BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi permainan hasil pembangunan yaitu *Game* V-Moe Attack. 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 *blackbox, whitebox,* dan *beta*.

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap kelanjutan dari kegiatan perancangan sistem. Wujud dari hasil implementasi ini nantinya adalah sebuah sistem yang siap untuk diuji dan digunakan.

4.1.1 Perangkat Kebutuhan Aplikasi

Untuk dapat menjalankan aplikasi ini ada spesifikasi perangkat keras dan lunak yang harus terpenuhi, penjelasannya sebagai berikut :

4.1.1.1 Perangkat Keras yang Digunakan

Spesifikasi perangkat keras (*hardware*) yang digunakan dalam pembangunan *game* V-Moe dapat dilihat pada tabel berikut.

Tabel 4.1 Perangkat keras yang digunakan

Perangkat Keras	Keterangan
Processor	AMD Phenom X6 1055t 2.8Ghz
Memory	DDR2 4GB
Harddisk	SATA 250 GB
Video Card	onboard
Monitor	Resolusi 1366x768
Mouse	Optic 1 buah
Keyboard	1 buah
Speaker	Advance

4.1.1.2 Perangkat Lunak yang Digunakan

Spesifikasi perangkat lunak (*software*) yang digunakan dalam pembangunan *game* V-Moe dapat dilihat pada tabel berikut.

Tabel 4.2 Perangkat Lunak yang Digunakan

Perangkat Lunak	Keterangan
Windows 7 Professional	Sebagai sistem operasi
Game Maker 8.0	Sebagai bahasa pemrograman
CorelDraw X5	Untuk mengolah gambar
Adobe Photoshop CS5	Untuk mengolah gambar
Microsoft Visio 2007	Perancangan antarmuka

4.1.2 Implementasi Antarmuka

Implementasi antar muka merupakan tahap dimana sebuah sistem siap dioperasikan pada tahap yang sebenarnya, sehingga bisa diketahui apakah sistem yang telah di buat ini sudah sesuai dengan apa yang sudah di rancang sebelumnya.

4.1.2.1 Tampilan Menu Utama Game

Ini merupakan tampilan menu utama *game* V-Moe, menu ini akan akan terlihat ketika aplikasi *game* dibuka. Dalam menu utama pemain dapat memulai permainan dengan memilih menu *play* atau dapat juga melihat petunjut permainan dengan memilih menu *how to play*. Tampilan menu utama *game* dapat dilihat gambar dibawah 4.1 :



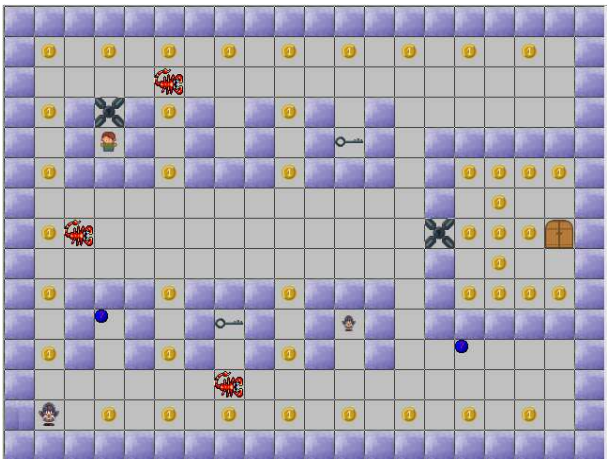
Gambar 4.1 Tampilan Menu Utama

Ketika pemain memilih menu *play* maka permainan akan dimulai, dan akan tampil arena permainan *level 1*. Pada *level* ini pemain berada di angkasa dan mengendalikan karakter dengan menggunakan *input keyboard*. Tampilan arena permainan *level 1* dapat dilihat pada gambar 4.2 :



Gambar 4.2 Tampilan *Game Level* 1

Setelah pemain berhasil selamat dalam *level 1*, maka pemain dapat melanjutkan permainan ke *level* berikutnya. Dalam *level 2* berada dalam markas para *monster*. Tampilan arena permainan pada *level 2* dapat dilihat pada gambar 4.3 :



Gambar 4.3 Tampilan *Game Level 2*

4.2 Pengujian Sistem

Pengujian sistem dilakukan bertujuan untuk menemukan kesalahan atau kekurangan pada perangkat lunak yang diuji. Pengujian bermaksud untuk

mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

Pengujian yang dilakukan yaitu dengan melakukan pengujian *blackbox*, *whitebox*, dan pengujian *beta*. Pengujian fungsional yang digunakan untuk menguji sistem yang baru adalah metode pengujian *blackbox*. Pengujian *blackbox* berfokus pada persyaratan fungsional perangkat lunak. Pengujian *whitebox* digunakan untuk meyakinkan semua perintah dan kondisi pada aplikasi dieksekusi secara minimal. Pengujian *whitebox* menggunakan dua *tools* yaitu *flow graph* yang digunakan untuk menggambarkan alur dari algoritma dan *graph matrix* yang digunakan untuk menggenerasi *flow graph*. Pengujian *beta* digunakan untuk mengetahui tanggapan user terhadap aplikasi *game*, dengan melakukan kuesioner.

4.2.1 Rencana Pengujian

Berikut ini adalah rencana-rencana pengujian yang akan dilakukan pada aplikasi *game* V-Moe. Rencana pengujian selengkapnya dapat di lihat pada tabel 4.3 :

Tabel 4.3 Rencana Pengujian Game

No	Komponen yang di Uji	Butir Uji	Jenis Uji
1	Gerak karakter	Bergerak ke Atas	*Black Box*
		Bergerak ke Bawah	*Black Box*
		Bergerak ke Kanan	*Black Box*
		Bergerak ke Kiri	*Black Box*
		Menembak	*Black Box*

2	Menu	Memilih Tombol *Play*	*Black Box*
		Memilih Tombol *How to play*	*Black Box*
		Memilih Tombol Highscore	*Black Box*
		Memilih Tombol About	*Black Box*
		Memilih Tombol Keluar *Game*	*Black Box*
3	Gerak Musuh AI	Penelusuran algorima A*	*White Box*

4.2.2 Kasus dan Hasil Pengujian

1. Pengujian Menu

Berikut ini adalah pengujian yang dilakukan untuk melihat menu game.

Tabel 4.4 Hasil Pengujian Menu

No	Kasus /Diuji	Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
1	Menu	Button *Play*	Menampilkan tampilan level 1 permainan	Mulai permainan	[√] Berhasil [ ] Tidak Berhasil
		Button *How to play*	Menampilkan menu petunjuk permainan	Sistem menampilkan info petunjuk permainan	[√] Berhasil [ ] Tidak Berhasil
		Button Highscore	Menampilkan nilai tertinggi dalam *game.*	Sistem menampilkan informasi nilai tertinggi dalam *game*.	[√] Berhasil [ ] Tidak Berhasil
		Button About	Menampilkan menu tentang pembuat aplikasi *game.*	Sistem menampilkan informasi aplikasi *game.*	[√] Berhasil [ ] Tidak Berhasil

		Button Exit	Keluar dari aplikasi *game* ketika pemain menekan tombol exit	Aplikasi *game* keluar	[√] Berhasil [ ] Tidak Berhasil

2. Pengujian Gerak Player

Berikut ini adalah pengujian yang dilakukan untuk melihat menu game.

Tabel 4.5 Tabel Pengujian Gerak Player

No	Kasus /Diuji	Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
2	Gerak *Play*er	Tombol ←	Menggerakkan *play*er ke arah kiri ketika pemain menekn tombol panah kiri	Karakter bergerak ke kiri	[√] Berhasil [ ] Tidak Berhasil
		Tombol →	Menggerakkan *play*er ke arah kanan ketika pemain menekan tombol panah kanan	Karakter bergerak ke k	[√] Berhasil [ ] Tidak Berhasil
		Tombol ↑	Menggerakkan *play*er ke arah atas ketika pemain menekan tombol panah atas	Karakter bergerak ke atas	[√] Berhasil [ ] Tidak Berhasil
		Tombol ↓	Menggerakkan *play*er ke arah bawah ketika pemain menekan tombol panah bawah	Karakter bergerak ke bawah	[√] Berhasil [ ] Tidak Berhasil
		Tombol ←	Menggerakkan *play*er ke arah kiri ketika pemain menekn tombol panah kiri	Karakter bergerak ke kiri	[√] Berhasil [ ] Tidak Berhasil

		Tombol Spasi	Karakter utama akan menembakan peluru ketika pemain menekan tombol spasi	Karakter menembakan peluru	[√] Berhasil [ ] Tidak Berhasil

Berdasarkan hasil pengamatan dan pengujian *blackbox* maka dapat disimpulkan bahwa fungsional dalam *game* berjalan sebagai mana yang diharapkan.

3. Pengujian penelusuran algoritma A* pada musuh

Pengujian proses algoritma A* merupakan proses pengujian dari pengimplementasian algoritma A* menggunakan pengujian *whitebox*, dan menguji apakah algoritma A* (A Star) berjalan dengan seharusnya pada *enemy*.

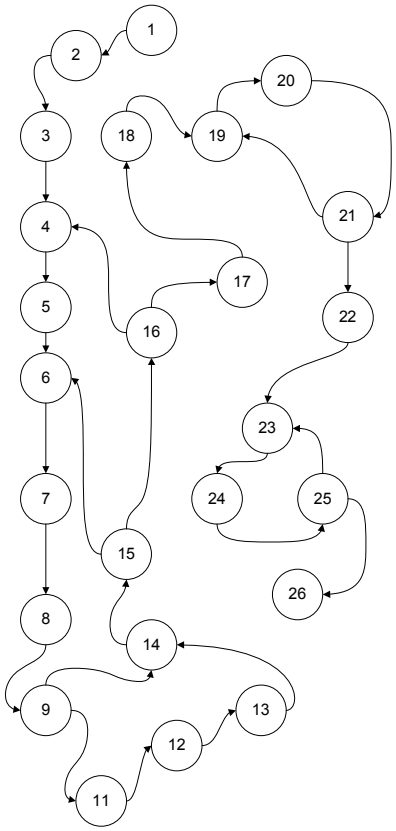
a. Algoritma

```
1  inisialisasi
   start = argument0;
   dst = argument1;
   OPEN = ds_prt_create();
   OPEN_LIST = ds_list_create();
   CLOSED = ds_list_create();
   ds_temp = ds_prt_create();
2  if (start == dst)
3  return ds_temp;
   start.H = abs(start.x-dst.x) + abs(start.y-dst.y);
   start.F = start.G + start.H;
   ds_prt_add(OPEN,start,start.F);
   ds_list_add(OPEN_LIST,start);
4  do {
5  current = ds_prt_delete_min(OPEN);
   pos = ds_list_find_index(OPEN_LIST,current);
   ds_list_delete(OPEN_LIST,pos);
```

```
   ds_list_add(CLOSED,current);
6  for (i=0; i<ds_list_size(current.ngs); i+=1) {
7      ng = ds_list_find_value(current.ngs,i);
       ng.checked = true;
8      if (ds_list_find_index(CLOSED,ng)== -1&&ng.passable==
true) {
9          if (ds_list_find_index(OPEN_LIST,ng) == -1) {
10             ng.parent = current;
               ng.G = current.G + ng.cost;
               ng.H = abs(ng.x-dst.x) + abs(ng.y-dst.y);
               ng.F = ng.G + ng.H;
               ds_prt_add(OPEN,ng,ng.F);
               ds_list_add(OPEN_LIST,ng);
11         }else {
               temp_G = current.G + ng.cost;
12             if (temp_G < ng.G) {
13                   ng.parent = current;
                     ng.G = current.G + ng.cost;
                     ng.H = abs(ng.x-dst.x) + abs(ng.y-dst.y);
                     ng.F = ng.G + ng.H;
                  }end if
14            }end if
          }end if
15     }end for
16 }end do until (ds_list_find_index(CLOSED,dst) != -1 ||
ds_prt_size(OPEN) == 0)
17 var mark_path,i,last,nearest,cnd;
   i = 0;
18 if (ds_list_find_index(CLOSED,dst) == -1) {
       nearest = start;
19     for (n=0; n<int_number(node); n+=1) {
           cnd = int_find(node,n);
           cnd.H = abs(cnd.x-dst.x)/32 + abs(cnd.y-dst.y)/32;
20         if ( cnd.H < nearest.H &&
ds_list_find_index(CLOSED,cnd) != -1)
               nearest = cnd;
21     }end for
       dst = nearest;
```

```
   }end if
22 if (start == dst)
       return ds_temp;
       ds_prt_add(ds_temp,dst,i);
23 do {
24     last = ds_prt_find_max(ds_temp);
       i += 1;
       ds_prt_add(ds_temp, last.parent, i);
25 } until (last.parent == start)
       ds_list_destroy(OPEN_LIST);
       ds_list_destroy(CLOSED);
       ds_prt_destroy(OPEN);
       return ds_temp;
26 end
```

b. Flow Graph



Gambar 4.4 Flow Graph A*

Keterangan :

 = Menggambarkan kondisi

 = Menggambarkan aksi

c. Cyclomatic Complexity (V(G)

V(G) = E – N + 2

V(G) = 30 – 26 + 2

V(G) = 4 + 2

V(G) = 6

Di mana :

E = jumlah edge pada grafik alir

N = jumlah node pada grafik alir

d. Independent path

Path 1 = 1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-22-23-24-25-26

Path 2 = 1-2-3-4-5-6-7-8-9-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26

Path 3 = 1-2-3-4-5-6-7-8-9-10-14-15-6-7-8-9-10-14-15-16-17-18-19-20-21-22-23-24-25-26

Path 4 = 1-2-3-4-5-6-7-8-9-10-14-15-16-4-5-6-7-8-9-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26

Path 5 = 1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-19-20-21-22-23-24-25-26

Path 6 = 1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-22-23-24-25-23-24-25-26

e. Graph Matrices

Berikut ini adalah graph matrices dari algoritma A* yang dapat dilihat pada tabel 4.6.

Tabel 4.6 Graph Matrices A*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	Hasil
1		1																									0
2			1																								0
3				1																							0
4					1																						0
5						1																					0
6							1																				0
7								1																			0
8									1																		0
9										1	1																1
10														1													0
11												1															0
12													1														0
13														1													0
14															1												0
15						1										1											1
16				1													1										1
17																		1									0
18																			1								0
19																				1							0
20																					1						0
21																			1			1					1
22																							1				0
23																								1			0
24																									1		0
25																							1			1	1
26																											Sum(E)+1=6 5+1=6

Berikut ini adalah hasil dari pengujian *whitebox* penelusuran algoritma A* pada objek musuh pada aplikasi *game* V-Moe.

Tabel 4.7 hasil pengujian white box

Path no.	Node(n)	Expected Result	Test Result	Conclusion
1	1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-22-23-24-25-26	Menampilkan rute ke *player*	Menampilkan rute ke *player*	Path dikerjakan

2	1-2-3-4-5-6-7-8-9-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26	Mengecek node dalam *OPEN list*	Node dalam *OPEN list* terdefinisi	Path dikerjakan
3	1-2-3-4-5-6-7-8-9-10-14-15-6-7-8-9-10-14-15-16-17-18-19-20-21-22-23-24-25-26	Mengecek node bukan berada dalam *OPEN list*	Node dalam *OPEN list* terdefinisi	Path dikerjakan
4	1-2-3-4-5-6-7-8-9-10-14-15-16-4-5-6-7-8-9-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26	Membandingkan Nilai F antara *node* awal dengan *node* tetangga	*Node* tetangga menjadi *current node*	Path dikerjakan
5	1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-19-20-21-22-23-24-25-26	Mencari nilai F setiap *node* terkecil dalam antrian	Nilai node yang mempunyai F terkecil terdefinisi	Path dikerjakan
6	1-2-3-4-5-6-7-8-9-10-14-15-16-17-18-19-20-21-22-23-24-25-23-24-25-26	Mengecek *node* terakhir sama dengan *node* awal	Node akhir = node awal	Path dikerjakan

Berdasarkan hasil pengamatan dan pengujian path maka dapat disimpulkan beberapa poin sebagai berikut :

1. Seluruh path dilalui dan dieksekusi
2. Setiap kondisi setidaknya diproses satu kali
3. Tidak ada perulangan tak hingga

4.2.3 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif, dimana pengujian dilakukan dengan mecoba aplikasi secara langsung ke kalangan

yang memang menjadi target *user* atau pengguna aplikasi *game* ini. Pengujian dilakukan dengan cara membagikan kuesioner kepada para pengguna guna mendapatkan data tentang kepuasan pengguna terhadap *game* V-Moe ini.

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 25 orang , dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian kepuasan terhadap aplikasi *game* V-Moe. Kuesioner terdiri dari 5 pernyataan dengan menggunakan skala 1 sampai 4 (skala likert).

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

$$P = \frac{S}{skorideal} \times 100$$

Keterangan :

P = Nilai persentase yang dicari

S = Jumlah frekuensi dikalikan dengan skor yang ditetapkan tiap jawaban

skorideal = Skor tertinggi dikalikan dengan jumlah sampel

Berikut adalah kuesioner yang disebarkan ke target user dan harus di isi oleh responden :

1) Apakah dengan 2 jenis gameplay berbeda game mejadi lebih interaktif ?
    a) Sangat Setuju
    b) Setuju
    c) Kurang Setuju
    d) Tidak Setuju
2) Apakah musuh dalam game ini lebih pintar?
    a) Sangat Setuju
    b) Setuju
    c) Kurang Setuju
    d) Tidak Setuju
3) Apakah anda kesulitan menyelesaikan misi dalam game ini?
    a) Sangat Setuju
    b) Setuju
    c) Kurang Setuju
    d) Tidak Setuju
4) Apakah kontrol karakter dalam game ini mudah dikendalikan?
    a) Sangat Setuju
    b) Setuju
    c) Kurang Setuju
    d) Tidak Setuju
5) Apakah secara umum menurut anda aplikasi ini sudah menarik?Sangat Setuju
    a) Sangat Setuju
    b) Setuju

c) Kurang Setuju

d) Tidak Setuju

4.2.3.1 Hasil Pengujian Beta adalah sebagai berikut:

1. Apakah dengan 2 jenis *gameplay* berbeda game mejadi lebih interaktif?

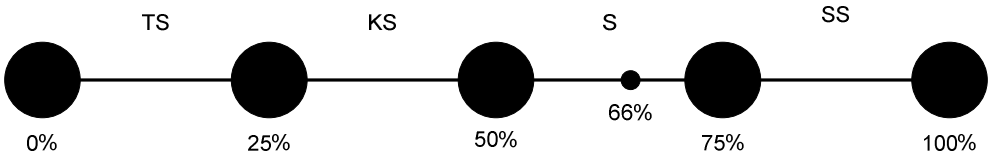
   Hasil kuesioner untuk pertanyaan ke-1 dapat dilihat pada tabel 4.8.

Tabel 4.8 *Perhitungan Persentase Pernyataan ke-1*

Pernyataan	Keterangan	Skor	Frekuensi	S
1	Sangat Setuju	4	2	8
	Setuju	3	14	42
	Kurang Setuju	2	8	16
	Tidak Setuju	1	1	1
Jumlah			25	66

$$P = \frac{66}{100} \times 100\% = 66\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap aplikasi *game* V-Moe adalah 64% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.6 :



Gambar 4.5 *Interpretasi Skor Pernyataan ke-1*

2. Apakah musuh dalam *game* ini lebih pintar?

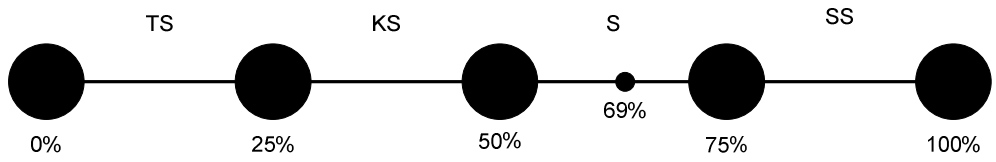
Hasil kuesioner untuk pertanyaan ke-2 dapat dilihat pada tabel 4.9.

Tabel 4.9 ***Perhitungan Persentase Pernyataan ke-2***

Pernyataan	Keterangan	Skor	Frekuensi	S
2	Sangat Setuju	4	1	4
	Setuju	3	18	54
	Kurang Setuju	2	5	10
	Tidak Setuju	1	1	1
Jumlah			25	69

$$P = \frac{69}{100} \times 100\% = 69\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap aplikasi *game* V-Moe adalah 57% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.7.



Gambar 4.6 ***Interpretasi Skor Pernyataan ke-2***

3. Apakah anda kesulitan menyelesaikan misi dalam game ini?

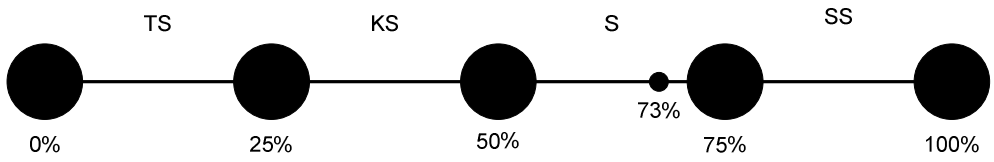
   Hasil kuesioner untuk pertanyaan ke-3 dapat dilihat pada tabel 4.10.

Tabel 4.10 ***Perhitungan Persentase Pernyataan ke-3***

Pernyataan	Keterangan	Skor	Frekuensi	S
3	Sangat Setuju	4	9	36
	Setuju	3	8	24
	Kurang Setuju	2	5	10
	Tidak Setuju	1	3	3
Jumlah			25	73

$$P = \frac{73}{100} \times 100\% = 73\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap aplikasi *game* V-Moe adalah 73% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.8.



Gambar 4.7 ***Interpretasi Skor Pernyataan ke-3***

4. Apakah kontrol karakter dalam game ini mudah dikendalikan? *

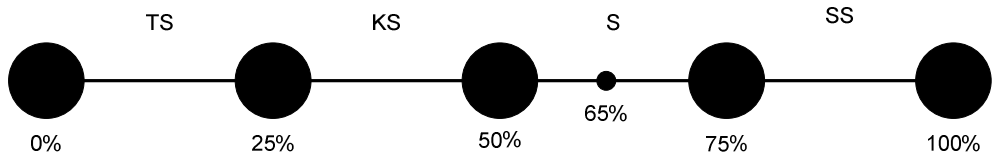
Hasil kuesioner untuk pertanyaan ke-4 dapat dilihat pada tabel 4.11.

Tabel 4.11 ***Perhitungan Persentase Pernyataan ke-4***

Pernyataan	Keterangan	Skor	Frekuensi	S
4	Sangat Setuju	4	2	8
	Setuju	3	15	45
	Kurang Setuju	2	4	8
	Tidak Setuju	1	4	4
Jumlah			25	65

$$P = \frac{65}{100} \times 100\% = 65\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap aplikasi *game* V-Moe adalah 65% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.9.



Gambar 4.8 ***Interpretasi Skor Pernyataan ke-4***

5. Apakah secara umum menurut anda aplikasi ini sudah menarik?

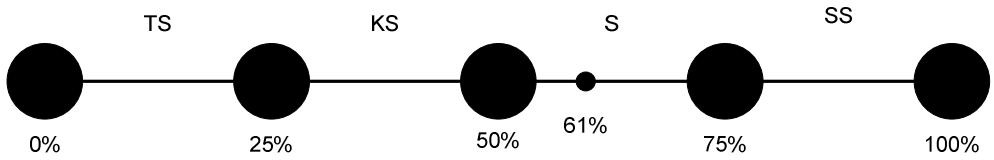
Hasil kuesioner untuk pertanyaan ke-5 dapat dilihat pada tabel 4.12.

Tabel 4.12 ***Perhitungan Persentase Pernyataan ke-5***

Pernyataan	Keterangan	Skor	Frekuensi	S
5	Sangat Setuju	4	1	4
	Setuju	3	13	39
	Kurang Setuju	2	7	14
	Tidak Setuju	1	4	4
Jumlah			25	61

$$P = \frac{61}{100} \times 100\% = 61\%$$

Berdasarkan hasil persentase nilai diatas, maka dapat disimpulkan bahwa penilaian terhadap aplikasi *game* V-Moe adalah 61% dari yang diharapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.10.



Gambar 4.9 ***Interpretasi Skor Pernyataan ke-5***

4.2.4 Kesimpulan Pengujian Beta

Berdasarkan pengujian beta yang telah didapat dari hasil kuesioner yang telah dijawab oleh para responden, maka bisa ditarik kesimpulan bahwa *game* V-Moe ini mempunyai musuh yang cukup pintar. Pergerakan karakter juga mudah untuk dikendalikan, misi permainan yang sulit untuk diselesaikan dan mempunyai dua jenis *gameplay* berbeda dalam *game* sudah cukup sehingga secara keseluruhan menjadikan game menjadi lebih menarik.