BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi yang dibangun yaitu *game* Permainan Tradisional Galah. Tahap ini dilakukan setelah tahap perancangan selesai dilakukan. Adapun hasil dari perancangan tersebut kemudian diimplementasikan pada bahasa pemrograman. Setelah implementasi, maka dilakukan pengujian terhadap aplikasi tersebut selanjutnya. Pengujian tersebut terdiri dari pengujian *black box*, *white box*, dan beta. Analisis sistem ini merupakan cakupan dari analisis yang berhubungan dengan *game* yang akan dibangun.

4.1.1 Perangkat Pengujian Aplikasi

Terdapat dua jenis perangkat yang digunakan dalam pengujian aplikasi ini, yaitu perangkat keras dan perangkat lunak. Spesifikasi perangkat keras yang digunakan selama pengujian aplikasi ini memiliki spesifikasi sebagai berikut:

1. *Processor Dual Core*.
2. *RAM* 4GB.
3. *Hard Disk* 160GB.
4. *Mouse, Keyboard, LCD*

Sedangkan perangkat lunak yang digunakan selama pengujian aplikasi g*ame* Permainan Tradisional Galah adalah sistem operasi Windows 7 dan M*ozilla Firefox*.

4.1.2 Implementasi Instalasi *Game*

Sebelum dapat memainkan *game* Permainan Tradisional Galah, *game* harus tersimpan di direktori komputer. Petunjuk dalam instalasi *game* ini adalah sebagai berikut:

1. *Copy file index.html* pada direktori komputer.
2. Buka dan klik pada *file* tersebut.

3. Setelah itu, *game* tersebut akan menampilkan permainan.

4.1.3 Implementasi Antarmuka

Adapun antarmuka yang ada pada *game* Permainan Tradisional Galah:

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* Permainan Tradisional Galah. Terdapat 2 *sub menu* di dalamnya, yaitu menu mulai yang berfungsi untuk memulai permainan, menu tentang yang berfungsi untuk melihat biodata pembuat.

2. Antarmuka Menu Mulai

Saat pengguna memilih menu mulai, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka yang didalamnya terdapat menu lanjut dan cara bermain. Berikut ini merupakan tampilan dari antarmuka menu mulai:



Gambar 4.2 Antarmuka Arena Permainan

Gambar 4.2 merupakan tampilan dari arena permainan yang akan dimainkan oleh pengguna sebelum pengguna memainkan permainan.



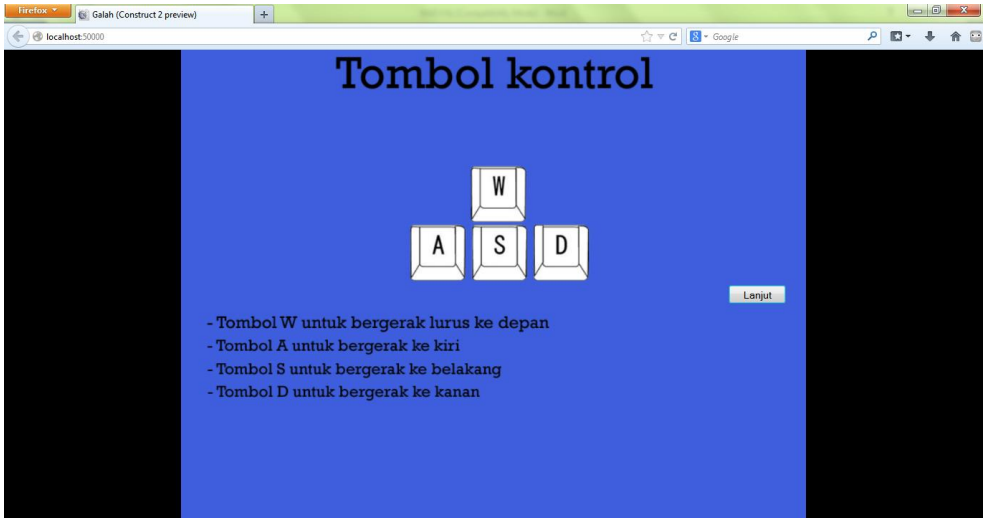
Gambar 4.3 Antarmuka Jalur Tempuh

Gambar 4.3 merupakan tampilan dari jalur permainan yang harus dilalui oleh pengguna sebelum pengguna memainkan permainan.



Gambar 4.4 Antar Muka Menu Cara Bermain

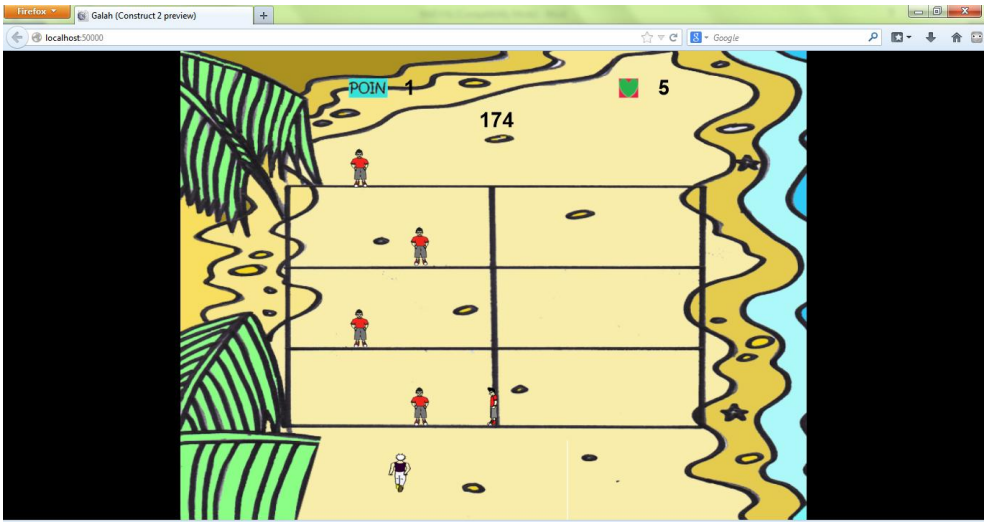
Gambar 4.4 merupakan tampilan dari cara bermain atau aturan permainan yang akan dimainkan oleh pengguna sebelum pengguna memainkan permainan.



Gambar 4.5 Antar Muka Menu Tombol yang Digunakan Saat Bermain

Gambar 4.5 merupakan tampilan dari tombol yang akan digunakan oleh pengguna pada saat pengguna memainkan permainan.

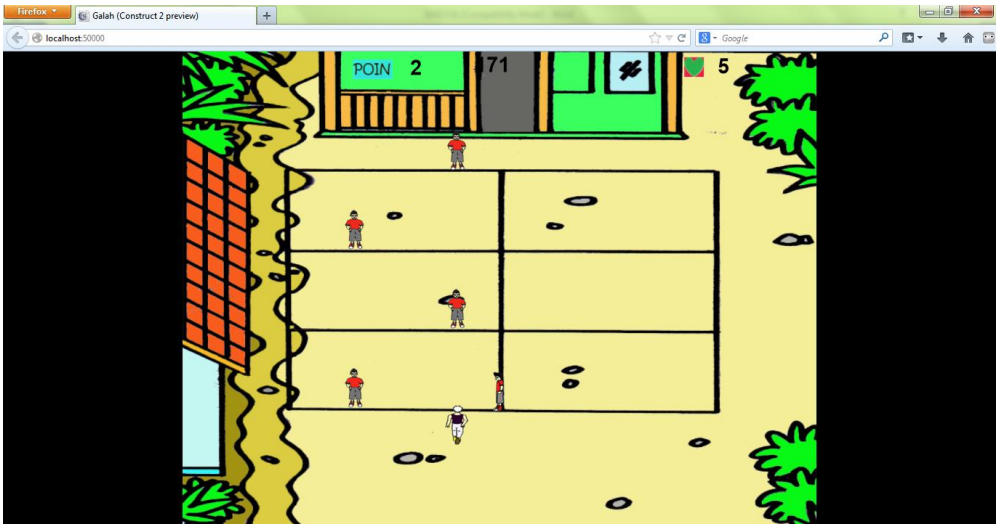
3. *Level* 1



Gambar 4.6 Tampilan *Level* 1

Gambar 4.6 merupakan tampilan arena *level* 1, ketika pemain menekan tombol lanjut pada menu intro maka sistem akan menampilkan *arena level* 1.

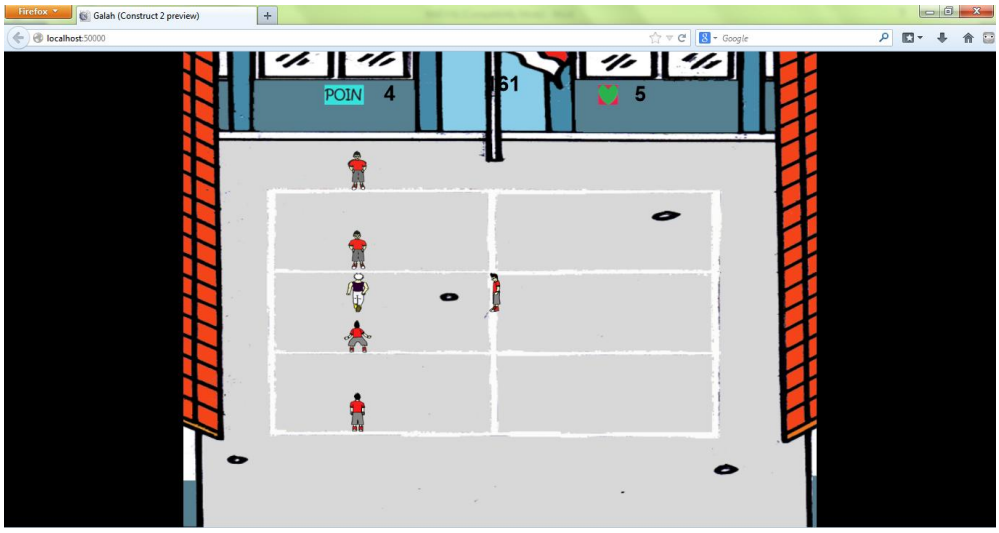
4. *Level* 2



Gambar 4.7 Tampilan *Level* 2

Gambar 4.7 merupakan tampilan arena *level* 2, ketika pemain berhasil menyelesaikan *level* 1 maka sistem akan menampilkan *arena level* 2.

5. *Level* 3



Gambar 4.8 Tampilan *Level* 3

Gambar 4.8 merupakan tampilan arena *level* 3, ketika pemain berhasil menyelesaikan *level* 2 maka sistem akan menampilkan *arena level* 3.

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* Permainan Tradisional Galah. Skenario pengujian yang akan dilakukan pada ini selengkapnya dapat dilihat pada tabel 4.1:

Tabel 4.1 Skenario Pengujian

Pengujian	Butir Uji	Jenis Pengujian
Menu Mulai	Memilih menu mulai	*Black Box*
	Menampilkan intro	*Black Box*
	Menampilkan petunjuk 1	*Black Box*
	Menampilkan petunjuk 2	*Black Box*
	Menampilkan cara main	*Black Box*
	Menampilkan arena *level* 1	*Black Box*
	Menampilkan arena *level* 2	*Black Box*
	Menampilkan arena *level* 3	*Black Box*
	Menggerakan *hero*	*Black Box*
	Hero menabrak musuh	*Black Box*
	Poin bertambah jika berhasil kembali atau ke garis akhir	*Black Box*
	Menampilkan *game over* jika waktu habis	*Black Box*
	Menampilkan *game over* jika lifepoint habis	*Black Box*
	Musuh mengikuti *hero*	*Black Box*
	Algoritma *greedy*	*White Box*
Menu Tentang	Menampilkan tentang	*Black Box*

4.2.1.2 Kasus dan Hasil Pengujian (*Black-Box*)

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*

Menu Mulai		
Masukan	Hasil yang diharapkan	Hasil pengujian
Memilih menu mulai	Menampilkan intro	[√] diterima [ ] ditolak
	Menampilkan petunjuk 1	[√] diterima [ ] ditolak
	Menampilkan petunjuk 2	[√] diterima [ ] ditolak
	Menampilkan cara main	[√] diterima [ ] ditolak
	Menampilkan arena *level* 1	[√] diterima [ ] ditolak
Tekan W	Hero bergerak ke atas	[√] diterima [ ] ditolak
Tekan A	Hero bergerak ke kiri	[√] diterima [ ] ditolak
Tekan S	Hero bergerak ke bawah	[√] diterima [ ] ditolak
Tekan D	Hero bergerak ke kanan	[√] diterima [ ] ditolak
Hero bergerak	Musuh mengikuti *hero*	[√] diterima [ ] ditolak
Hero menabrak musuh	*Lifepoint* berkurang	[√] diterima [ ] ditolak
Hero berhasil ke garis finish	Poin bertambah	[√] diterima [ ] ditolak
Berhasil mengumpulkan 3 poin	Menampilkan arena *level* 2	[√] diterima [ ] ditolak
Berhasil	Menampilkan arena *level* 3	[√] diterima [ ] ditolak

mengumpulkan 6 poin		
Lifepoint habis	Menampilkan *game over*	[√] diterima [ ] ditolak
Waktu habis	Menampilkan *game over*	[√] diterima [ ] ditolak
Menu Tentang		
Masukan	Hasil yang diharapkan	Hasil pengujian
Memilih menu tentang	Menampilkan tentang	[√] diterima [ ] ditolak

4.2.1.3 Kasus dan Hasil Pengujian (*White Box*)

Pada pengujian *white box* dilakukan proses pengujian pada proses jalur musuh yang merupakan proses pengujian dari pengimplementasian algoritma *greedy*, dan menguji apakah algoritma *greedy* berjalan dengan seharusnya pada musuh.

Karena pada *level* 1 tidak diterapkan algoritma *greedy* maka pengujian hanya dilakukan untuk *level* 2 dan *level* 3, pengujiannya adalah sebagai berikut:

1. Pengujian Permainan *Level* 2 (*white box*)

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

   a. Algoritma Yang Akan Diuji

Berikut ini merupakan algoritma *greedy* untuk *level* 2 bisa dilihat pada gambar 4.9.

```
1               for (i = 0, leni = tickarr.length; i < leni; i++)
2                 tickarr[i].tick();
3                 his.handleSaveLoad();
                this.isInOnDestroy--;
4                   var tickarr = this.objects_to_tick.valuesRef();
                    i = 0;

5                   while (this.changelayout && i++ != 10){
6                           this.doChangeLayout(this.changelayout);
                    }

7                   for (i = 0, leni = this.eventsheets_by_index.length; i < leni; i++)
                                        this.eventsheets_by_index[i].hasRun = false;
8                   if (this.running_layout.event_sheet)
9                   this.running_layout.event_sheet.run();
                    this.registered_collisions.length = 0;
                    this.layout_first_tick = false;
10                  this.isInOnDestroy++;

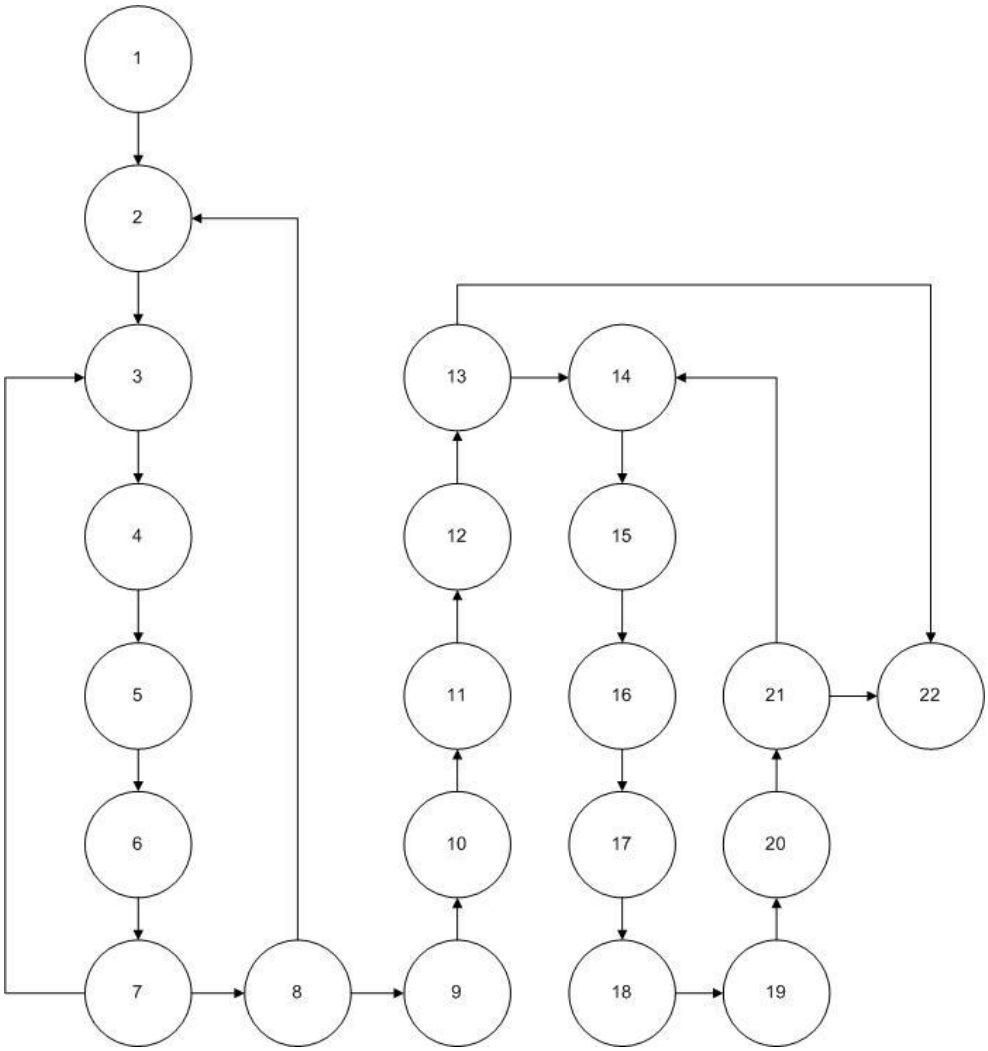
11                  for (i = 0, leni = this.types_by_index.length; i < leni; i++){
              type = this.types_by_index[i];
12                  if (type.is_family||(!type.behaviors.length&& !type.families.length))
              continue;
13                  for (j = 0, lenj = type.instances.length; j < lenj; j++){
              var inst = type.instances[j];
14                          for (k = 0, lenk = inst.behavior_insts.length; k < lenk; k++){
                      binst = inst.behavior_insts[k];
15                                  if (binst.tick2)
                                            binst.tick2();
16                            }
17                      }
18                  }

                    tickarr = this.objects_to_tick2.valuesRef();
19                  for (i = 0, leni = tickarr.length; i < leni; i++)
20                          tickarr[i].tick2();
21                          this.isInOnDestroy--;
22          };
```

Gambar 4.9 Algoritma *Greedy Level* 2

b. *Flowgraph* Permainan *Level* 2

Berikut ini merupakan *flowgraph* permainan *level* 2 yang dapat dilihat pada gambar berikut:



Gambar 4.10 *Flowgraph* Permainan *Level* 2

Keterangan:

= Menggambarkan Kondisi

→ = Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 25, N = 22

V(G) = E – N + 2

V(G) = 25 – 22 + 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),(8)

Path 2 = (1),(9),(10),(11),(12), (13)

Path 3 = (1),(13),(14),(15),(16),(17)

Path 4 = (1) ,(17),(18),(19),(20),(21)

Path 5 = (1),(19),(20),(21),(22)

e. *Graph Matriks* Permainan *Level* 2

Tabel 4.3 *Graph Matriks*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	N=(E-1)
1		1																					0
2			1																				0
3				1																			0
4					1																		0
5						1																	0
6							1																0
7			1					1															1
8		1							1														1
9										1													0
10											1												0
11												1											0

1																					
12												1									0
13													1							1	1
14														1							0
15															1						0
16																1					0
17																	1				0
18																		1			0
19																			1		0
20																				1	0
21													1							1	1
22																					0
SUM(E)+1																					4+1=5

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.

2. Pengujian Permainan *Level* 3 (*white box*)

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

a. Algoritma Yang Akan Diuji

Berikut ini merupakan algoritma *greedy* untuk *level* 3 bisa dilihat pada gambar 4.11:

```
1               for (i = 0, leni = tickarr.length; i < leni; i++)
2                 tickarr[i].tick();
3                 his.handleSaveLoad();
                this.isInOnDestroy--;
4                   var tickarr = this.objects_to_tick.valuesRef();
                    i = 0;

5                   while (this.changelayout && i++ != 10){
6                           this.doChangeLayout(this.changelayout);
                    }

7                   for (i = 0, leni = this.eventsheets_by_index.length; i < leni; i++)
                                        this.eventsheets_by_index[i].hasRun = false;
8                   if (this.running_layout.event_sheet)
9                   this.running_layout.event_sheet.run();
                    this.registered_collisions.length = 0;
                    this.layout_first_tick = false;
10                  this.isInOnDestroy++;

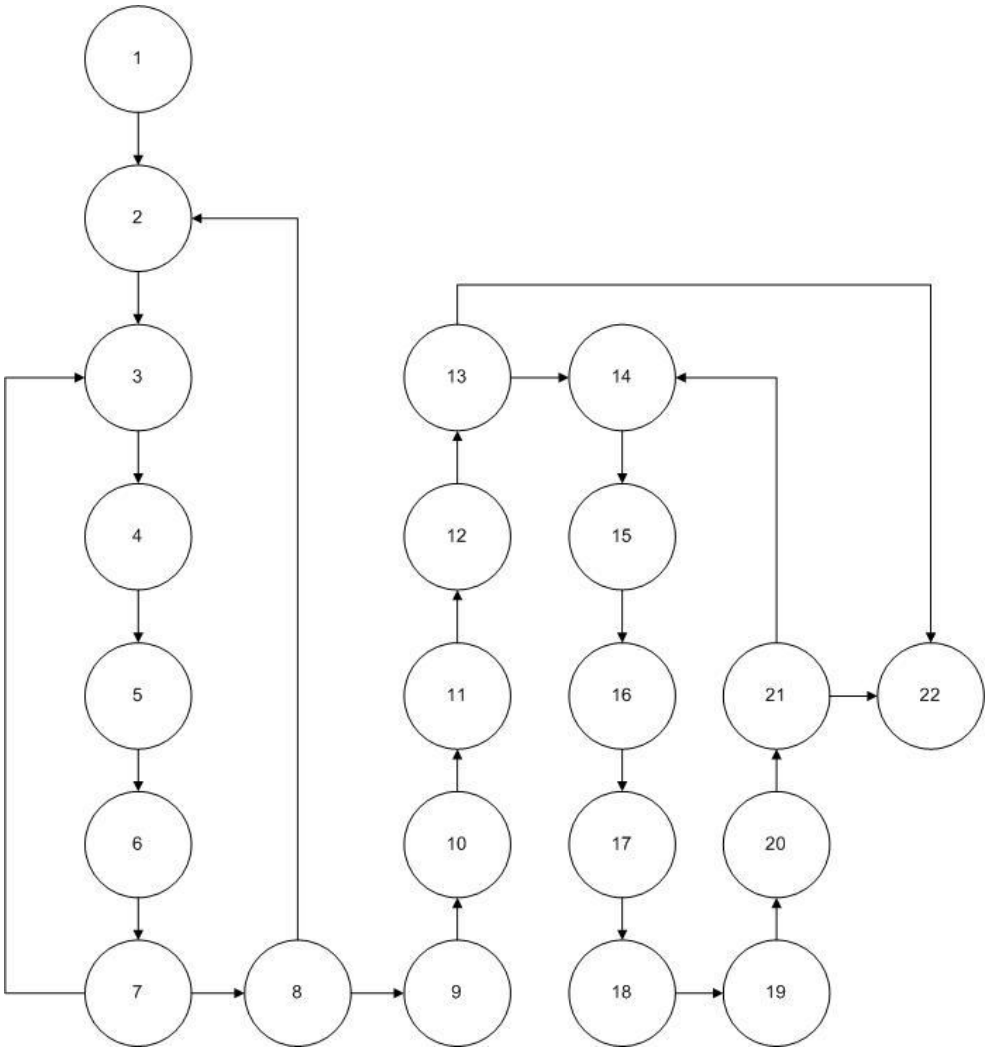
11                  for (i = 0, leni = this.types_by_index.length; i < leni; i++){
                type = this.types_by_index[i];
12                  if (type.is_family||(!type.behaviors.length && !type.families.length))
                continue;
13                  for (j = 0, lenj = type.instances.length; j < lenj; j++){
                var inst = type.instances[j];
14                          for (k = 0, lenk = inst.behavior_insts.length; k < lenk; k++){
                        binst = inst.behavior_insts[k];
15                                  if (binst.tick2)
                                            binst.tick2();
16                          }
17                      }
18                  }

                    tickarr = this.objects_to_tick2.valuesRef();
19                  for (i = 0, leni = tickarr.length; i < leni; i++)
20                          tickarr[i].tick2();
21                          this.isInOnDestroy--;
22          };
```

Gambar 4.11 Algoritma *Greedy Level* 3

b. *Flowgraph* Permainan *Level* 3

Berikut ini merupakan *flowgraph* permainan *level* 3 yang dapat dilihat pada gambar berikut:



Gambar 4.12 *Flowgraph* Permainan *Level* 3

Keterangan:

= Menggambarkan Kondisi

→ = Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 25, N = 22

V(G) = E – N + 2

V(G) = 25 – 22 + 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),(8)

Path 2 = (1),(9),(10),(11),(12), (13)

Path 3 = (1),(13),(14),(15),(16),(17)

Path 4 = (1) ,(17),(18),(19),(20),(21)

Path 5 = (1),(19),(20),(21),(22)

e. *Graph Matriks* Permainan *Level* 3

Table 4.4 *Graph Matriks*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	N=(E-1)
1		1																					0
2			1																				0
3				1																			0
4					1																		0
5						1																	0
6							1																0
7			1					1															1
8		1							1														1
9										1													0
10											1												0
11												1											0

1																					
12												1									0
13													1							1	1
14														1							0
15															1						0
16																1					0
17																	1				0
18																		1			0
19																			1		0
20																				1	0
21													1							1	1
22																					0
SUM(E)+1																					4+1=5

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 ditetapkan jawaban

Skor ideal = Nilai tertinggi dikalikan dengan jumlah sampel

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

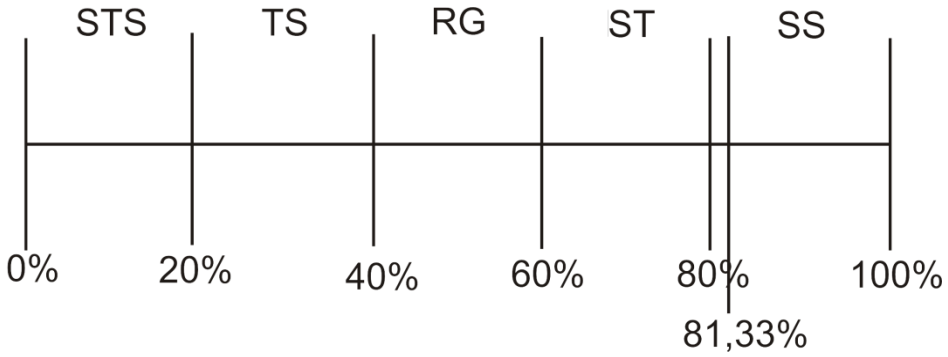
1. Apakah *game* ini mudah dimainkan?

Tabel 4.5 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 mudah tidaknya dimainkan pada *game* Permainan Tradisional Galah adalah 81, 67% 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-1

2. Apakah Anda kesulitan untuk menyelesaikan *game* ini?

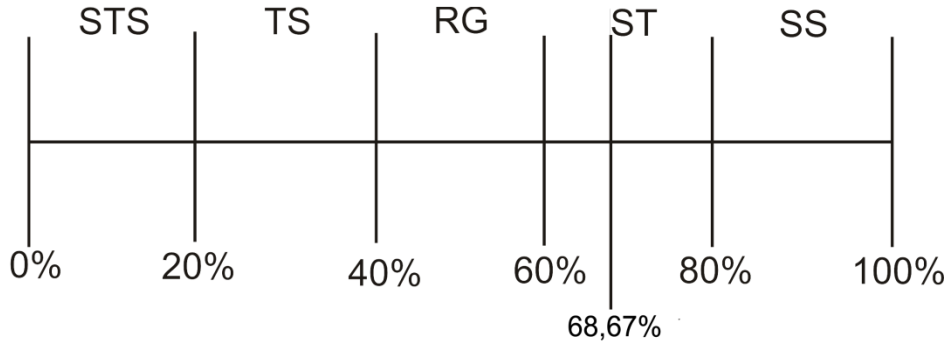
Tabel 4.6 Perhitungan persentase pertanyaan ke-2

Pertanyaan	Keterangan	Skor	Frekuensi	S
2	Sangat Setuju	5	0	0
	Setuju	4	16	64
	Ragu-ragu	3	11	33

Pertanyaan	Keterangan	Skor	Frekuensi	S
	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 *game* Permainan Tradisional Galah adalah 68,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.14.



Gambar 4.14 Interpretasi Skor Pertanyaan ke-2

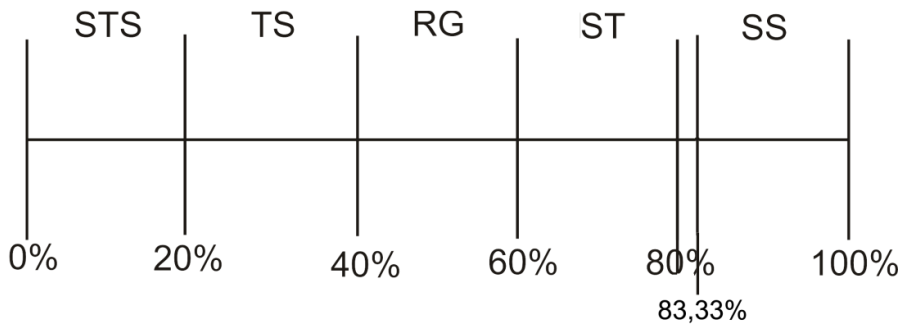
3. Apakah *game* ini membantu Anda mengetahui salah satu permainan tradisional Indonesia?

Tabel 4.7 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 pengetahuan pemain mengenai salah satu permainan tradisional Indonesia setelah memainkan *game* Permainan Tradisional Galah adalah 83,33% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.15.



Gambar 4.15 Interpretasi Skor Pertanyaan ke-3

4. Apakah skenario permainan yang dibuat sudah menarik untuk dimainkan?

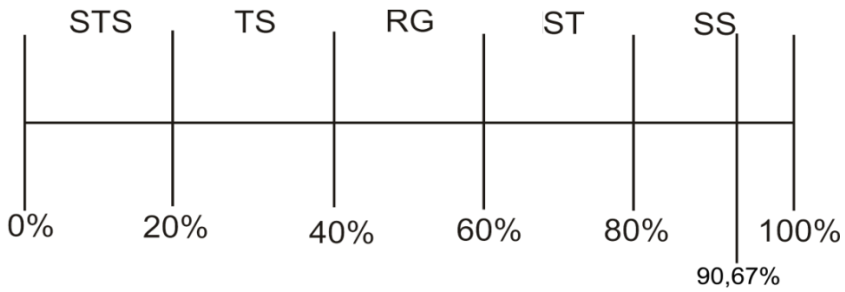
Tabel 4.8 Perhitungan persentase pertanyaan ke-4

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 skenario pada game Permainan Tradisional Galah adalah 90,67% dari yang diharapkan 100%, atau dapat

dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.16.



Gambar 4.16 Interpretasi Skor Pertanyaan ke-4

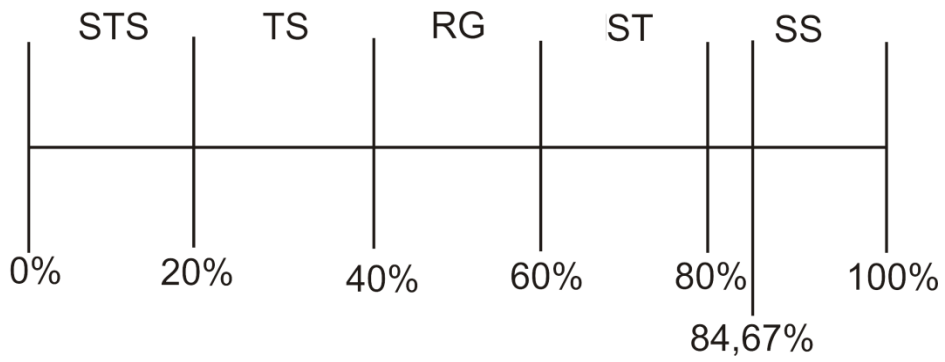
5. Apakah tampilan grafik 2D pada *game* ini sudah bagus?

Tabel 4.9 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 tampilan secara umum pada *game* Permainan Tradisional Galah adalah 84,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.17.



Gambar 4.17 Interpretasi Skor Pertanyaan ke-5

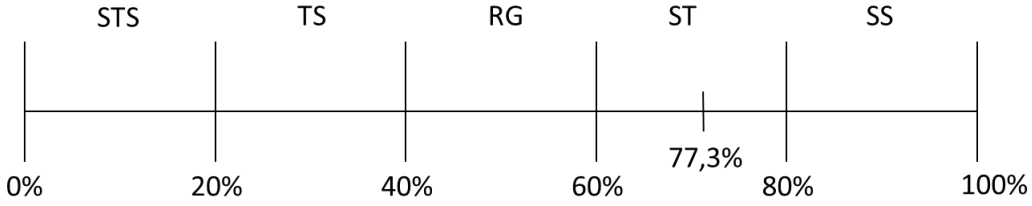
6. Apakah *game* ini menarik untuk dimainkan secara berulang-ulang?

Tabel 4.10 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 *game* Permainan Tradisional Galah adalah 77,3% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.18.



Gambar 4.18 Interpretasi Skor Pertanyaan ke-6

4.2.2.2 Kesimpulan Hasil Kuisioner

Hasil dari pengujian *beta* (kuisioner) dapat disimpulkan bahwa:

1. Pemain merasa sangat mudah dalam memainkan *game* ini.
2. Dengan kecerdasan yang dimiliki musuh, pemain merasa tingkat kesulitan game ini cukup menantang.
3. Pemain merasa mengetahui salah satu permainan tradisional Indonesia setelah bermain game ini.
4. Pemain merasa skenario yang dibuat pada *game* ini sudah menarik.
5. Pemain merasa tampilan *game* ini menarik dan cukup bagus.
6. Pemain merasa tertarik untuk memainkan *game* ini lagi.