BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

IV.1 Implementasi Sistem

Implementasi sistem adalah tahap pengujian kelayakan pada sistem yang dibuat sehingga aplikasi yang dibuat tidak menyimpang dari perancangan sistemnya. Game ini dibuat menggunakan Construct 2 R95 sehingga game ini merupakan game berbasiskan HTML 5. Implementasi game kedalam *smartphone* berbasiskan android menggunakan PhoneGap sebagai *tool*-nya. Perangkat keras yang digunakan untuk mengimplimentasikan dan menguji game ini memiliki spesifikasi sebagai berikut :

1. *Processor* dengan kecepatan 2,3GHz
2. Memori 2GB
3. *Harddisk* 250GB
4. *Video Card* dengan memori 256MB
5. Monitor
6. *Mouse* dan *Keyboard*
7. *Speaker*

Sedangkan perangkat lunak yang digunakan untuk mengimplimentasikan dan menguji game ini adalah :

Windows 7 Ultimate sebagai sistem operasi

Construct 2 R95 sebagai perangkat lunat pembangunan game

PhoneGap sebagai *tools* untuk mengimplementasi game kedalam *smartphone*

IV.1.1. Implementasi Antar Muka

Implementasi antar muka adalah bagian yang menunjukan bentuk tampilan setiap antarmuka pada aplikasi yang sudah dibangun. Implementasi antar muka pada game ini ditunjukan pada Gambar IV- 1 sampai dengan Gambar IV- 2.

1. Antarmuka Menu Utama

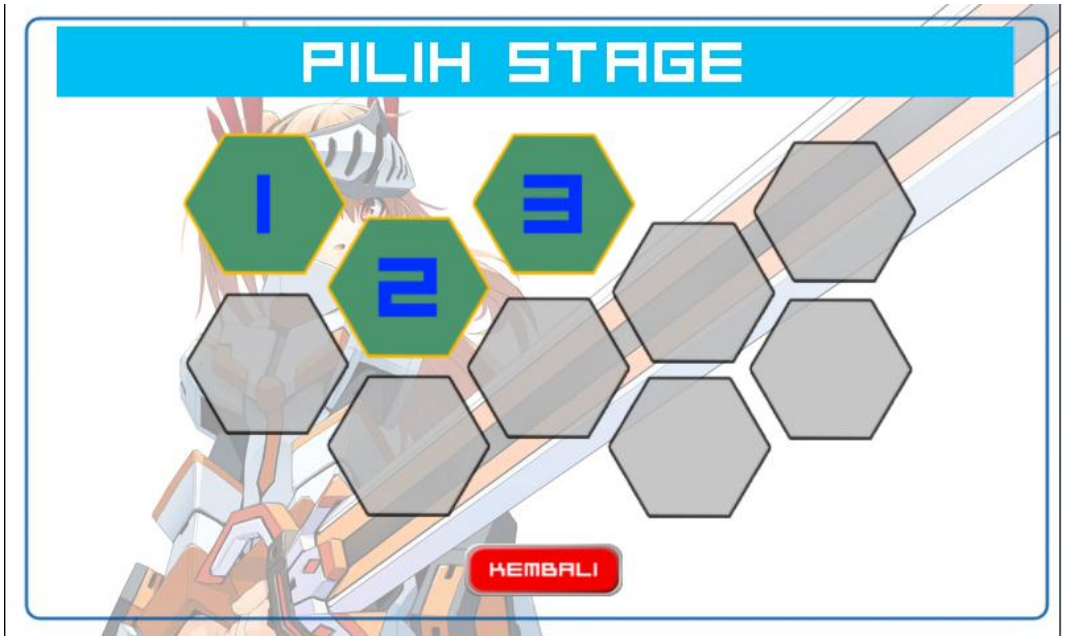
   Antarmuka menu utama ditunjukan pada Gambar IV-1. Tombol yang tersedia adalah tombol main yang berfungsi sebagai akses menuju menu main dan tombol petunjuk main yang berfungsi sebagai akses menuju menu petunjuk main.



Gambar IV- 1 Antarmuka Menu Utama

2. Antarmuka Menu Main

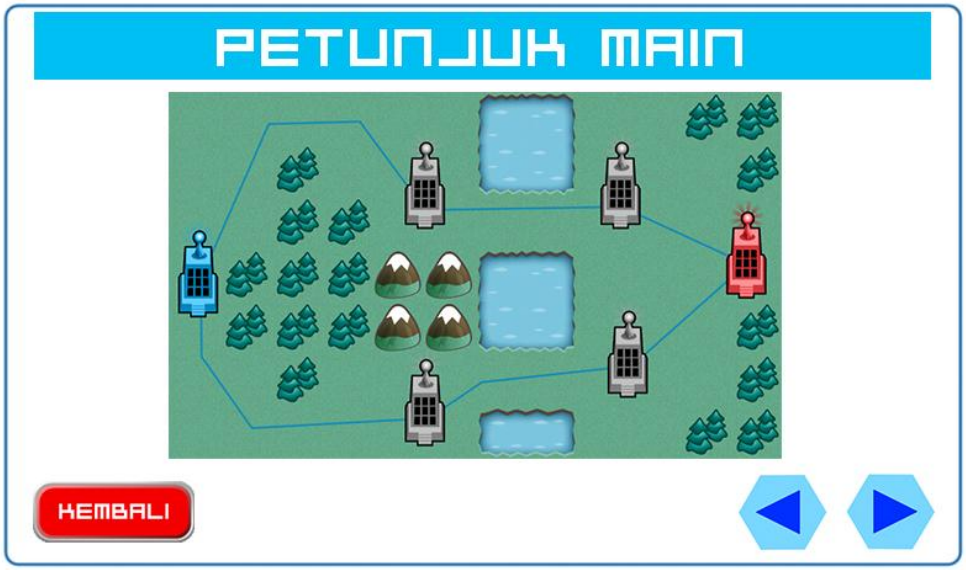
   Antarmuka Menu Main ditunjukan pada Gambar IV-2 .Tombol yang tersedia adalah tombol 1 sebagai akses menuju stage 1, tombol 2 sebagai akses menuju satge 2, tombol 3 sebagai akses menuju stage 3 dan tombol kembali untuk kembali ke menu utama.



Gambar IV- 2 Antarmuka Menu Main

3. Antarmuka Petunjuk Main

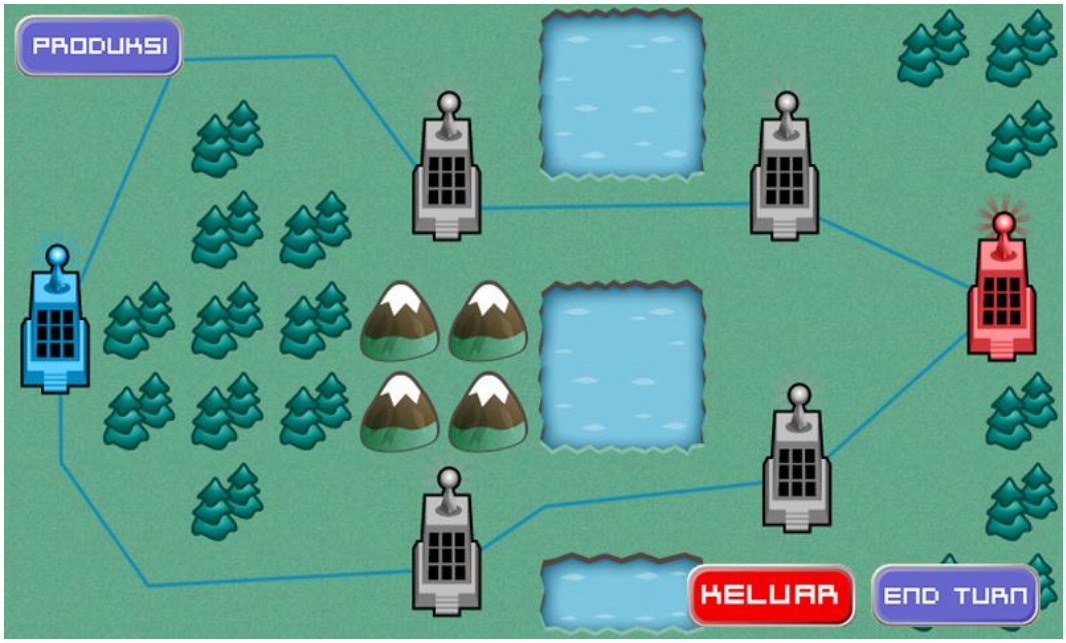
   Antarmuka petunjuk main ditunjukan pada Gambar IV-3. Tombol yang tersedia adalah tombol ► sebagai tombol untuk melihat petunjuk selanjutnya, tombol ◄ sebagai tombol untuk melihat petunjuk sebelumnya dan tombol kembali sebagai tombol untuk kembali ke menu utama.



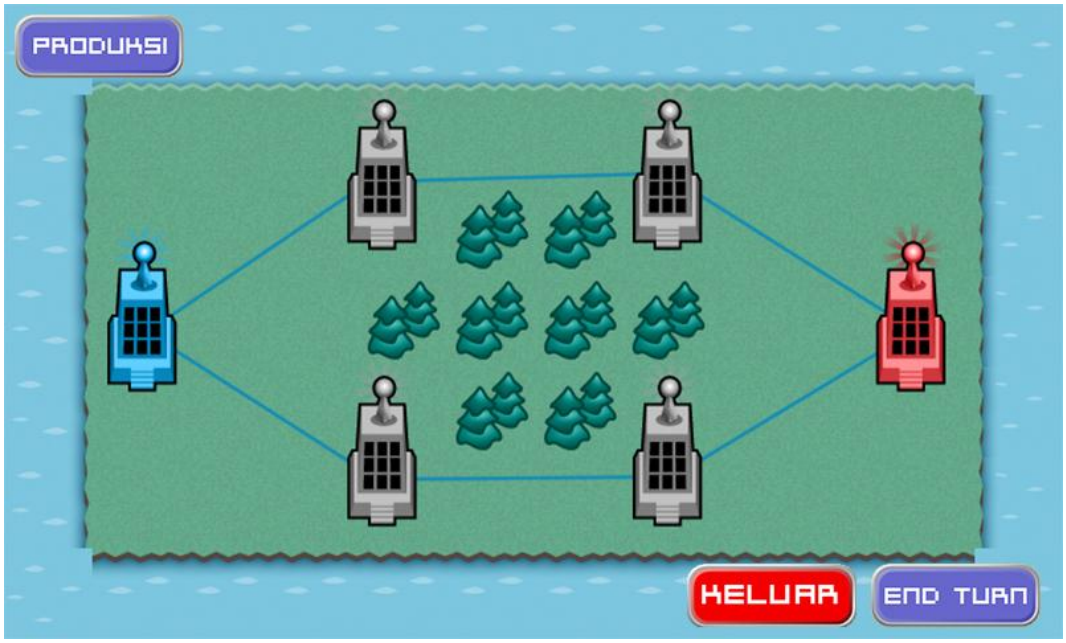
Gambar IV- 3 Antarmuka Petunjuk Main

4. Antarmuka Stage

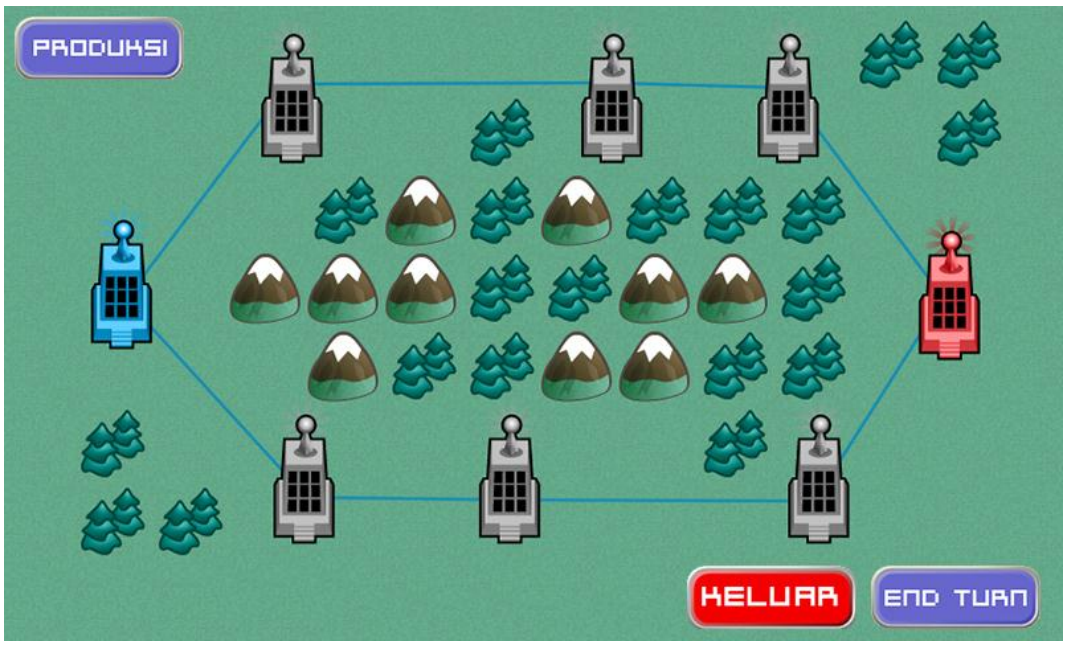
Antarmuka stage terdiri dari antarmuka stage 1 yang ditunjukan pada Gambar IV-4, antarmuka stage 2 yang ditunjukan pada Gambar IV-5 dan antarmuka stage 3 yang ditunjukan pada Gambar IV-6. Tombol yang tersedia adalah tombol produksi yang berfungsi untuk mengaktifkan fungsi produksi unit, tombol *end turn* yang berfungsi untuk mengakhiri *turn* (giliran) dan tombol keluar yang berfungsi untuk kembali ke menu main. Pada antarmuka ini juga terdapat *node* yang jumlahnya berbeda di setiap stage. Jika *node* milik AI (berwarna merah) atau *node* netral (berwarna abu-abu) dipilih maka akan menjalankan fungsi menyerang. Jika *node* milik *player* (berwarna biru) dipilih maka akan menjalankan fungsi bertahan.



Gambar IV- 4 Antarmuka Stage 1



Gambar IV- 5 Antarmuka Stage 2

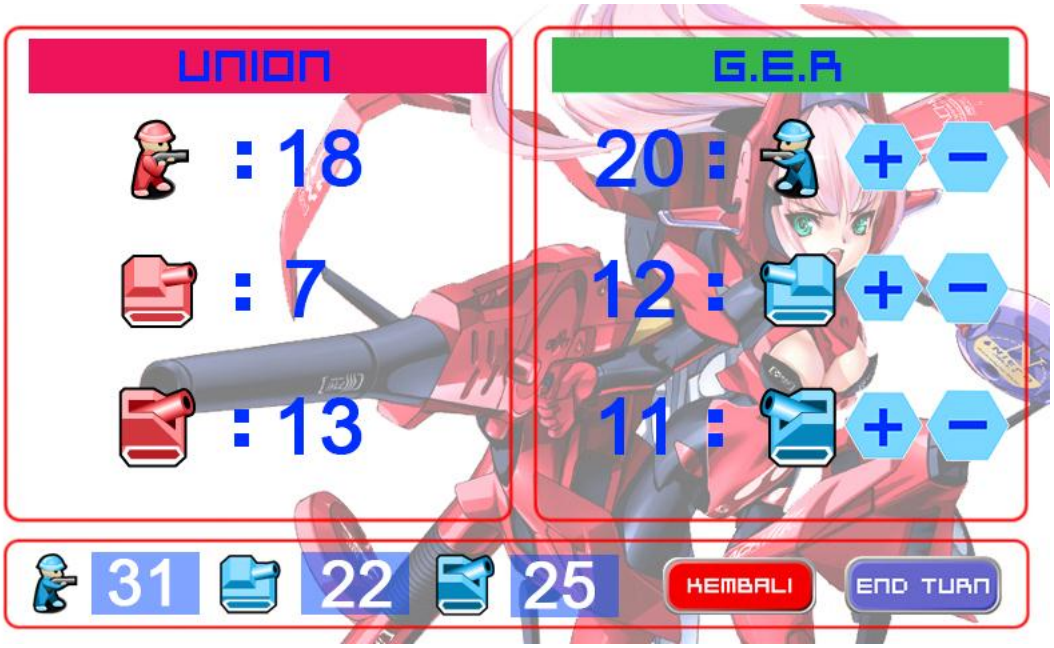


Gambar IV- 6 Antarmuka Stage 3

5. Antarmuka Menyerang *Node*

   Antarmuka menyerang *node* ditunjukan pada Gambar IV-7. Antarmuka menyerang node memiliki fungsi untuk menyerang *node* netral dan *node* musuh. Tombol yang tersedia adalah tombol tambah (+) yang berfungsi untuk

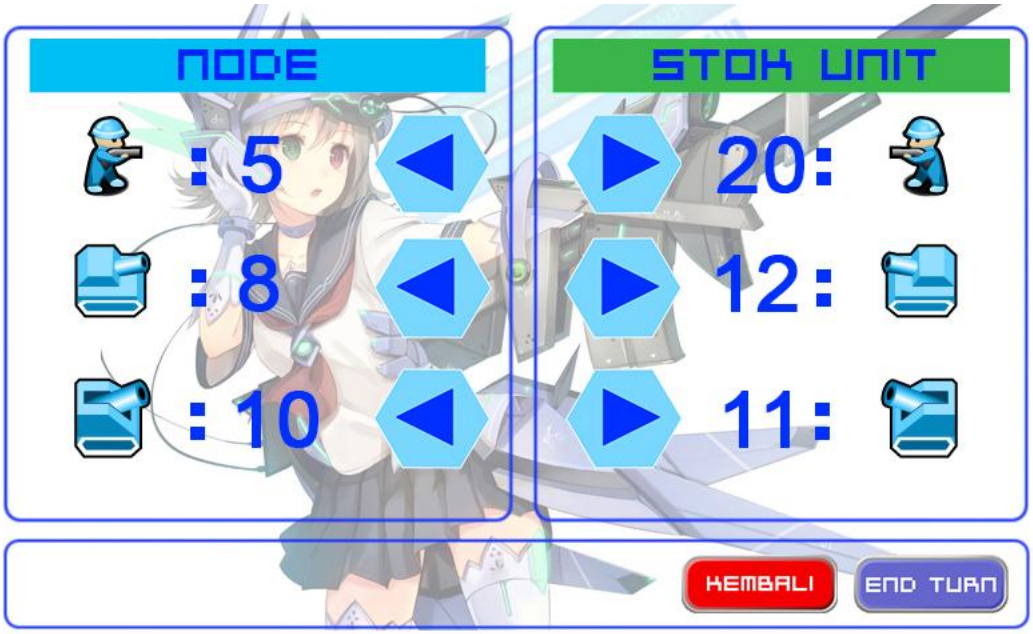
menambah jumlah unit yang akan menyerang, tombol kurang (-) yang berfungsi untuk mengurangi jumlah unit yang akan menyerang, tombol *end turn* yang berfungsi untuk mengakhiri *turn* (giliran) dan tombol kembali untuk membatalkan serangan sekaligus kembali ke antarmuka stage.



Gambar IV- 7 Antarmuka Menyerang *Node*

6. Antarmuka Mempertahankan *Node*

   Antarmuka mempertahankan *node* ditunjukan pada Gambar IV-8. Antarmuka mempertahankan node memiliki fungsi untuk memperkuat pertahanan *node* milik *player*. Tombol yang tersedia adalah tombol ◄ yang berfungsi untuk mengurangi jumlah unit yang akan mempertahankan *node*, tombol ► yang berfungsi untuk mengurangi jumlah unit yang akan mempertahankan *node*, tombol *end turn* yang berfungsi untuk mengakhiri *turn* (giliran) dan tombol kembali untuk membatalkan aksi memperkuat pertahanan sekaligus kembali ke antarmuka stage.



Gambar IV- 8 Antarmuka Mempertahankan *Node*

7. Antarmuka Memproduksi Unit

   Antarmuka memproduksi unit ditunjukan pada Gambar IV-9. Antarmuka memproduksi unit memiliki fungsi memproduksi unit yang akan dipakai dalam permainan. Tombol yang tersedia adalah tombol tambah (+) yang berfungsi untuk memproduksi unit, tombol kurang (-) yang berfungsi untuk menjual unit, tombol *end turn* yang berfungsi untuk mengakhiri *turn* (giliran) dan tombol kembali untuk membatalkan produksi sekaligus kembali ke antarmuka stage.



Gambar IV- 9 Antarmuka Memproduksi Unit

IV.1.2. Implementasi Pesan

Implementasi pesan adalah bagian yang menunjukan bentuk tampilan setiap pesan pada aplikasi yang sudah dibangun. Implementasi antar muka pada game ini ditunjukan pada Gambar IV-10 sampai dengan Gambar IV-13.

1. Pesan Keluar Permainan

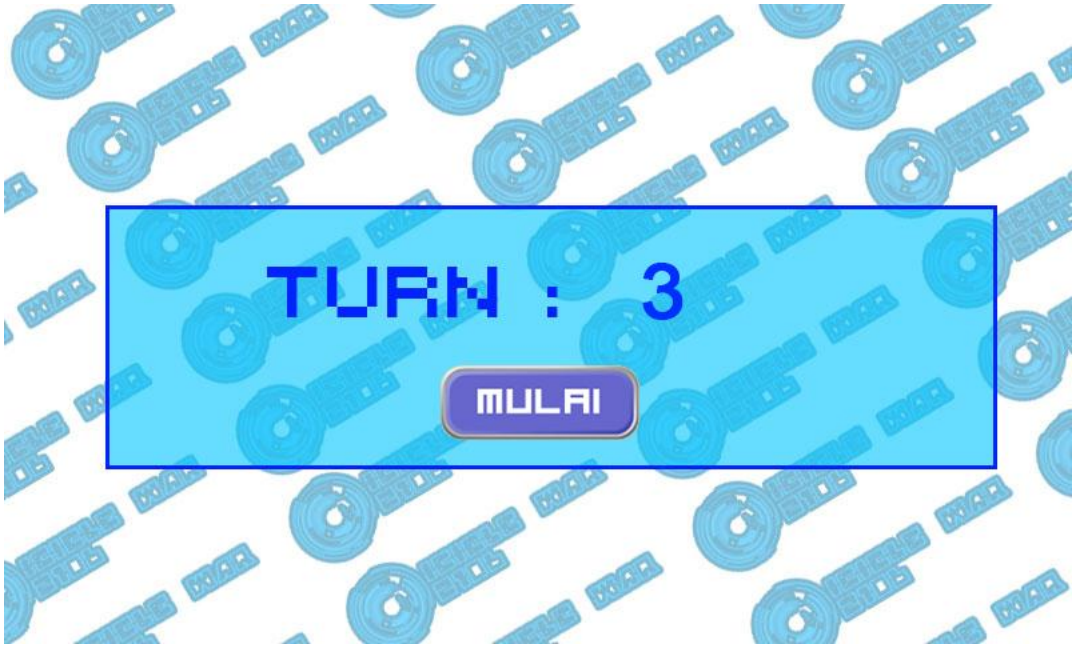
   Pesan keluar permainan ditunjukan pada Gambar IV-10. Pesan keluar permainan berfungsi sebagai validasi jika player ingin keluar dari permainan dan kembali ke menu play.



Gambar IV- 10 Pesan Keluar Permainan

2. Pesan *End Turn*

   Pesan *end turn* ditunjukan pada Gambar IV-11. Pesan *end turn* berfungsi sebagai pemberi informasi nilai turn selanjutnya.



Gambar IV- 11 Pesan *End Turn*

3. Pesan Menang

   Pesan menang ditunjukan pada Gambar IV-12. Pesan menang berfungsi sebagai pemberi informasi bahwa *player* telah berhasil mengalahkan AI dalam permainan dan pemberi informasi skor yang diperoleh oleh *player* yang berupa sisa unit milik *player* dan sisa *resource point* milik *player*.



Gambar IV- 12 Pesan Menang

4. Pesan Kalah

   Pesan kalah ditunjukan pada Gambar IV-13. Pesan kalah berfungsi sebagai pemberi informasi bahwa *player* telah dikalahkan AI dalam permainan dan pemberi informasi skor yang diperoleh oleh *player* yang berupa sisa unit milik *player* dan sisa *resource point* milik *player*.



Gambar IV- 13 Pesan Kalah

IV.2 Pengujian Sistem

Pengujian sistem adalah bagian penelusuran dalam rangka menemukan kesalahan pada sistem yang dibangun. Tujuan pengujian sistem adalah untuk mencari kesalahan pada sistem yang dibangun sebelum sistem digunakan oleh pengguna.

IV.2.1. Rencana Pengujian

Rencana pengujian perangkat lunak ini dibagi menjadi 2 bagian yaitu pengujian *black box* dan pengujian *white box.*

Pengujian *white box* dilakukan untuk menguji kesalahan logik dan asumsi yang tidak tepat pada kemungkinan eksekusi. Pengembangan kode-kode program selalu memungkinkan terjadinya alur program yang tidak dan tereksekusi dan kesalahan *typography* yang sulit ditemukan kalau tidak dijalankan. Pada umumnya pengujian ini dilakukan hanya pada lingkungan developer.

Pengujian *black box* dilakukan untuk mencari fungsi-fungsi program yang tidak benar atau hilang. Kesalahan lain yang mungkin terjadi dalam pembuatan program adalah kesalahan dalam struktur data, kesalahan antar muka, inisialisasi dan akhir program dan kesalahan performansi.

Strategi pengujian akan dilakukan dengan menggunakan pengujian *Beta*. Pengujian *beta* adalah pengujian pada lingkungan pengguna. Pengujian ini dilakukan dengan menggunakan metode pengumpulan data melalui kuesioner. Pengguna akan mencoba game ini, kemudian pengguna akan memberikan *feedback* pada kuesioner yang telah disediakan . Deskripsi rencana pengujian dapat dilihat dalam Tabel IV- 1.

Tabel IV- 1 Rencana Pengujian

No.	Kelas Uji	Detail Pengujian	Jenis Pengujian
1.	Uji algoritma *resource assignment*	Pengujian pada proses penentuan nilai keputusan pada algoritma *resource assignment*	*White Box*
2.	Uji antarmuka Menu Utama	Menekan tombol main	*Black Box*
		Menekan tombol petunjuk main	*Black Box*
3.	Uji antarmuka Menu Main	Menekan tombol 1	*Black Box*
		Menekan tombol 2	*Black Box*
		Menekan tombol 3	*Black Box*
		Menekan tombol kembali	*Black Box*
4.	Uji antarmuka Menu Petunjuk Main	Menekan tombol ►	*Black Box*
		Menekan tombol ◄	*Black Box*
		Menekan tombol kembali	*Black Box*
5.	Uji antarmuka Stage	Menekan tombol produksi	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol Keluar	*Black Box*
		Memilih node netral	*Black Box*
		Memilih node musuh	*Black Box*
		Memilih node kawan	*Black Box*
6.	Uji antarmuka Menyerang Node	Menekan tombol tambah (+)	*Black Box*
		Menekan tombol kurang (-)	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol kembali	*Black Box*

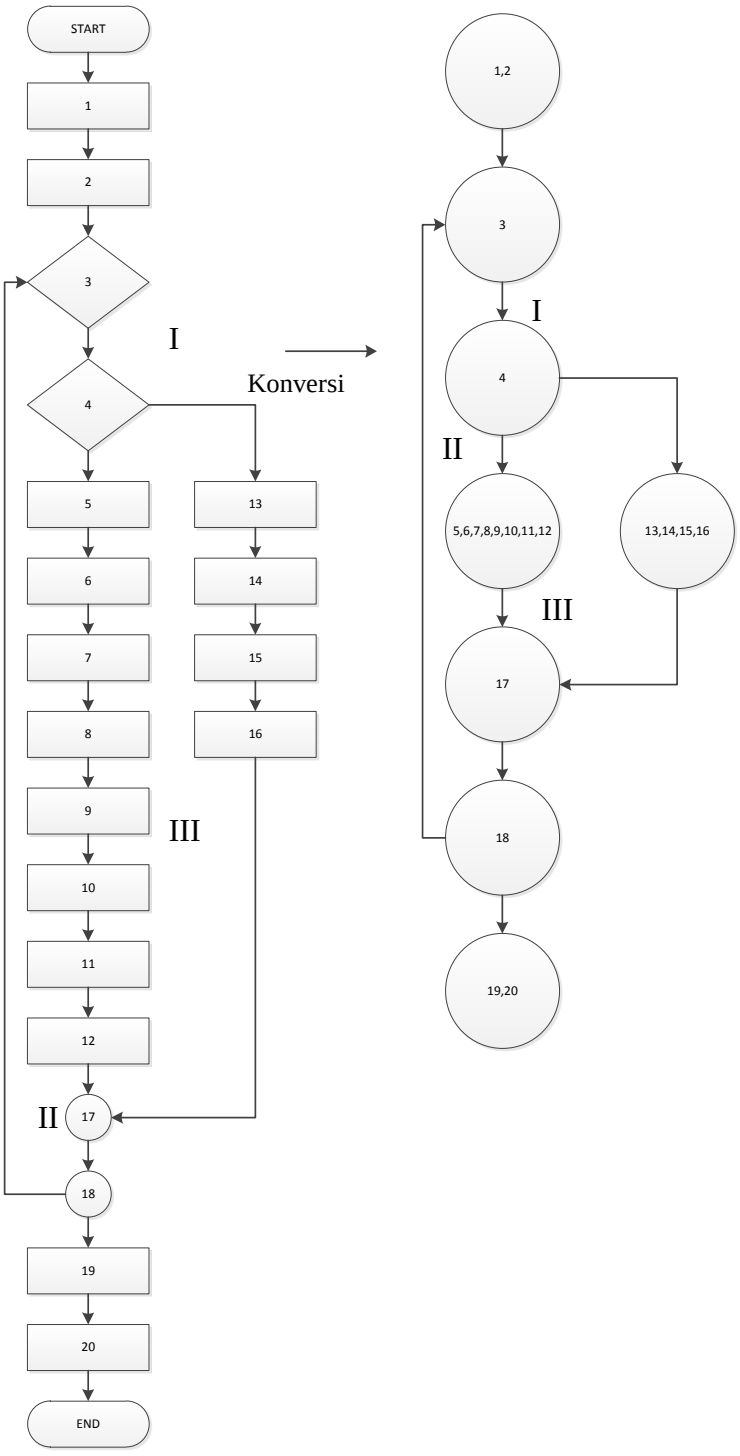
7.	Uji antarmuka Mempertahankan Node	Menekan tombol ►	*Black Box*
		Menekan tombol ◄	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol kembali	*Black Box*
8.	Uji antarmuka Memproduksi Unit	Menekan tombol tambah (+)	*Black Box*
		Menekan tombol kurang (-)	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol kembali	*Black Box*

IV.2.2. Pengujian *White Box*

Pengujian *white box* bertujuan untuk mengetahui kinerja logika yang dibuat pada sebuah perangkat lunak apakah berjalan dengan baik atau tidak. Pengujian *white box* akan digunakan pada algoritma *resource assignment.* Pengujian *white box* dilakukan dengan beberapa tahap, diantaranya :

a. Pengubahan pseucode menjadi *flowchart* kemudian menjadi *flowgraph.*

```
1   i ←0
2   M ←2
3   While Node[i,0] != null
4      if Node[i,1] = 2 then
5         countAttack(i,M)
6         countDefend(i,M)
7         countBuild(i,M)
8         countSkip(i,M)
9         sortModifier(i)
10        NodeCount[i,0] ←n
11        NodeCount[i,1] ← CountMod[3,1]
12        NodeCount[i,2] ← CountMod[3,0]
      else
14        NodeCount[i,0] ← n
15        NodeCount[i,1] ← -1
16        NodeCount[i,2] ← 0
17     endif
18  endwhile
19  sortNodeCount(n)
20  dec_val ←NodeCount[n-1,1]
```



Gambar IV- 14 Pengubahan *Flowchart* menjadi *Flowgraph*

b. *Cyclometic complexity*

V(G) = E – N + 2

= 8 – 7 + 2

= 3

Dimana :

E : Jumlah busur pada *flow graph*

N : Jumlah simpul pada *flow graph*

c. Jumlah *region*

Jumlah Region = Jumlah Kurva Tertutup + 1 (Region Terluar)

Jumlah Region = 2 + 1

Jumlah Region = 3

d. *Independent path*

(i) Path 1 : 1-2-3-4-5-6-7-8-9-10-11-12-17-18-19-20

(ii) Path 2 : 1-2-3-4-13-14-15-16-17-18-19-20

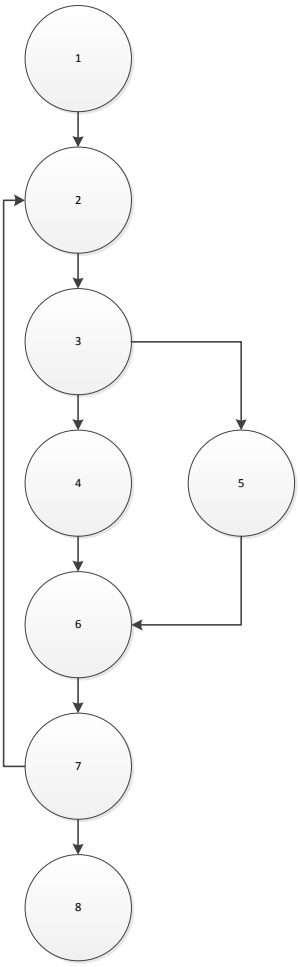
e. Jumlah *predicated node*

Predicated node = 3,4

V(G) = P + 1

= 2 + 1 = 3

f. Penomoran ulang *flowgraph*



Gambar IV- 15 Penomoran Ulang *Flowgraph*

g. *Graph matrix*

Simpul	1	2	3	4	5	6	7	8	Jumlah
1		1							0
2			1						0
3				1	1				1
4						1			0
5						1			0
6							1		0
7		1						1	1
8									0
Jumlah (x)									2

V(g) = x + 1

V(g) = 2 + 1

V(g) = 3

h. Kesimpulan

Kesimpulan pengujian white box dilakukan setelah dilakukannya langkah uji kasus yang terakhir yaitu memeriksa kondisi alur algoritma. Pengujian kondisi alur algoritma akan diperlihatkan pada Tabel IV- 2.

Tabel IV- 2 Pengujian Kondisi Alur Algoritma

No	Kasus Uji	Hasil yang Diharapkan	Hasil Sesuai Uji Kasus	Keterangan

1	Memasukkan nilai n=6 dengan asumsi seluruh nilai Node[i,1] adalah 0 ke dalam kondisi :
	
	For i←0 To n-1 do
	If Node[i,1] = 2
	Then
	……
	Else
	NodeCount[i,0] ← n
	NodeCount[i,1] ← -1
	NodeCount[i,2] ← 0
	Endif
		Menghasilkan 6 nilai array untuk NodeCount[0,1] s.d NodeCount[5,1]	Nilai array untuk NodeCount[0,1] s.d NodeCount[5,1]yang dihasilkan adalah -1.	[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

2	Memasukkan nilai i=4 dan nilai Node[4,1] adalah 2 ke dalam kondisi :
	
	If Node[i,1] = 2
	Then
	countAttack(i,M)
	countDefend(i,M)
	countBuild(i,M)
	Else
	NodeCount[i,0] ← n
	NodeCount[i,1] ← -1
	NodeCount[i,2] ← 0
	Endif
		Menghasilkan nilai array untuk NodeCount[4,1] Yaitu antara 0 dan 1	Nilai yang dihasilkan adalah NodeCount[4,1] = 0,33	[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

3	Memasukkan nilai i=4 dan nilai Node[4,1] adalah 0 ke dalam kondisi :
	
	If Node[i,1] = 2
	Then
	countAttack(i,M)
	countDefend(i,M)
	countBuild(i,M)
	Else
	NodeCount[i,0] ← n
	NodeCount[i,1] ← -1
	NodeCount[i,2] ← 0
	Endif
		Menghasilkan nilai array untuk NodeCount[4,1] Yaitu -1	Nilai yang dihasilkan adalah NodeCount[4,1] = -1	[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

Berdasarkan pengujian pada setiap metode, dihasilkan nilai *Cyclometic Complexity* yang sama yaitu 3. Maka dapat disimpulkan bahwa pengujian *white box* pada algoritma *resource assignment* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

IV.2.3. Pengujian *Black Box*

Pengujian *black box* adalah metode pengujian yang berfokus pada persyaratan perangkat lunak. Tujuan pengujian *black box* adalah menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang, menguji hasil keluaran tanpa melihat bagaimana proses untuk mendapatkan keluaran tersebut dan mengetahui kemampuan perangkat lunak dalam memenuhi kebutuhan pengguna. Kasus dan hasil pengujian dapat dilihat pada Tabel IV-2.

Tabel IV- 3 Kasus dan Hasil Pengujian Black Box

No.	Kelas Uji	Detail Pengujian	Hasil Yang Diharapkan	Hasil Pengujian
1.	Menu Utama	Menekan tombol main	Menampilkan menu main	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol petunjuk main	Menampilkan menu petunjuk main	[√] Berhasil [ ] Tidak Berhasil

2.	Menu Main	Menekan tombol 1	Menampilkan stage 1	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol 2	Menampilkan stage 2	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol 3	Menampilkan stage 3	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kembali	Menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil
3.	Menu Petunjuk Main	Menekan tombol ►	Menampilkan petunjuk berikutnya	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol ◄	Menampilkan petunjuk sebelumnya	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kembali	Menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil
4.	Stage	Menekan tombol produksi	Menampilkan antarmuka memproduksi unit	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol end turn	Menampilkan pesan end turn	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol Keluar	Menampilkan pesan keluar permainan	[√] Berhasil [ ] Tidak Berhasil
		Memilih node netral	Menampilkan antarmuka menyerang node	[√] Berhasil [ ] Tidak Berhasil
		Memilih node musuh	Menampilkan antarmuka menyerang node	[√] Berhasil [ ] Tidak Berhasil
		Memilih node kawan	Menampilkan antarmuka	[√] Berhasil [ ] Tidak Berhasil

			mempertahankan node	
5.	Menyerang Node	Menekan tombol tambah (+)	Menambah jumlah unit untuk menyerang	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kurang (-)	Mengurangi jumlah unit untuk menyerang	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol end turn	Menampilkan pesan *end turn*	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kembali	Menampilkan antarmuka *stage*	[√] Berhasil [ ] Tidak Berhasil
6.	Mempertahankan Node	Menekan tombol ►	Menambah jumlah unit pada *node*	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol ◄	Mengurangi jumlah unit pada *node*	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol end turn	Menampilkan pesan *end turn*	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kembali	Menampilkan antarmuka *stage*	[√] Berhasil [ ] Tidak Berhasil
7.	Memproduksi Unit	Menekan tombol tambah (+)	Memproduksi unit	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kurang (-)	Menjual unit	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol end turn	Menampilkan pesan *end turn*	[√] Berhasil [ ] Tidak Berhasil
		Menekan tombol kembali	Menampilkan antarmuka stage	[√] Berhasil [ ] Tidak Berhasil

IV.2.4. Pengujian *Beta*

Pengujian *beta* adalah pengujian pada lingkungan pengguna. Pengujian ini menggunakan metode pengumpulan data melalui kuisioner. Pengguna akan mencoba game ini, kemudian pengguna akan memberikan *feedback* pada kuesioner yang telah disediakan.

Untuk mengetahui tanggapan dan penilaian pengguna terhadap game ini. Telah disebarkan kuesioner kepada 30 responden dengan usia berdasarkan pada kebutuhan pengguna. Berikut adalah pertanyaan dan hasil kuesioner yang telah dibagikan dengan menggunakan rumus :

Y = P/Q * 100%

Keterangan :

P = Banyaknya jawaban responden tiap soal

Q = Jumlah responden

Y = Nilai prosentase jawaban

Hasil Pengujian Beta adalah sebagai berikut:

1. Ketika anda memainkan *game* ini, apakah langkah-langkah yang diambil oleh AI sulit diprediksi oleh anda?

	Kategori Jawaban		
	Ya	Biasa	Tidak
Frekuensi Jawaban	14	13	3
Prosentase Nilai	46%	43%	11%

2. Apakah menurut anda AI pada *game* ini sulit dikalahkan?

	Kategori Jawaban		
	Ya	Biasa	Tidak
Frekuensi Jawaban	10	12	8
Prosentase Nilai	33%	40%	27%

3. Apakah menurut anda cara bermain pada *game* ini mudah dimengerti?

	Kategori Jawaban		
	Ya	Biasa	Tidak
Frekuensi Jawaban	10	12	8
Prosentase Nilai	33%	40%	27%

4. Apakah kontrol pada *game* mudah dioperasikan?

	Kategori Jawaban		
	Ya	Biasa	Tidak
Frekuensi Jawaban	15	8	7
Prosentase Nilai	50%	27%	23%

5. Apakah informasi pada antarmuka *game* sudah cukup jelas?

	Kategori Jawaban		
	Ya	Biasa	Tidak
Frekuensi Jawaban	15	10	5
Prosentase Nilai	50%	33%	17%

Kesimpulan dari pengujian beta ini adalah :

1. AI menjadi lebih sulit diprediksi langkahnya serta menjadi semakin sulit dikalahkan bagi sebagian *player* ketika algoritma *resource assignment* diaplikasikan pada AI.
2. Aturan-aturan yang ada pada game ini cukup mudah dimengerti oleh pengguna (*player*).
3. Informasi serta kontrol pada game ini ditampilkan dengan cukup baik sehingga membuktikan bahwa *game turn-based strategy* cocok diaplikasikan pada perangkat *mobile*.