BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan *Game*, tahap ini merupakan tahap dimana *Game* siap untuk dioperasikan. *Game* ini diimplementasikan dengan nama *the kingdom of* pandawa dan *file extension*-nya *.jar* (*application package file*).

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang dapat digunakan untuk menjalankan *game* ini tidak harus memiliki spesifikasi yang tinggi, seperti *processor Core* I5 dan *VGA Card* 1GB. Sebagai ilustrasi sistem minimum agar dapat menjalankan *game* lutung kasarung *the jumper* adalah sebagai berikut :

1. *Processor dual core*.
2. *RAM* dengan kapasitas minimum 256*MB*.
3. *VGA Card* dengan kapasitas minimum 256 *MB*.

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan *Game the kingdom of* pandawa adalah *Java Virtual Machine* (JVM).

4.1.3 Implementasi Instalasi *Game*

Sebelum dapat memainkan *game the kingdom of* pandawa, *Game* harus tersimpan di direktori komputer. Petunjuk dalam instalasi *game* ini adalah sebagai berikut :

1. *Copy file the kingdom of* pandawa.*jar* pada direktori komputer.
2. Buka dan klik pada *file* tersebut.
3. Setelah itu, *game* tersebut akan menampilkan permainan.

4.1.4 Implementasi Antarmuka

Adapun antarmuka yang ada pada *game the kingdom of* pandawa :

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 :

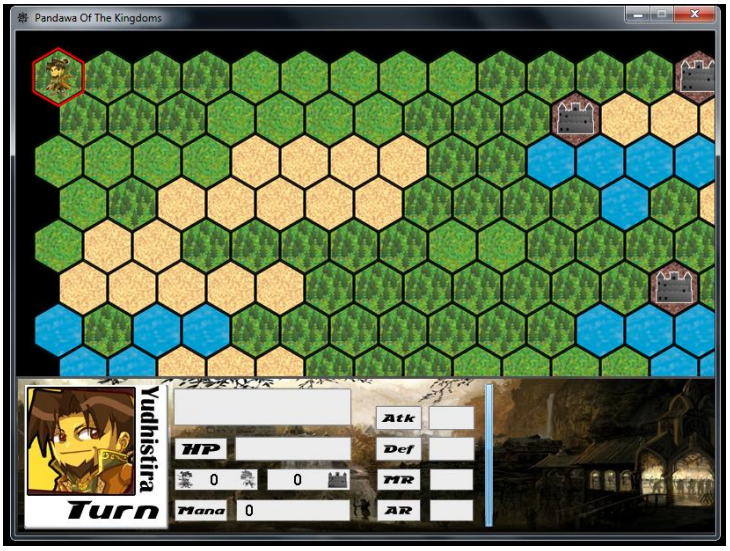


Gambar 4.1 Antarmuka Menu Utama

   Gambar 4.1 merupakan menu utama pada *game the kingdom of* pandawa. Terdapat 2 sub menu di dalamnya, yaitu menu *story game* yang berfungsi untuk memulai permainan, menu *how to play* yang berfungsi untuk melihat petunjuk bermain.

2. Antarmuka Menu *Story Game*

   Saat pengguna memilih menu *Story Game*, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka yang didalamnya terdapat menu *level*. Berikut ini merupakan tampilan dari antarmuka menu *Story Game* :

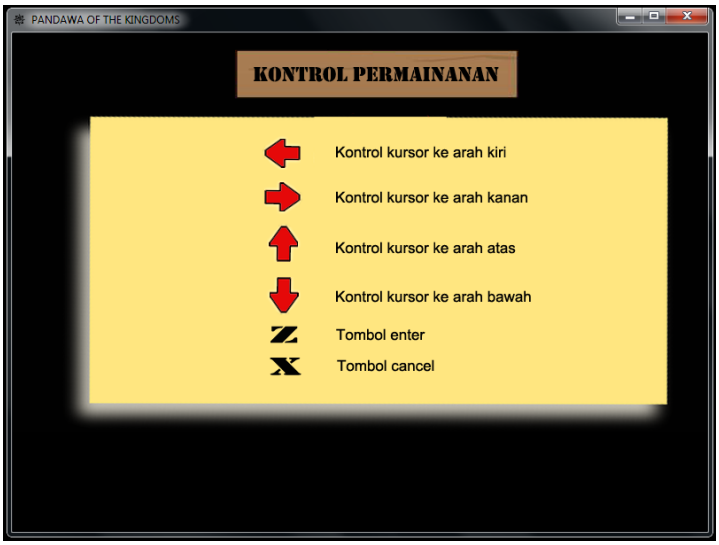


Gambar 4.2 Antarmuka Menu *Story Game*

Gambar 4.2 merupakan *Story Game* yang terdapat pada *game the kingdom of* pandawa. Terdapat 3 (tiga) level di dalamnya, yaitu *level* 1, *level* 2, dan *level* 3.

3. Antarmuka Menu *How to play*

Saat pengguna memilih menu *how to play*, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka *how to play*. Berikut ini merupakan tampilan dari antarmuka menu *how to play* :

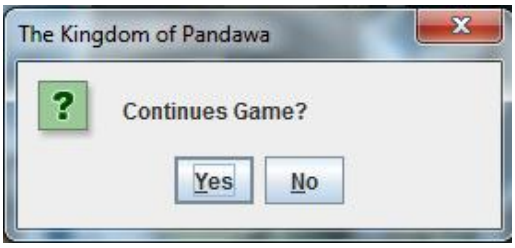


Gambar 4.3 Antarmuka Menu *How to play*

Gambar 4.3 merupakan menu *How to play* yang terdapat pada *game The Kingdom of* Pandawa. Terdapat petunjuk kontrol *game* di dalamnya.

4.1.5 Implementasi Pesan

Adapun pesan pada *game The Kingdom of* Pandawa adalah sebagai berikut:



Gambar 4.4 Pesan lanjut permainan

Gambar 4.4 merupakan pesan yang terdapat dalam menu *Story Game*. Pesan ini berfungsi untuk melanjutkan ke permainan selanjutnya.

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. Cara pengujian dilakukan melalui dua tahap, yaitu pengujian *alpha* (fungsional) dan pengujian *beta*.

4.2.1 Pengujian *Alpha*

Pengujian dilakukan terhadap *game* untuk memastikan bahwa *game* 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 *Game*

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada *game the kingdom of* pandawa. Skenario pengujian yang akan dilakukan pada ini selengkapnya dapat dilihat pada table 4.1 :

Tabel 4.1 Skenario Pengujian *Game*

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1.	Menu Utama	Klik menu *Story Game*	*Black box*
		Klik menu *How to play*	*Black box*
2.	Menu *Story Game*	Klik *Story Game*	*Black box*
		Permainan *level* 1	*White box*
		Klik *move*	*Black box*
		Klik *attack*	*Black box*
		Klik *summon*	*Balck box*
		Klik *change turn*	*Black box*
		Klik pesan	*Black box*
		Permainan *level* 2	*White box*
		Klik *move*	*Black box*
		Klik *attack*	*Black box*
		Klik *summon*	*Black box*
		Klik *change turn*	*Black box*
		Klik pesan	*Black box*
		Permainan *level* 3	*White box*
		Klik *move*	*Black box*
		Klik *attack*	*Black box*
		Klik *summon*	*Black box*
		Klik *change turn*	*Black box*
		Klik pesan	*Black box*
3.	Menu *How to play*	Klik Menu *How to play*	*Black box*
		Klik tombol kembali	*Black box*

4.	Algoritma A* (A-Star)	Algoritma A* pada level 1	*White box*
		Algoritma A* pada level 2	*White box*
		Algoritma A* pada level 3	*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	Klik menu *Story Game*	Sistem menampilkan halaman menu *Level*	[√] Berhasil [ ] Tidak Berhasil
		Klik menu *How to play*	Sistem menampilkan halaman menu *how to play*	[√] Berhasil [ ] Tidak Berhasil
2.	Menu *Story Game*	Klik menu *Story Game*	Sistem menampilkan halaman menu *Level*	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan dunia *level* 1	[√] Berhasil [ ] Tidak Berhasil
		Permainan *Level* 1	Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)	
		Klik *move*	Sistem	[√] Berhasil

			menampilkan pergerakan karakter	[ ] Tidak Berhasil
		Klik *attack*	Sistem menampilkan serangan karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik *summon*	Sistem memunculkan karakter lain di arena permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik *change turn*	Sistem memindahkan giliran main karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan dunia *level* 2	[√] Berhasil [ ] Tidak Berhasil
		Permainan *Level* 2	Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)	
		Klik *move*	Sistem menampilkan pergerakan karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik *attack*	Sistem menampilkan serangan karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik *summon*	Sistem memunculkan karakter lain di arena permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik *change turn*	Sistem memindahkan giliran main karakter	[√] Berhasil [ ] Tidak Berhasil

		Klik pesan	Sistem menampilkan dunia *level* 3	[√] Berhasil [ ] Tidak Berhasil
		Permainan *Level* 3	Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)	
		Klik *move*	Sistem menampilkan pergerakan karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik *attack*	Sistem menampilkan serangan karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik *summon*	Sistem memunculkan karakter lain di arena permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik *change turn*	Sistem memindahkan giliran main karakter	[√] Berhasil [ ] Tidak Berhasil
		Klik pesan	Sistem menampilkan permainan telah berakhir	[√] Berhasil [ ] Tidak Berhasil
3.	Menu *How to play*	Klik Menu *How to play*	Sistem menampilkan informasi petunjuk permainan	[√] Berhasil [ ] Tidak Berhasil
		Klik kembali	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil

4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)

Berikut ini merupakan kasus menu *Story Game* untuk menguji permainan *level* 1, permainan *level* 2, dan permainan *level* 3 di dalam perangkat lunak yang sudah dibangun dengan menggunakan metode *white box testing* berdasarkan skenario pengujian *game* yang terdapat pada tabel 4.1 yang sudah dijabarkan.

1. Pengujian Permainan *level* 1 (*white box*)

   Pengujian permainan *level* 1 merupakan proses dimana akan diuji algoritma A* terhadap musuh dalam melakukan pencarian rute terpendek. Berikut ini merupakan pengujian algoritma A* yang di terapkan pada permainan *level* 1 dengan menggunakan metode *white box*.

   a. Algoritma Yang Akan Di Uji

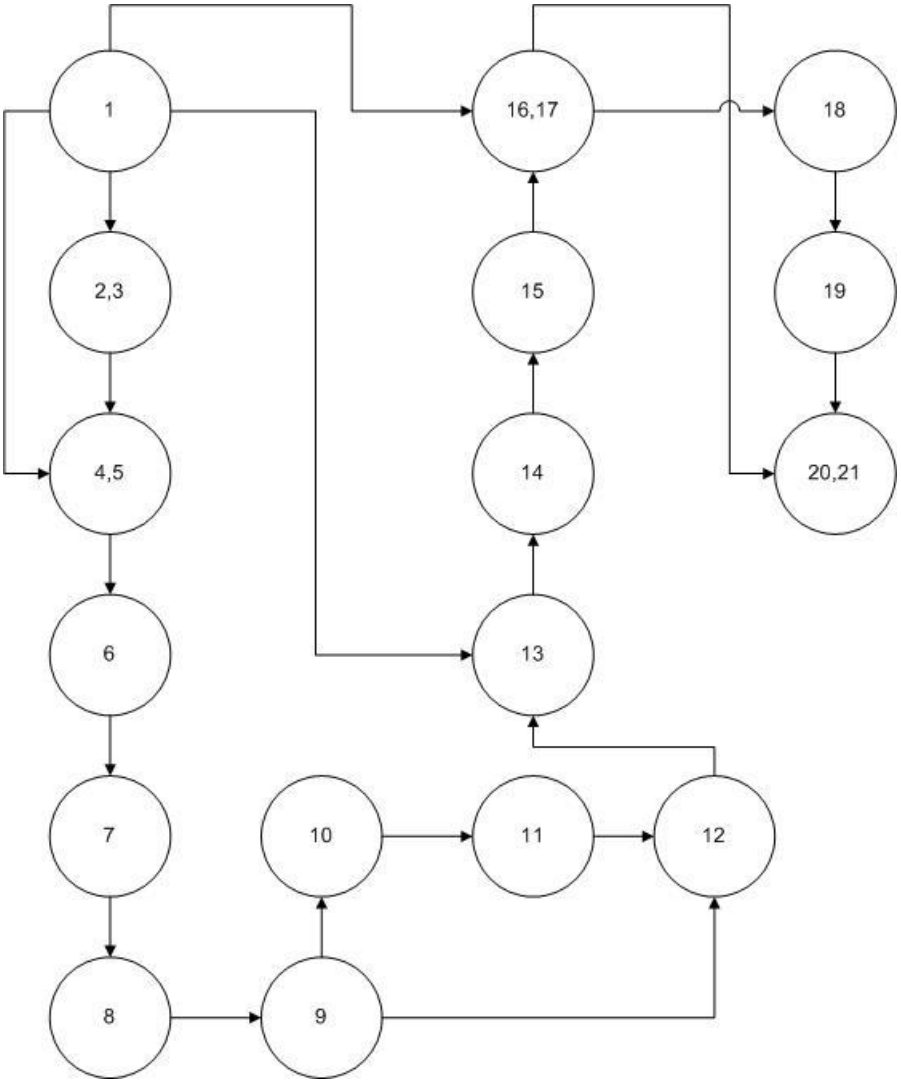
   Berikut ini merupakan algoritma A* untuk *level* 1 yang penerapannya menggunakan bahasa pemrograman *java* :

```
1.   while(!lstOpen.isEmpty)
     {
2.   if(curHexa.equals(dunia.getTarget())
     {
3.   break;
     }
4.   List<HexaMap> lstHexaSekitar ←
     dunia.getHexaSekitar(curHexa);
5.   for(HexaMap hexa : lstHexaSekitar)
     {
6.   if(lstClose.contains(hexa))
     {
     int i ← lstOpen.indexOf(hexa);
7.   if(i >= 0){
8.   HexaMap oldHexa ← lstOpen.get(i);
     HexaMap oldParent ← oldHexa.getParent();
     int oldCost ← hitungNilaiG(oldHexa);
     oldHexa.setParent(curHexa);
9.   if(hitungNilaiG(oldHexa) > oldCost)
10.  oldHexa.setParent(oldParent);
     }
```

```
11.  else
     {
12.  hexa.setParent(curHexa);
     lstOpen.add(hexa);
     }
     lstOpen.remove(curHexa);
     lstClose.add(curHexa);
13.  if(!lstOpen.isEmpty())
     {
14.  curHexa ← (Collections.min(lstOpen,
     Comparator<HexaMap>(){
15.  public int compare(HexaMap hexa1, HexaMap hexa2)
     {
16.  return Astar.hitungNilaiF(hexa1) -
     Astar.hitungNilaiF(hexa2);
     }
17.  }));
     }
     }
18.  lstHasil ← ArrayList<HexaMap>();
19.  while(curHexa.getParent()!=null)
     {
20.  lstHasil.add(curHexa.getParent));
     curHexa ← curHexa.getParent();
     }
21.  Collections.reverse(lstHasil);
     }
```

b. *Flow Graph* Permainan *Level* 1

Berikut ini merupakan *flow graph* permainan *level* 1 yang dapat dilihat pada gambar 4.5 :



Gambar 4.5 *Flow Graph* Permainan level 1

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 21, N = 18

V(G) = E – N + 2

V(G) = 21 – 18 + 2

V(G) = 3 + 2 = 5

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1),(2,3),(4,5),(6), (7)

Path 2 = (1),(4,5),(7),(8), (9),(10)

Path 3 = (1),(9),(10),(11),(12)

Path 4 = (1) ,(13),(14),(15),(16,17)

Path 5 = (1),(16,17),(18),(19),(20,21)

e. *Graph Matriks* Permainan *Level* 1

Table 4.3 *Graph Matriks* Permainan *Level* 1

	(1)	(2,3)	(4,5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16,17)	(18)	(19)	(20,21)
(1)		1					1				1				1		
(2,3)			1														
(4,5)				1		1											
(6)					1												
(7)						1											
(8)							1										
(9)								1									
(10)									1								
(11)										1							
(12)											1						
(13)												1					
(14)													1				
(15)														1			
(16,17)															1		
(18)																1	
(19)																	1
(20,21)																	

Keterangan :

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

f. Kesimpulan

Dari hasil pengujian permainan *level* 1 yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *path* telah dikerjakan, serta tidak terjadi pengulangan.

2. Pengujian Permainan *level* 2 (*white box*)

Pengujian permainan *level* 2 merupakan proses dimana akan diuji algoritma A* terhadap musuh dalam melakukan pencarian rute terpendek. Berikut ini merupakan pengujian algoritma A* yang di terapkan pada permainan *level* 2 dengan menggunakan metode *white box*.

a. Algoritma Yang Akan Di Uji

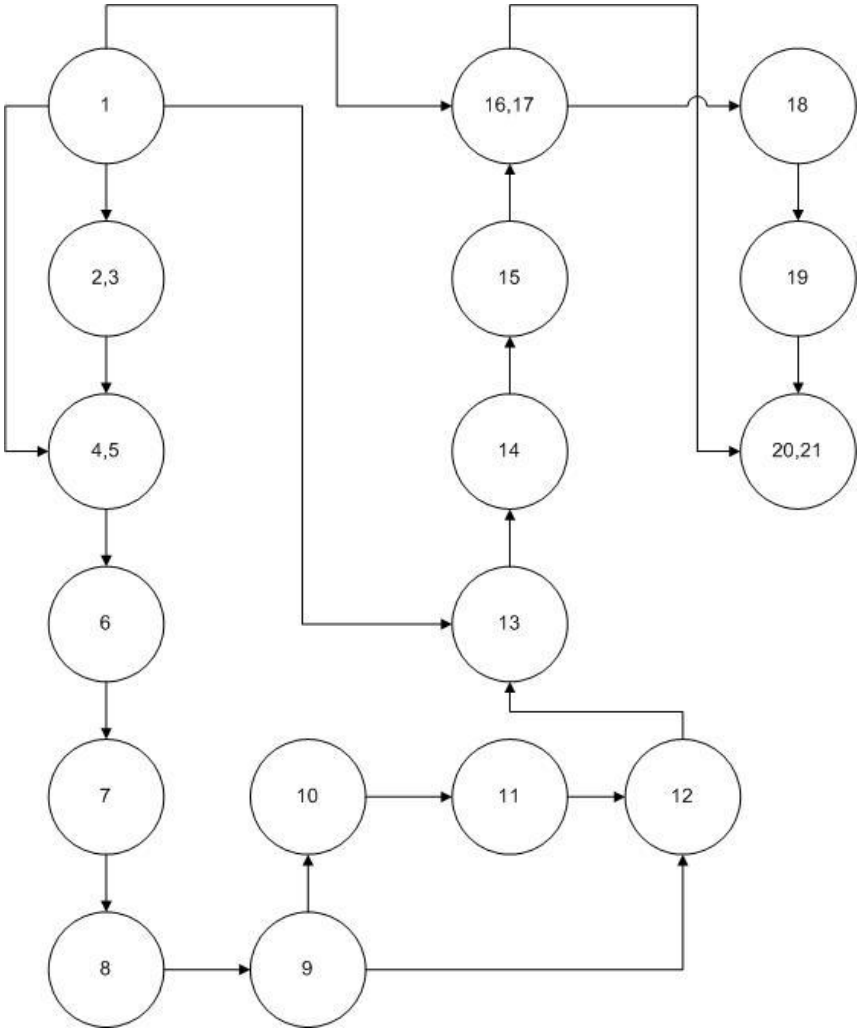
Berikut ini merupakan algoritma A* untuk *level* 2 yang penerapannya menggunakan bahasa pemrograman *java* :

```
1.   while(!lstOpen.isEmpty)
     {
2.   if(curHexa.equals(dunia.getTarget())
     {
3.   break;
     }
4.   List<HexaMap> lstHexaSekitar ←
     dunia.getHexaSekitar(curHexa);
5.   for(HexaMap hexa : lstHexaSekitar)
     {
6.   if(lstClose.contains(hexa))
     {
     int i ← lstOpen.indexOf(hexa);
7.   if(i >= 0){
8.   HexaMap oldHexa ← lstOpen.get(i);
     HexaMap oldParent ← oldHexa.getParent();
     int oldCost ← hitungNilaiG(oldHexa);
     oldHexa.setParent(curHexa);
9.   if(hitungNilaiG(oldHexa) > oldCost)
```

```
10.  oldHexa.setParent(oldParent);
     }
11.  else
     {
12.  hexa.setParent(curHexa);
     lstOpen.add(hexa);
     }
     lstOpen.remove(curHexa);
     lstClose.add(curHexa);
13.  if(!lstOpen.isEmpty())
     {
14.  curHexa ← (Collections.min(lstOpen,
     Comparator<HexaMap>(){
15.  public int compare(HexaMap hexa1, HexaMap hexa2)
     {
16.  return Astar.hitungNilaiF(hexa1) -
     Astar.hitungNilaiF(hexa2);
     }
17.  }));
     }
     }
18.  lstHasil ← ArrayList<HexaMap>();
19.  while(curHexa.getParent()!=null)
     {
20.  lstHasil.add(curHexa.getParent));
     curHexa ← curHexa.getParent();
     }
21.  Collections.reverse(lstHasil);
     }
```

b. *Flow Graph* Permainan *Level* 2

Berikut ini merupakan *flow graph* permainan *level* 2 yang dapat dilihat pada gambar 4.6 :



Gambar 4.6 *Flow Graph* Permainan level 2

Keterangan :

= Menggambarkan Kondisi

→ = Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 21, N = 18

V(G) = E – N + 2

V(G) = 21 – 18 + 2

V(G) = 3 + 2 = 5

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1),(2,3),(4,5),(6), (7)

Path 2 = (1),(4,5),(7),(8), (9),(10)

Path 3 = (1),(9),(10),(11),(12)

Path 4 = (1) ,(13),(14),(15),(16,17)

Path 5 = (1),(16,17),(18),(19),(20,21)

e. *Graph Matriks* Permainan *Level* 2

Table 4.4 *Graph Matriks* Permainan *Level* 2

	(1)	(2, 3)	(4, 5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16, 17)	(18)	(19)	(20, 21)
(1)		1					1				1				1		
(2,3)			1														
(4,5)				1		1											
(6)					1												
(7)						1											
(8)							1										
(9)								1									
(10)									1								
(11)										1							
(12)											1						
(13)												1					
(14)													1				
(15)														1			
(16,17)															1		
(18)																1	
(19)																	1
(20,21)																	

Keterangan :

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

f. Kesimpulan

Dari hasil pengujian permainan *level* 2 yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *path* telah dikerjakan, serta tidak terjadi pengulangan.

3. Pengujian Permainan *level* 3 (*white box*)

Pengujian permainan *level* 3 merupakan proses dimana akan diuji algoritma A* terhadap musuh dalam melakukan pencarian rute terpendek. Berikut ini merupakan pengujian algoritma A* yang di terapkan pada permainan *level* 3 dengan menggunakan metode *white box*.

a. Algoritma Yang Akan Di Uji

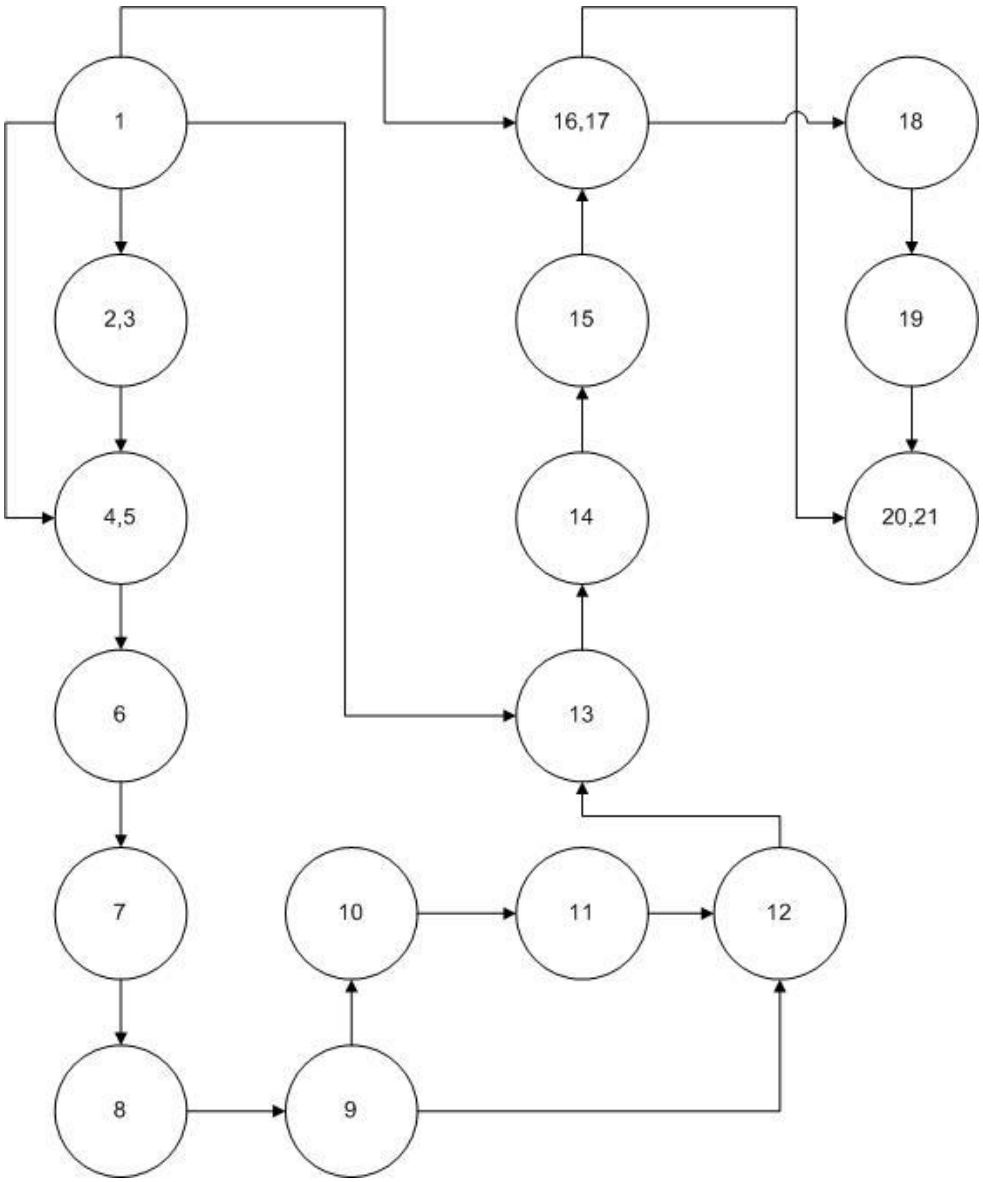
Berikut ini merupakan algoritma A* untuk *level* 3 yang penerapannya menggunakan bahasa pemrograman *java* :

```
1.   while(!lstOpen.isEmpty)
     {
2.   if(curHexa.equals(dunia.getTarget())
     {
3.   break;
     }
4.   List<HexaMap> lstHexaSekitar ←
     dunia.getHexaSekitar(curHexa);
5.   for(HexaMap hexa : lstHexaSekitar)
     {
6.   if(lstClose.contains(hexa))
     {
     int i ← lstOpen.indexOf(hexa);
7.   if(i >= 0){
8.   HexaMap oldHexa ← lstOpen.get(i);
     HexaMap oldParent ← oldHexa.getParent();
     int oldCost ← hitungNilaiG(oldHexa);
     oldHexa.setParent(curHexa);
9.   if(hitungNilaiG(oldHexa) > oldCost)
```

```
10.  oldHexa.setParent(oldParent);
     }
11.  else
     {
12.  hexa.setParent(curHexa);
     lstOpen.add(hexa);
     }
     lstOpen.remove(curHexa);
     lstClose.add(curHexa);
13.  if(!lstOpen.isEmpty())
     {
14.  curHexa ← (Collections.min(lstOpen,
     Comparator<HexaMap>(){
15.  public int compare(HexaMap hexa1, HexaMap hexa2)
     {
16.  return Astar.hitungNilaiF(hexa1) -
     Astar.hitungNilaiF(hexa2);
     }
17.  }));
     }
     }
18.  lstHasil ← ArrayList<HexaMap>();
19.  while(curHexa.getParent()!=null)
     {
20.  lstHasil.add(curHexa.getParent));
     curHexa ← curHexa.getParent();
     }
21.  Collections.reverse(lstHasil);
     }
```

b. *Flow Graph* Permainan *Level* 3

Berikut ini merupakan *flow graph* permainan *level* 3 yang dapat dilihat pada gambar 4.7 :



Gambar 4.7 *Flow Graph* Permainan level 3

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 21, N = 18

V(G) = E – N + 2

V(G) = 21 – 18 + 2

V(G) = 3 + 2 = 5

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. *Independent Path*

Path 1 = (1),(2,3),(4,5),(6), (7)

Path 2 = (1),(4,5),(7),(8), (9),(10)

Path 3 = (1),(9),(10),(11),(12)

Path 4 = (1) ,(13),(14),(15),(16,17)

Path 5 = (1),(16,17),(18),(19),(20,21)

e. *Graph Matriks* Permainan *Level* 3

Table 4.5 *Graph Matriks* Permainan *Level* 3

	(1)	(2,3)	(4,5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16,17)	(18)	(19)	(20,21)
(1)		1					1				1				1		
(2,3)			1														
(4,5)				1		1											
(6)					1												
(7)						1											
(8)							1										
(9)								1									
(10)									1								
(11)										1							
(12)											1						
(13)												1					
(14)													1				
(15)														1			
(16,17)															1		
(18)																1	
(19)																	1
(20,21)																	

Keterangan :

3. Baris dan kolom merepresentasikan simpul
4. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Kesimpulan

Dari hasil pengujian permainan *level* 3 yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *path* telah dikerjakan, serta tidak terjadi pengulangan.

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.

4.2.2.1 Kuesioner

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 9 tahun ke atas, dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian penerapan sistem yang telah dibangun. Kuesioner terdiri dari 6 pernyataan dengan menggunakan skala 1 sampai 5 (skala likert).

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

$$P = \frac{S}{skor\ ideal} \times 100$$

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.

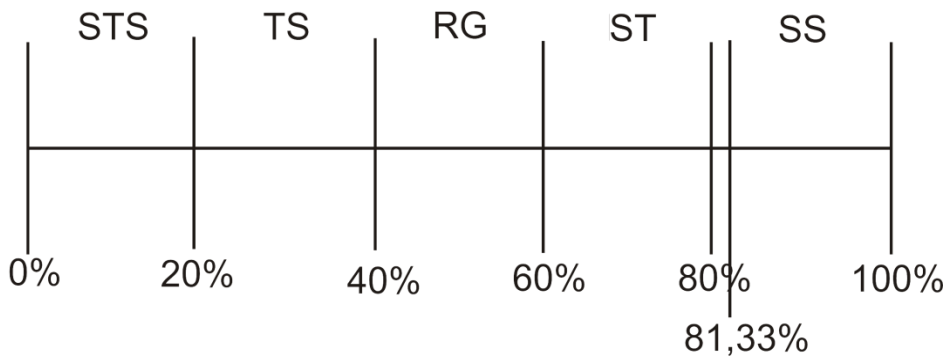
1. *Gameplay* pada *game* ini mudah.

Tabel 4.6 Perhitungan persentase pertanyaan ke-1

Pertanyaan	Keterangan	Skor	Frekuensi	S
1	Sangat Setuju	5	10	50
	Setuju	4	18	72
	Ragu-ragu	3	2	6
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**122**

$$P = \frac{122}{150} \times 100\% = 81.33\%$$

Dapat disimpulkan bahwa penilaian terhadap tampilan secara umur baik tidaknya pada permainan *The Kingdom of* Pandawa adalah 81,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.8.



Gambar 4.8 Interpretasi Skor Pertanyaan ke-1

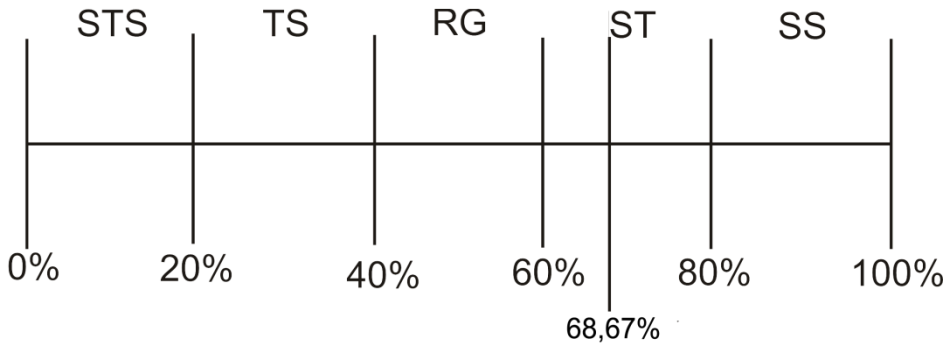
2. Sulit untuk menyelesaikan *game* ini.

Tabel 4.7 Perhitungan persentase pertanyaan ke-2

Pertanyaan	Keterangan	Skor	Frekuensi	S
2	Sangat Setuju	5	0	0
	Setuju	4	16	64
	Ragu-ragu	3	11	33
	Tidak Setuju	2	3	6
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**103**

$$P = \frac{103}{150} \times 100\% = 68.67\%$$

Dapat disimpulkan bahwa penilaian terhadap kecerdasan musuh pada permainan *The Kingdom of* Pandawa adalah 68,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.9.



Gambar 4.9 Interpretasi Skor Pertanyaan ke-2

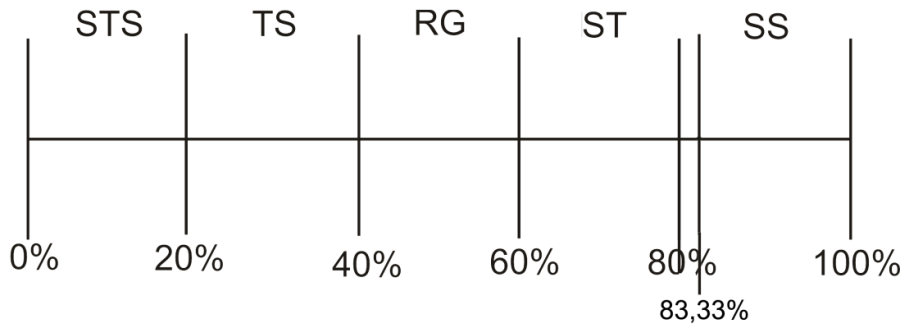
3. *Game* ini dapat membantu anda untuk mengingat kembali tokoh cerita pandawa dan kurawa.

Tabel 4.8 Perhitungan persentase pertanyaan ke-3

Pertanyaan	Keterangan	Skor	Frekuensi	S
3	Sangat Setuju	5	9	45
	Setuju	4	17	68
	Ragu-ragu	3	4	12
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**125**

$$P = \frac{125}{150} \times 100\% = 83.33\%$$

Dapat disimpulkan bahwa penilaian terhadap deteksi tubrukan pada permainan *The Kingdom of* Pandawa adalah 83,33% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.10.



Gambar 4.10 Interpretasi Skor Pertanyaan ke-3

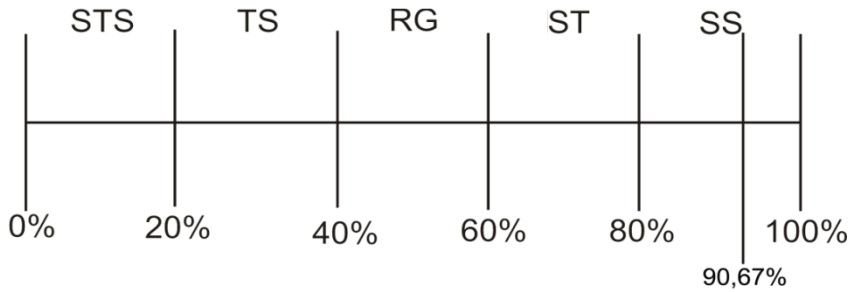
4. Skenario permainan yang di buat sudah menarik untuk di mainkan.

Tabel 4.9 Perhitungan persentase pertanyaan ke-4

Pertanyaan	Keterangan	Skor	Frekuensi	S
4	Sangat Setuju	5	19	95
	Setuju	4	9	36
	Ragu-ragu	3	1	3
	Tidak Setuju	2	1	2
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**136**

$$P = \frac{136}{150} \times 100\% = 90.67\%$$

Dapat disimpulkan bahwa penilaian terhadap audio pada permainan *The Kingdom of* Pandawa adalah 90,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.11.



Gambar 4.11 Interpretasi Skor Pertanyaan ke-4

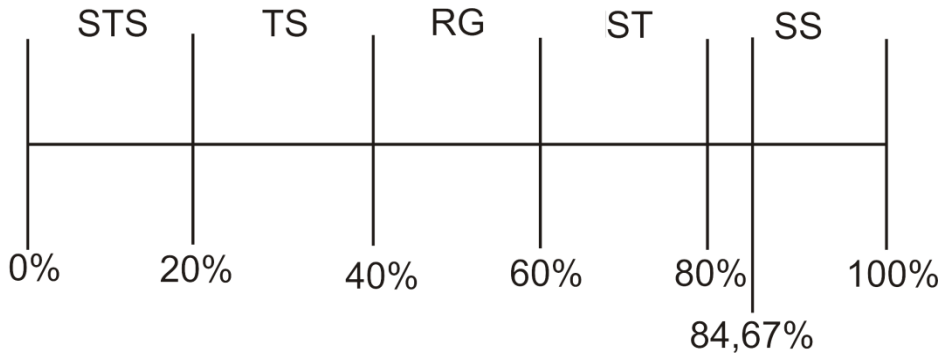
5. Tampilan grafik 2D pada *game* ini sudah bagus.

Tabel 4.10 Perhitungan persentase pertanyaan ke-5

Pertanyaan	Keterangan	Skor	Frekuensi	S
5	Sangat Setuju	5	10	50
	Setuju	4	17	68
	Ragu-ragu	3	3	9
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**127**

$$P = \frac{127}{150} \times 100\% = 84.67\%$$

Dapat disimpulkan bahwa penilaian terhadap ketertarikan pada permainan *The Kingdom of* Pandawa adalah 84,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.12.



Gambar 4.12 Interpretasi Skor Pertanyaan ke-5

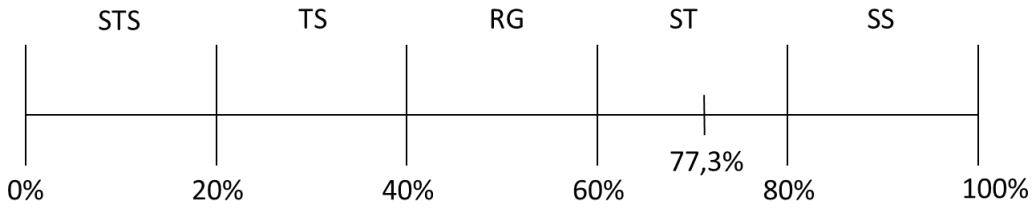
6. *Game* ini menarik untuk dimainkan secara berulang-ulang.

Tabel 4.11 Perhitungan persentase pertanyaan ke-6

Pertanyaan	Keterangan	Skor	Frekuensi	S
6	Sangat Setuju	5	6	30
	Setuju	4	14	56
	Ragu-ragu	3	10	30
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**116**

$$P = \frac{116}{150} \times 100\% = 77{,}3\%$$

Dapat disimpulkan bahwa penilaian terhadap ketertarikan pada permainan *The Kingdom of* Pandawa adalah 77,3% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.13.



Gambar 4.13 Interpretasi Skor Pertanyaan ke-6