai permainan *game* pada *level* 1 dapat dilihat pada gambar 4.7.



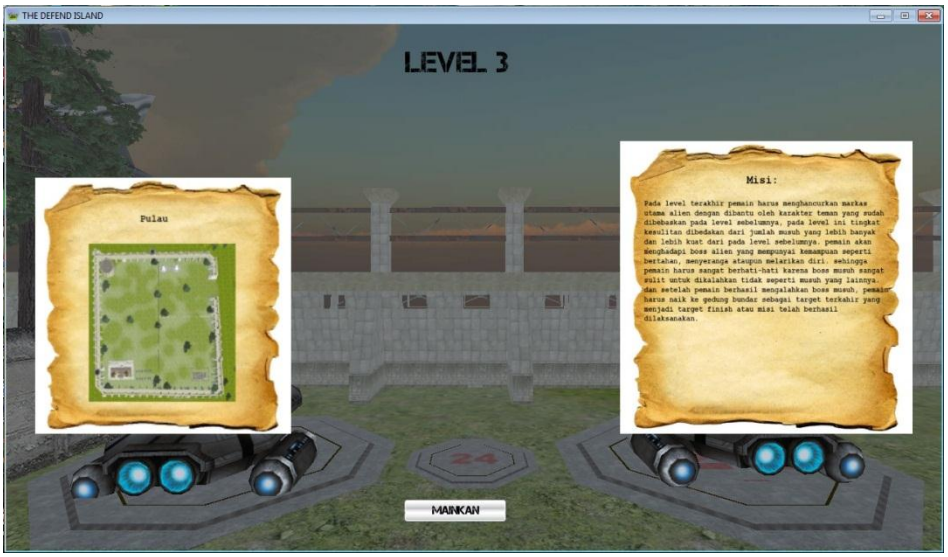
Gambar 4.7 Implementasi Memulai Permainan *Level* 1



Gambar 4.8 *Scene Video Level* 2



Gambar 4.9 Implementasi Memainkan *Level* 2



Gambar 4.10 *Scene Video Level* 3



Gambar 4.11 Impelentasi Memainkan *Level* 3



Gambar 4.12. Implementasi *Mission Complete*



Gambar 4.13 Implementasi *Game Over*

4.3 Pengujian

Pengujian pada sistem yang baru dibangun dilakukan melalui dua tahap yaitu pengujian *alpha* (fungsional) dan pengujian *betha*.

4.3.1 Pengujian *Alpha*

Metode yang digunakan dalam pengujian alpha adalah metode *blackbox* yang berfokus pada persyaratan fungsional dari perangkat lunak yang dibangun.

4.3.1.1 Skenario Pengujian Aplikasi *Game*

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game The Defend Island*. Skenario pengujian dari *game The Defend Island* dapat dilihat pada tabel 4.1

Tabel 4.2 Skenario Pengujian

No.	Komponen Yang Diuji	Skenario Pengujian	Jenis Pengujian
1.	*Main Menu*	Klik Mulai	*Black Box*
		Klik Petunjuk	*Black Box*
		Klik *Credits*	*Black Box*
		Klik Keluar	*Black Box*
2.	*Scene Video*	Tekan Tombol Spasi	*Black Box*
3.	Memulai Permainan	Tekan Tombol ESC	*Black Box*
		Klik Melanjutkan	*Black Box*
		Klik Mengulang *Level*	*Black Box*
		Klik Menu Utama	*Black Box*
		Klik Petunjuk	*Black Box*
		Klik *Credits*	*Black Box*
		Klik Keluar	*Black Box*
		Berhasil Menyelesaikan Misi	*Black Box*
		Gagal Menyelesaikan Misi	*Black Box*
4.	Petunjuk	Klik kembali	*Black Box*

5.	*Credits*	Klik kembali	*Black Box*
6	Misi Selesai	Klik kembali	*Black Box*
6.	*Game Over*	Klik kembali	*Black Box*
7.	Algoritma SMA*	Pengecekan *node* terdekat	*White Box*
8.	Algoritma *Fuzzy Logic*	Perilaku NPC	*White Box*

4.3.1.2 Kasus dan Hasil pengujian (*Black Box*)

Kasus dan hasil uji perangkat lunak yang sudah dibangun dengan menggunakan metode *black box* berdasarkan pengamatan yang dilakukan dan menghasilkan kesimpulan dari pengujian tersebut yang dijelaskan diantaranya sebagai berikut.

1. Pengujian Menu Utama

Pengujian main menu dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.2.

Tabel 4.3 Pengujian *Main Menu*

Kasus dan Hasil Uji (data normal)				
No	**Masukan**	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
1.	Klik tombol Mulai	Memulai permainan dengan menampilkan scene *video* terlebih dahulu	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
2.	Klik tombol Petunjuk	Menampilkan halaman Petunjuk	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
3.	Klik tombol *credits*	Menampilkan halaman *credits*	Tombol berfungsi sesuai dengan yang	[√] Diterima

			diharapkan	[ ] Ditolak
4.	Klik tombol Keluar	Keluar dari aplikasi *game*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

2. Pengujian *Scene Video*

Pengujian scene video dengan kasus danhasil uji, kemudian di amati dan menghasilkan kesimpulan dapat di lihat pada tabel 4.3

Tabel 4.4 Pengujian *Scene Video*

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Tekan tombol spasi	Menghentikan *scene video* dan masuk ke dalam *level* permainan	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

3. Memulai Permainan

Pengujian memulai permainan dengan kasus dan hasil uji, kemudian di amati dan menghasilkan kesimpulan dapat di lihat pada tabel 4.4.

Tabel 4.5 Pengujian Memulai Permainan

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Tekan tombol ESC	Memberhentikan permainan dan menampilkan *pause menu*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
Klik tombol	Menampilkan	Tombol dapat berfungsi	[√] Diterima

melanjutkan	kembali halaman permainan	sesuai dengan yang diharapkan	[ ] Ditolak
Klik tombol mengulang *level*	Mereset ulang halaman permainan kembali seperti pertama kali dimainkan	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
Klik tombol menu utama	Keluar dari permainan dan menampilkan halaman menu utama	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
Klik tombol petunjuk	Menampilkan halaman petunjuk	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
Klik tombol *credits*	Menampilkan halaman *credits*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
Klik tombol keluar	Keluar dari aplikasi *game*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak
Berhasil menyelsaikan misi	Menampilkan halaman misi selesai	Tombol berfungsi	
Gagal menyelesaikan misi	Menampilkan halaman *game over*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

4. Pengujian Petunjuk

Pengujian petunjuk dengan kasus dan hasil uji, kemudian di amati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.5.

Tabel 4.6 Pengujian Petunjuk

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik tombol kembali	Menampilkan kembali halaman sebelumnya	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

5. Pengujian *Credits*

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

Tabel 4.7 Pengujian *Credits*

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik tombol kembali	Menampilkan kembali halaman sebelumnya	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

6. Pengujian *Mission Complete*

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

Tabel 4.8 Pengujian *Mission Complete*

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik tombol *start menu*	Menampilkan kembali halaman *main menu*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

7. Pengujian *Game Over*

Pengujian *game over* dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.8.

Tabel 4.9 Pengujian *Game Over*

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik tombol *back*	Menampilkan kembali halaman *main menu*	Tombol dapat berfungsi sesuai dengan yang diharapkan	[√] Diterima [ ] Ditolak

4.3.1.3 Kasus dan Hasil Pengujian (*White Box*)

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

1. Pengujian *White Box* Algoritma SMA*

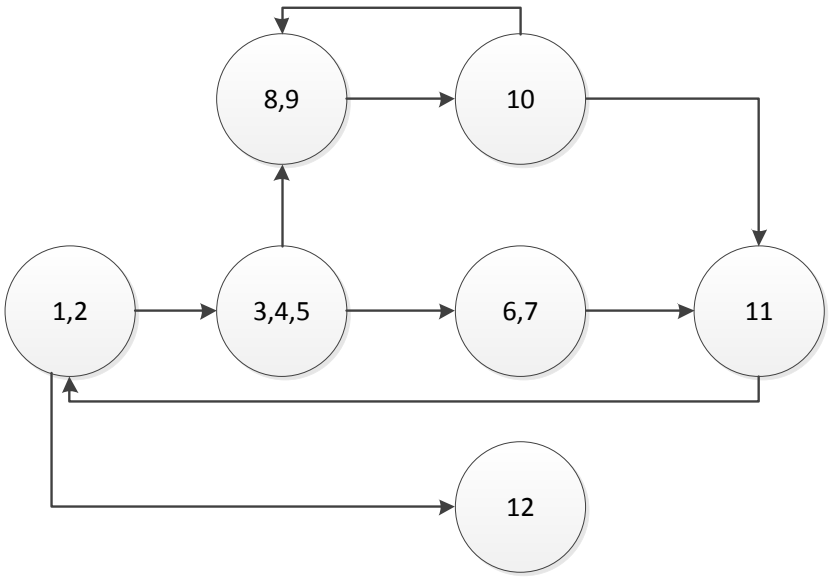
Pengujian algoritma SMA* merupakan proses dimana akan diuji terhadap musuh dalam melakukan pencarian rute terpendek yang terdapat dalam *game The Defend Island*.

A. Prosedur pencarian jalan

Berikut adalah pseudocode prosedur pencarian jalan

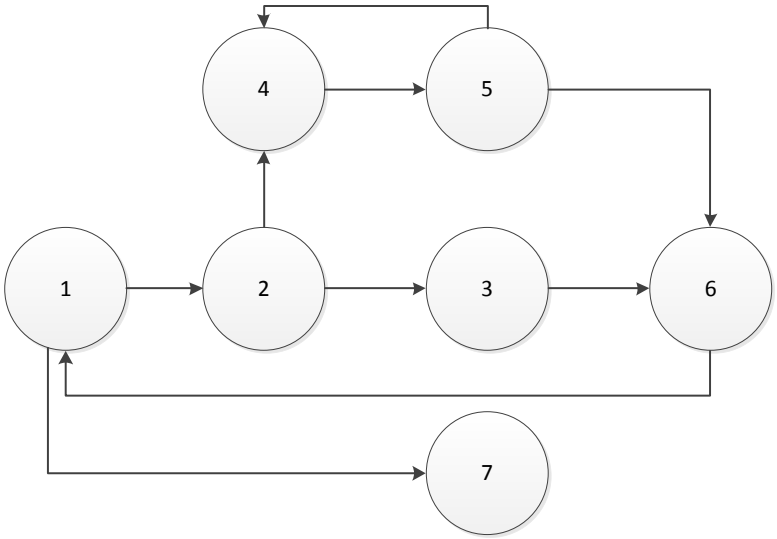
```
1 :  openList.Add(nodeAwal);
2 :  while(foundTarget==false){
3 :          checkingNode = GetSmallestFValueNode();
4 :          pindah checkingNode ke closedList
5 :          if (checkingNode == targetNode)
6 :          then foundTarget = true;
7 :                  TraceBackPath(musuh);
8 :          else for (var node : PathfindNode in checkingNode.dekat)
9 :                  cekNode(checkingNode, node);
10 :             endfor
11 :         endif
12 :  endwhile
```

Berdasarkan *pseudocode* prosedur pencarian jalan, maka dibuat *flow graph* yang bisa dilihat pada gambar 4.14.



Gambar 4.14 Flowgraph prosedur pencarian jalan

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut:



Gambar 4.15 Penyederhanaan Flow Graph pencarian jalan

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

a. *Cyclomatic Complexity* V(G)

E = 9, N = 7

V(G) = E - N + 2

V(G) = 9 – 7 + 2

V(G) = 2 + 2 = 4

Keterangan :

E = Jumlah Aksi

N = Jumlah Kondisi

b. *Independent Path*

Path 1 = 1 - 7

Path 2 = 1-2-3-6-1-7

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

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

c. *Graph Matriks* Algoritma SMA* Pencarian Jalan

N	1	2	3	4	5	6	7	N(E)-1
1		1					1	1
2			1	1				1
3						1		0
4					1			0
5				1		1		1
6	1							0
7								0
							Sum(E)+1	4

Keterangan :

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

d. Keismpulan

Kesimpulan yang didapatkan berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 4, maka dapat disimpulkan bahwa pada proses pencarian dengan algoritma SMA* pada prosedur pencarian jalan berjalan dengan baik ada *game The Defend Island*, karena setiap pengujian menghasilkan nilai yang sama.

e. Hasil Uji Algoritma SMA* Pencarian Jalan

Tabel 4.10 Hasil Uji Algoritma SMA* pencarian jalan

No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
1	foundTarget = true	Pencarian jalan gagal dilakukan	Pencarian jalan tidak dilakukan	[√] Diterima [ ] Ditolak
2	• foundTarget = false • CheckingNode = target node	Pencarian jalan dilakukan dan fungsi TraceBackPath dilakukan	Pencarian jalan dilakukan dan fungsi TraceBackPath dilakukan	[√] Diterima [ ] Ditolak
3	• foundTarget = false • Checking node != target node	Pencarian jalan dilakukan dan pengecekan *node* dilakukan	Pencarian jalan dilakukan dan pengecekan *node* dilakukan	[√] Diterima [ ] Ditolak
4	• foundTarget = false • Checking node != target node • *Node* di checkingNode	Pengecekan *node* dilakukan lagi sebanyak jumlah *node* dekat	Pengecekan *node* dilakukan lagi sebanyak *node* dekat	[√] Diterima [ ] Ditolak

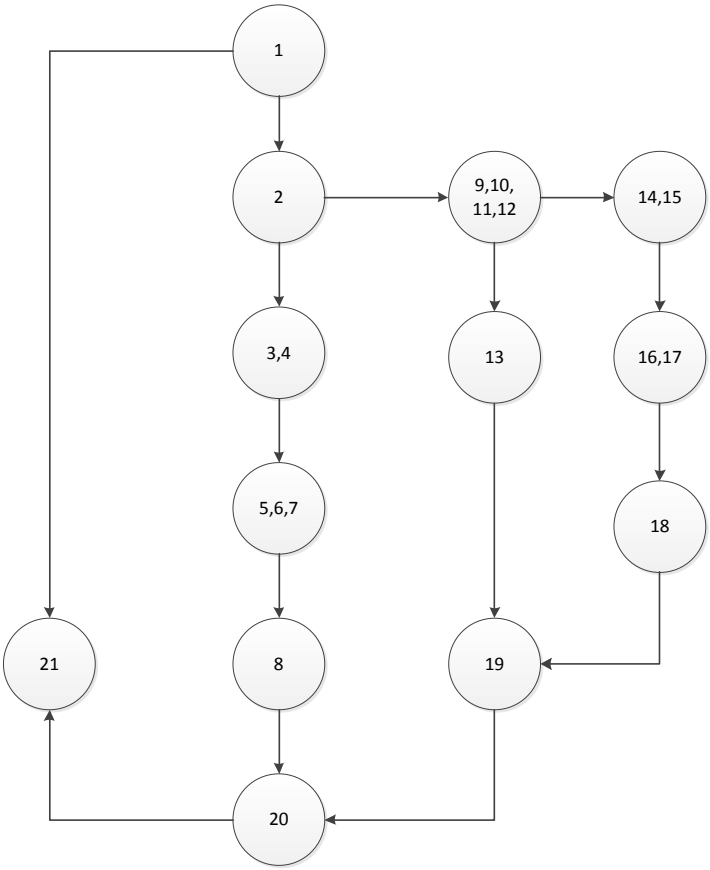
No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
	.dekat masih ada			

B. Prosedur Pengecekan *node* terdekat

Berikut ini adalah *pseudocode* prosedur pengecekan *node* terdekat sebagai berikut :

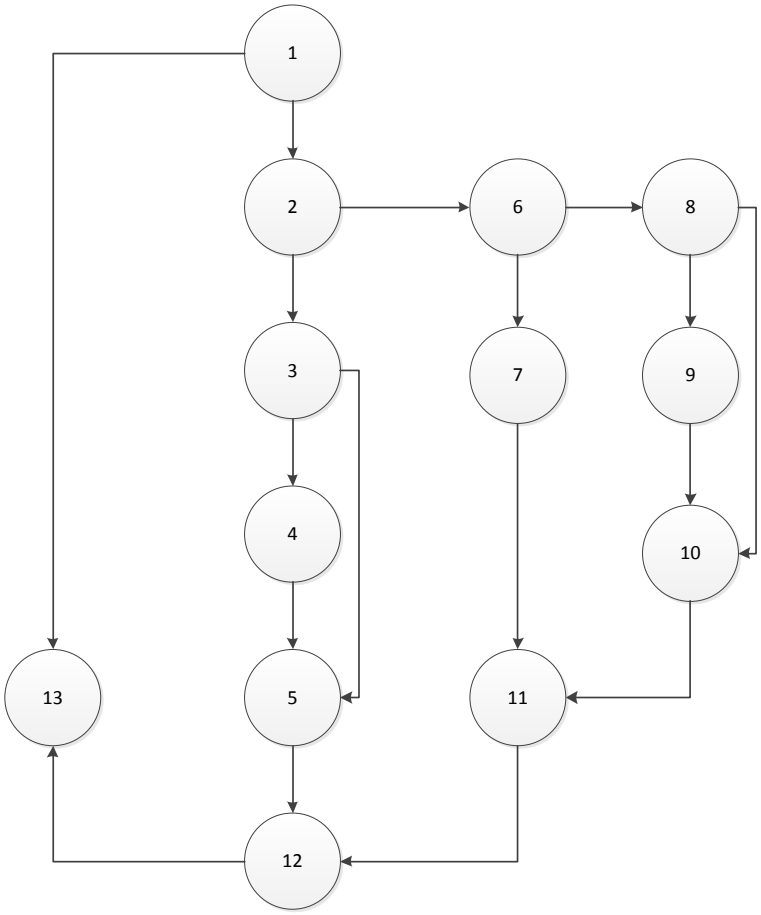
```
1 :  if (closedList.Contains(testing) == false)
2 :  then if (openList.Contains(testing) == true)
3 :       then hitung G cost baru
4 :            if (newGCost < testing.getGValue())
5 :            then testing.setParent(currentNode);
6 :                 testing.setGValue(newGCost);
7 :                 testing.CalculateFValue();
8 :            endif
9 :       else testing.setParent(currentNode);
10:           testing.setGValue(currentNode.getGValue()+baseMovementCost);
11:           testing.CalculateFValue();
12:           if(openList.Count<limit)
13:           then AddToOpenList(testing);
14:           else var test : PathfindNode =GetBiggestFValueNode();
15:                if(testing.getFValue()<=test.getFValue())
16:                then openList.Remove(test);
17:                     AddToOpenList(testing);
18:                endif
19:           endif
20:       endif
21:  endif
```

Berdasarkan *pseudocode* prosedur pengecekan *node* terdekat, maka dibuat *flow graph* yang bisa dilihat pada gambar 4.15.



Gambar 4.15 *FlowGraph procedure* Pengecekan *node* Terdekat

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut :



Gambar 4.16 Penyederhanaan Flow Graph Pengecekan *Node* Terdekat

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

a. *Cyclomatic Complexity* V(G)

E = 16, N = 13

V(G) = E – N + 2

V(G) = 17 – 13 + 2

V(G) = 4 + 2 = 6

Keterangan

E = Jumlah aksi

N = Jumlah Kondisi

b. *Independent Path*

Path 1 = 1-13

Path 2 = 1-2-3-4-5-12-13

Path 3 = 1-2-3-5-12-13

Path 4 = 1-2-6-7-11-12-13

Path 5 = 1-2-6-8-9-10-11-12-13

Path 6 = 1-2-6-8-10-11-12-13

c. *Graph Matriks* Algoritma SMA* Pengecekan *Node* terdekat

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

Keterangan :

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

d. Hasil Uji Algoritma SMA*

Tabel 4.11 Hasil Uji Algoritma SMA* Pengecekan Node Terdekat

No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
1	closedList.Contains(testing) = true	Pengecekan *node* tidak dilakukan	Pengecekan *node* gagal dilakukan	[√] Diterima [ ] Ditolak
2	• closedList.Contains(testing) = false • openList.Contains(testing) = true • newGCost > testing.getGValue()	Hanya dilakukan perhitungan nilai g *cost* baru	Hanya dilakukan perhitungan nilai g *cost* baru	[√] Diterima [ ] Ditolak
3	• closedList.Contains(testing) = false • openList.Contains(testing) = true • newGCost < testing.getGValue()	Dilakukan perhitungan nilai g *cost* baru dan dilakukan penggantian *parent* dengan g *cost* yang lebih baik	Perhitungan nilai g *cost* baru dilakukan dan penggantian *parent* dengan g *cost* yang lebih baik	[√] Diterima [ ] Ditolak
4	• closedList.Contains(testing) = false • openList.Contains(testing) = false • openList.Count < limit	Dilakukan perhitungan nilai g dan f serta pemindahan *node* ke *open list*	Terjadi pemindahan *node* ke *open list* dengan perhitungan nilai g dan f-nya	[√] Diterima [ ] Ditolak

No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
5	• closedList.Contains(testing) = false • openList.Contains(testing) = false • openList.Count > limit • testing.getFValue() > test.getFValue()	Dilakukan perhitungan nilai g dan f serta mengambil *node* dengan nilai terburuk di *open list*	Dilakukan perhitungan nilai g dan f serta mengambil *node* dengan nilai terburuk di *open list*	[√] Diterima [ ] Ditolak
6	• closedList.Contains(testing) = false • openList.Contains(testing) = false • openList.Count > limit • testing.getFValue() <= test.getFValue()	Dilakukan perhitungan nilai g dan f serta mengambil *node* dengan nilai terburuk di *open list* lalu dilakukan penghapusan *node* dengan nilai terburuk	Dilakukan perhitungan nilai g dan f serta mengambil *node* dengan nilai terburuk di *open list* lalu dilakukan penghapusan *node* dengan nilai terburuk	[√] Diterima [ ] Ditolak

f. Kesimpulan

kesimpulan yang didapatkan berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Comlexity* yang sama yaitu 6, maka dapat disimpulkan bahwa pada proses pencarian algoritma SMA* pada prosedur pengecekan node terdekat berjalan dengan baik pada *game The Defend Island*, karena setiap pengujian menghasilkan nilai yang sama.

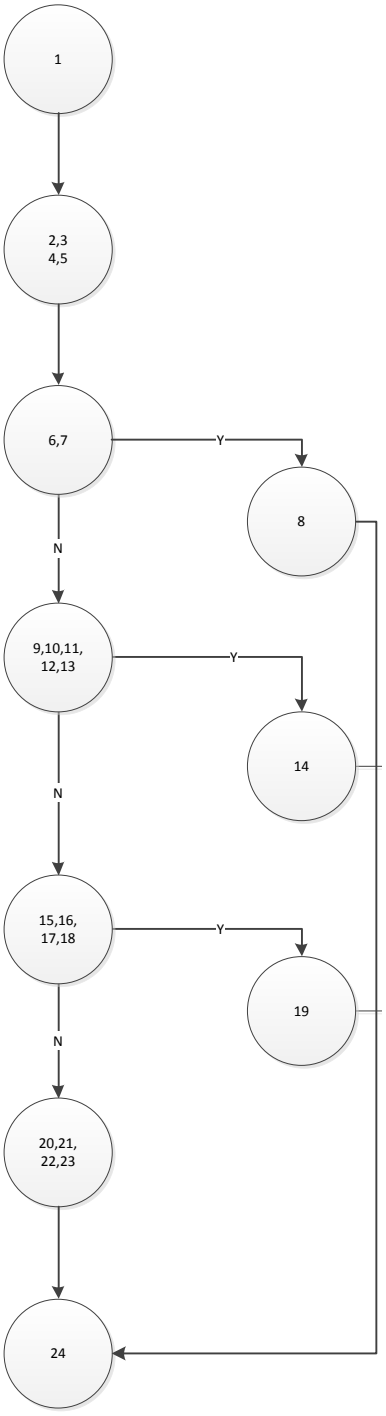
2. Pengujian *White Box* Algoritma *Fuzzy Logic*

Berikut ini merupakan pengujian algoritma fuzzy logic yang diterapkan pada *game The* Defend *Island.*

```
1.  nilaiJarak = jrk;
2.  fungsiKeanggotaanTrapesium ftrapesium = new
     fungsiKeanggotaanTrapesium();
3.  fungsiKeanggotaanSegitiga fsegitiga = new
     fungsiKeanggotaanSegitiga();
4.  nilaiKekuatan = kkt;
5.  fungsiKeanggotaanTrapesium ftrapesium = new
     fungsiKeanggotaanTrapesium();
6.  return nilaiImplikasi = Mathf.Min(n1, n2);
7.  implikasi imp = new implikasi();
8.  jarak jrk = new jarak(nilaiJarak);
9.  kekuatan kkt = new kekuatan(nilaiKekuatan);
10. jrk.periksaJarak();
11. kkt.periksaKekuatan();
12. jarak = (target.position - myTransform.position).magnitude;
13. rules rule = new rules(jarak,hitPoints);
14. defuzzyfikasi def = new defuzzyfikasi();
15. rule.periksaRule();
16. def. hasilDefuzzy();
17. if((balikKeMarkas.position - myTransform.position).magnitude
     < 1.0){
18. StartCoroutine(tambahDarah());}
19. else if(def.hasilDefuzzyfikasi <= 12.0f) {
20. balikKeMarkas();
21. myTransform.position += myTransform.forward * moveSpeed *
     Time.deltaTime;
22. anim.animation.CrossFade(walkAnim);
23. playWalkSounds();}
24. else if(def.hasilDefuzzyfikasi > 8.0f &&
     def.hasilDefuzzyfikasi <= 22.0f){
25. bertahan();
26. anim.animation.CrossFade(attackAnim2);
27. audio.Stop();}
28. else if(def.hasilDefuzzyfikasi > 18.0f &&
     def.hasilDefuzzyfikasi <= 30.0f){
29. menyerang();
30. damage = Random.Range(20, 30);
31. anim.animation.CrossFade(walkAnim);}
```

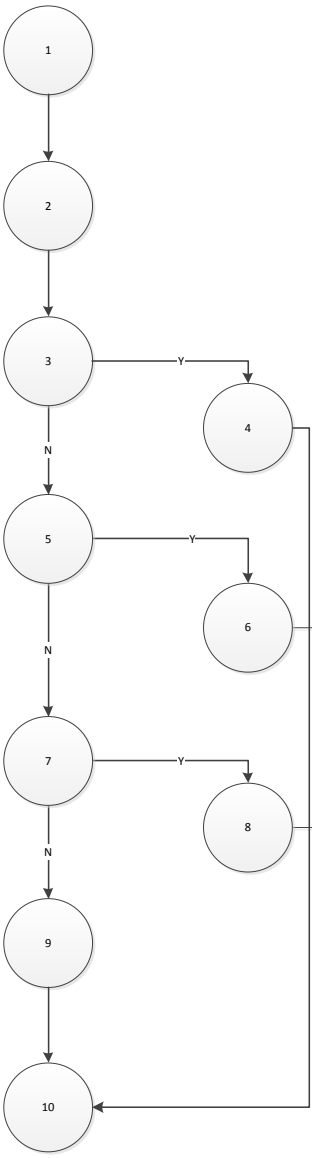
a. *FlowGraph* Algoritma *Fuzzy Logic*

Berdasarkan pseudocode diatas maka flow graph algoritma fuzzy logic adalah sebagai berikut :



Gambar 4.17 Flow Graph Algoritma *Fuzzy Logic*

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut:



Gambar 4.18 Penyederhanaan Flow Graph Algoritma *Fuzzy Logic*

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

b. *Cyclomatic Comlexity* V(G)

Cyclomatic complexity akan digunakan dalam menguji path yang nantinya akan digunakan untuk mencari jumlah path dalam satu dan juga sebagai penentuan jalur atau path pencarian NPC terhadap *player*. Apabila menggunakan basis path, nilai yang dihitung untuk *Cyclomatic complexity* menentukan jalur indpenden dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali.

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

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*

Independent path adalah jalur yang melintas atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru.

Path 1 = 1-2-3-5-7-9-10

Path 2 = 1-2-3-4-10

Path 3 = 1-2-3-5-6-10

Path 4 = 1-2-3-4-7-8-10

d. *Graph Matriks* Algoritma *Fuzzy Logic*

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

Keterangan :

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

e. Hasil Uji Algoritma Fuzzy Logic

Tabel 4.12 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*) 2(*void*) 3(*void*) 4(*void*) 12(*void*)	NPC menambah *hitpoints*	*Path* dikerjakan

Kasus dan Hasil Uji				
No. *Path*	**Butir uji algoritma *fuzzy logic***	***Node* (n), input kondisi**	**Pengamatan**	**Kesimpulan**
2	Hasil *defuzzy* <= 10 NPC melarikan diri ke markas	1,(*void*) 2(*void*) 3(*void*) 5(*void*) 6(*void*) 12(*void*)	NPC melarikan diri ke markas	*Path* dikerjakan
3	Hasil *defuzzy* > 10 dan <= 20 NPC bertahan	1,(*void*) 2(*void*) 3(*void*) 5(*void*) 7(*void*) 8(*void*) 12(*void*)	NPC bertahan	*Path* dikerjakan
4	Hasil *defuzzy* > 20 dan <= 30 NPC	1,(*void*) 2(*void*)	NPC menyerang	*Path* dikerjakan

Kasus dan Hasil Uji				
No. *Path*	**Butir uji algoritma *fuzzy logic***	***Node* (n), input kondisi**	**Pengamatan**	**Kesimpulan**
	menyerang	3(*void*) 5(*void*) 7(*void*) 9(*void*) 10(*void*) 12(*void*)		

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 sehingga algoritma yang di implementasikan pada game sesuai dengan yang diharapkan.

4.3.1.3 Kesimpulan Pengujian *Alpha*

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa sistem sudah berjalan dengan baik pada *game The Defend Island* 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.

4.4.1 Pengujian Beta

Pengujian *beta* merupakan pengujian yang dilakukan secara objektif, dimana pengujian langsung melibatkan para pengguna aplikasi permainan secara langsung dengan menggunakan kuisioner mengenai kepuasan para pengguna terhadap aplikasi *game* yang dibangun dengan kandungan *point* syarat.

Berdasarkan hasil pengujian beta, dicari persentase masing-masing jawaban dengan menggunakan rumus :

$$P=\Sigma s/\text{skor ideal}*100\% \quad (4.1)$$

Keterangan :

Keterangan:

P = Nilai presentase yang dicari

Σs = Jumlah skor yang didapatkan dari setiap pilihan jawaban dikalikan dengan nilai yang telah ditetapkan sebelumnya

Skor ideal = Nilai tertinggi pada *point* pilihan dikalikan dengan jumlah responden.

Berikut ini adalah hasil presentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan persamaan 4.1 diatas. Kuesioner ini telah di ujikan pada 30 responden. Hasil perhitungan presentase masing-masing jawaban adalah sebagai berikut :

Untuk pertanyaan No 1

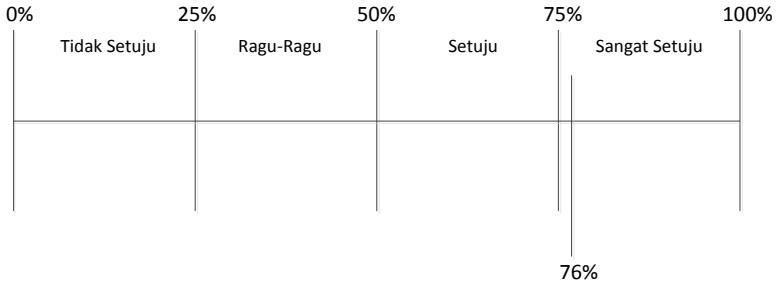
"Apakah anda setuju *game The Defend Island* menarik"

Tabel 4.13 Hasil Pengujian *Beta* Pertanyaan No 1

Kategori Jawaban	Skor	Responden	S
Sangat Setuju	4	9	36
Setuju	3	14	42
Ragu-ragu	2	7	14
Tidak Setuju	1	0	0
JUMLAH		**30**	**92**

P = 92 / 120 * 100% = 76%

Berikut adalah kriteria interpretasi skor :



Gambar 4.19 InterpretasiSkor Pertanyaan No 1

Berdasarkan hasil presntase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap *game The Defend Island* menarik adalah 76% dari yang diharapkan 100%, atau dapat dikategorikan sangat setuju.

Untuk pertanyaan No 2

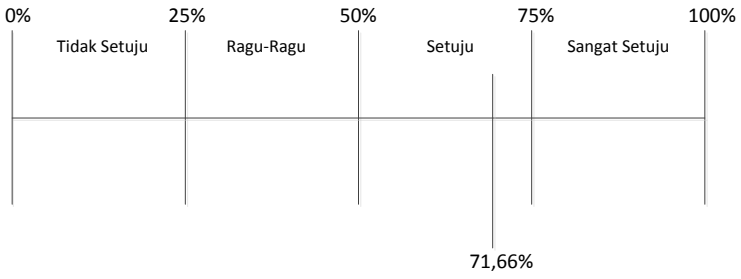
"Apakah anda setuju *interface* (antarmuka) pada *game The Defend Island* menarik"

Tabel 4.13 Pengujian Beta No 2

Kategori Jawaban	**Skor**	**Responden**	**S**
Sangat Setuju	4	5	20
Setuju	3	16	48
Ragu-ragu	2	9	18
Tidak Setuju	1	0	0
JUMLAH		**30**	**86**

P = 86 /120 * 100% = 71,66%

Berikut adalah kriteria interpretasi skor :



Gambar 4.20 Interpretasi Skor Pertanyaan No 2

Berdasarkan hasil presentase diatas, maka dapat disimpulkan bahwa penilaian terhadap *game The Defend Island* mempunyai antarmuka yang menarik adalah 71,66% dari yang diharapkan 100%, atau dapat dikategorikan setuju.

Untuk pertanyaan No 3

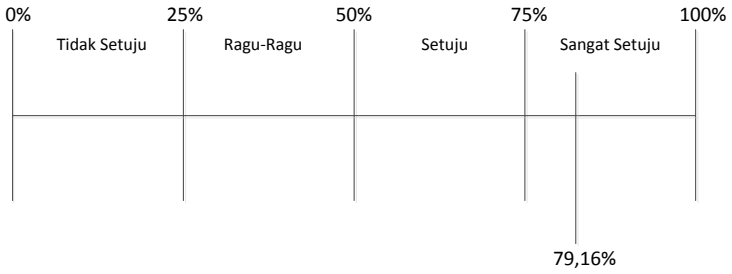
"Apakah anda setuju pergerakan musuh dalam menemukan *player* dan kemampuan memilik opsi keputusan pergerakan sudah terlihat pada di dalam *game The Defend Island*"

Tabel 4.14 Pengujian Beta No 3

Kategori Jawaban	Skor	Responden	S
Sangat Setuju	4	10	40
Setuju	3	15	45
Ragu-ragu	2	5	10
Tidak Setuju	1	0	0
JUMLAH		**30**	**95**

P = 95 / 120 * 100% = 79,16%

Berikut adalah kriteria interpretasi skor :



Gambar 4.21 Interpretasi Skor Pertanyaan No 3

Berdasarkan hasil presentase dari nilai diatas, maka dapat disimpulkan bahwa penerapan kecerdasan buatan yang ada pada musuh adalah 79,16% setuju dari 100% yang diharapkan, atau dapat dikatakan sangat setuju.

Untuk pertanyaan No 4

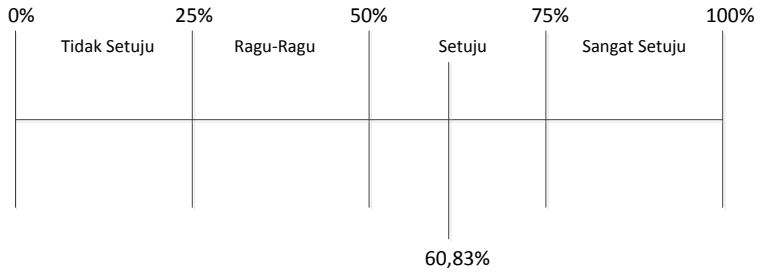
"Apakah anda setuju jenis musuh yang ada pada *game The Defend Island* bervariatif..?"

Tabel 4.15 Pengujian Beta No 4

Kategori Jawaban	Skor	Responden	S
Sangat Setuju	4	3	12
Setuju	3	12	36
Ragu-ragu	2	10	20
Tidak Setuju	1	5	5
JUMLAH		**30**	**73**

P = 73 /120* 100% = 60,83%%

Berikut adalah kriteria interpretasi skor :



Gambar 4.22 Interpretasi Skor Pertanyaan No 4

Berdasarkan hasil dari presentase diatas, maka dapat disimpulkan bahwa penilaian terhadap variasi musuh pada *game The Defend Island* pada setiap *level*-nya adalah 60,83% dari yang diharapkan 100%, atau dapat dikatakan setuju.

Untuk pertanyaan No 5

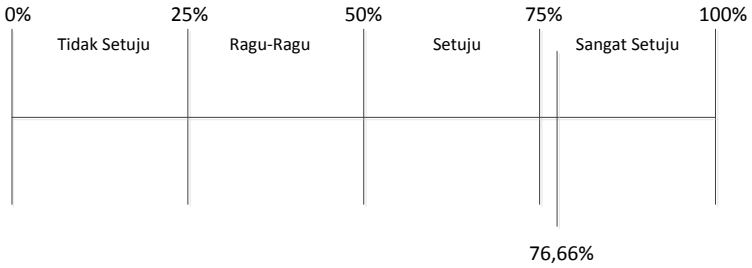
"Apakah anda setuju alur cerita dan *gameplay* pada *game The Defend Island* menarik"

Tabel 4.16 Pengujian Beta No 5

Kategori Jawaban	Skor	Responden	S
Sangat Setuju	4	7	28
Setuju	3	18	54
Ragu-ragu	2	5	10
Tidak Setuju	1	0	0
JUMLAH		**30**	**92**

P = 92 / 120 * 100% = 76,66%

Berikut adalah kriteria interpretasi skor :



Gambar 4.23 Interpretasi Skor Pertanyaan No 5

Berdasarkan hasil presentase diatas, maka dapat disimpulan bahwa penilaian terhadap alur cerita dan *gameplay* pada *game The Defend Island* adalah76,66% dari yang diharapkan 100%, atau dapat dikategorikan sangat setuju.

Untuk pertanyaan No 6

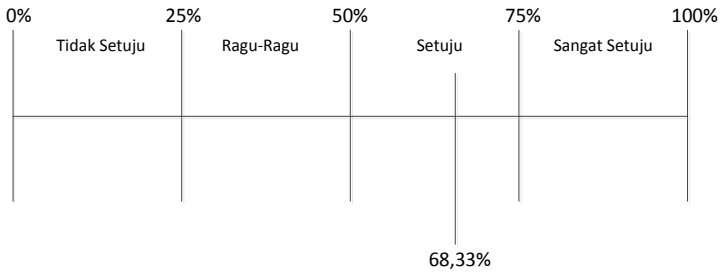
"Apakah anda setuju kemampuan musuh untuk mengambil opsi keputusan membuat game ini lebih menarik dan menantang untuk dimainkan"

Tabel 4.17 Pengujian Beta No 6

Kategori Jawaban	Skor	Responden	S
Sangat Setuju	4	4	16
Setuju	3	17	51
Ragu-ragu	2	6	12
Tidak Setuju	1	3	3
JUMLAH		**30**	**82**

P = 82/120 * 100% = 68,33%

Berikut adalah kriteria interpretasi skor :



Gambar 4.24 Interpretasi Skor Pertanyaan No 6

Berikut adalah hasil dari presentase diatas, maka penilaian apakah musuh dapat memilik opsi keputusan dalam melakukan pergerakan adalah 68,33% dari yang diharapkan 100%, maka dapat dikatakan sangat setuju.

Untuk pertanyaan No 7

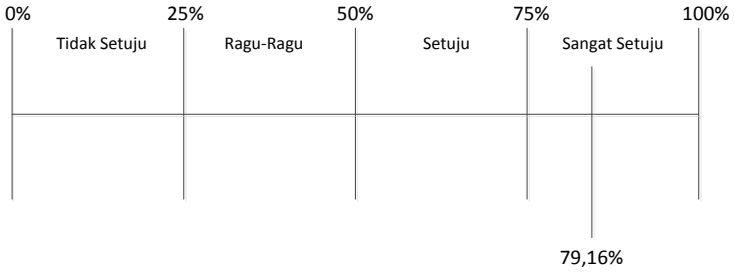
"Apakah anda setuju info petunjuk permainan setiap *level* sudah cukup jelas dan membantu dalam memainkan *game* ini"

Tabel 4.18 Pengujian Beta No 7

Kategori Jawaban	**Skor**	**Responden**	**S**
Sangat Setuju	4	10	40
Setuju	3	15	45
Ragu-ragu	2	5	10
Tidak Setuju	1	0	0
JUMLAH		**30**	**95**

P = 95/120 * 100 = 79,16%

Berikut adalah kriteria interpretasi skor :



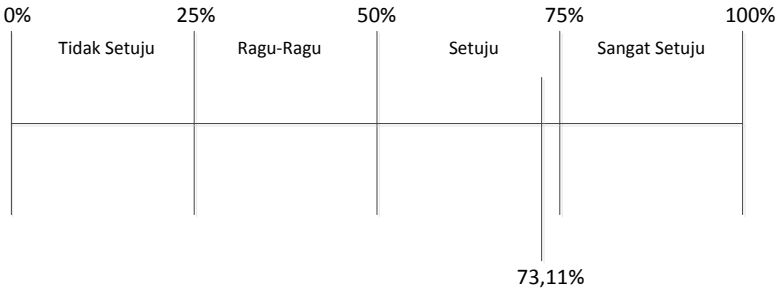
Gambar 4.25 Interpretasi Skor Pertanyaan No 7

Berdasarkan hasil dari penelitian diatas, maka dapat disimpulkan bahwa penilaian terhadap informasi misi dalam permainan dan setiap level nya sudah cukup jelas adalah 79,16% dari yang diharapkan 100%, maka dapat dikategorikan sangat setuju untuk pentujuk dan indormasi pada permainan.

4.4.2 Kesimpulan *Beta*

Berdasarkan dari jawaban yang diberikan oleh responden, bisa ambil kesimpulan bahwa "*Game Action The Defend Island*" dinyatakan menarik untuk untuk dimainkan sebesar 76%, untuk desain *interface* yang ada *pada game the defend island* memiliki persentase sebesar 71,66%, untuk pergerakan musuh dalam menemukan *player* serta kemampuan dalam memilih keputusan pergerakan pada *game the defend island* adalah 79,16%, komponen musuh yang bervariatif pada *game The Defend Island* memiliki persentase sebesar 60,83%, untuk *gameplay* dan alur cerita pada *game The Defend Island* memiliki bobot persentase sebesar 76,66, kemampuan musuh dalam mengambil keputusan pada *game The Defend Island* apakah sudah menarik dan menantang untuk dimainkan sebesar 68,33%, dan info atau pentujuk dalam *game The Defend Island* pada tiap *level* sudah cukup jelas dan membantu dalam memainkan *game* ini sebesar 79,61%.

Dari hasil perhitungan kuesioner, maka *game The Defend Island* menghasilkan nilai rata-rata 73.11% atau dikatakan sangat setuju dari 7 pertanyaan yang diperoleh dari responden terhadap game yang dibangun. Berikut adalah kriteria interpretasi skor nilai rata-rata sebagai berikut :



Gambar 4.26 Interpretasi Skor Nilai Rata-rata