BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

Setelah melakukan analisis dan perancangan secara terperinci maka selanjutnya akan dilakukan tahap implementasi. Implementasi merupakan penerapan cara kerja sebuah sistem berdasarkan hasil analisis berdasarkan rancangan yang sudah dibuat. Bahasa pemrograman dalam sistem ini menggunakan bahasa *Java* dan *Corel Draw X4* sebagai *tools* untuk desain gambar.

4.1.1 Perangkat Kebutuhan Aplikasi

Untuk menjalankan *game* ini tentunya membutuhkan perangkat keras (*Hardware*) dan perangkat lunak (*software*). Berikut merupakan penjelasan dari perangkat keras dan perangkat lunak yang digunakan.

4.1.1.1 Perangkat Keras

Berikut ini merupakan spesifikasi perangkat keras yang di gunakan untuk menjalankan *game destroyer* garuda ini :

1. Processor : Intel Core 2 Duo
2. *Harddisk* : 500 gb
3. RAM : 2 gb
4. VGA : 512 mb
5. Monitor
6. *Mouse, keyboard dan speaker*

4.1.1.2 Perangkat Lunak

Berikut ini merupakan spesifikasi perangkat lunak yang di gunakan untuk menjalankan *game* destroyer garuda ini :

1. Sistem Operasi Windows XP, Windows 7 dan sistem operasi lainnya
2. Download JDK 1.6 ke atas.

3. Install JDk
4. Download Greenfoot 2.0 ke atas.
5. Install Greenfoot.
6. Klik file DestroyerGarudaFix.jar

4.1.2 Identifikasi Kelas

Identifikasi kelas dapat di lihat dari tabel 4.1 :

Tabel 4. 1 Identifikasi Kelas

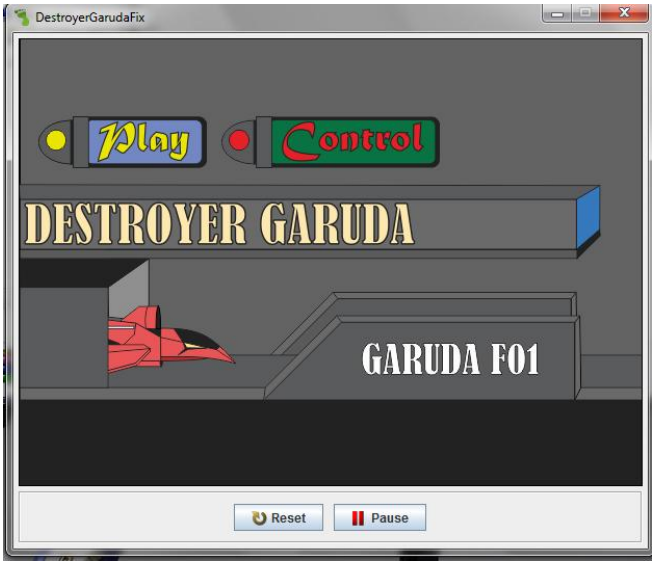
Nama Kelas	Nama File Fisik
Background	Background.java
Enemy	Enemy.java
*Game*Over	*Game*Over.java
Bcontrol	Bcontrol.java
Finish	Finish.java
Health	Health.java
GarudaF01RMode	GarudaF01Rmode.java
MbHealth	MbHealth.java
MidiPlayer	MidiPlayer.java
Bback	Bback.java
ControlScreen	ControlScreen.java
Bplay	Bplay.java
Mover	Mover.java
Ufo	Ufo.java
Bullet	Bullet.java
Bullet2	Bullet2.java
ShooterUfo	ShooterUfo.java
Bos	Bos.java
RobotUfo	RobotUfo.java

4.1.3 Implementasi Antarmuka

Implementasi antarmuka merupakan tahap dimana sistem sebuah sistem siap untuk di jalankan pada tahap yang sebenarnya.

4.1.3.1 Tampilan Menu Utama

Pada tampilan ini terdapat 2 menu yaitu *play* dan *control*, menu *play* untuk masuk ke level 1 sedangkan menu *control* untuk melihat tombol-tombol untuk memainkan *game* ini. Untuk lebih jelasnya dapat dilihat pada gambar 4.1



Gambar 4.1 Tampilan menu utama

4.1.3.2 Tampilan *Control*

Pada tampilan ini berisi mengenai informasi tombol-tombol yang di gunakan dalam *game* ini dan juga menu *back* untuk kembali ke tampilan menu utama. Untuk lebih jelasnya dapat di lihat pada gambar 4.2 :



Gambar 4.2 Tampilan *control*

4.1.3.3 Tampilan *Level* 1

Pada tampilan ini berisi mengenai tampilan *level* 1 *game*, yang di mana di *level* ini *player* harus menghadapi musuh ufo dan juga robot ufo. Untuk lebih jelasnya dapat di lihat pada gambar 4.3 :



Gambar 4.3 Tampilan *level* 1

4.1.3.4 Tampilan *Level* 2

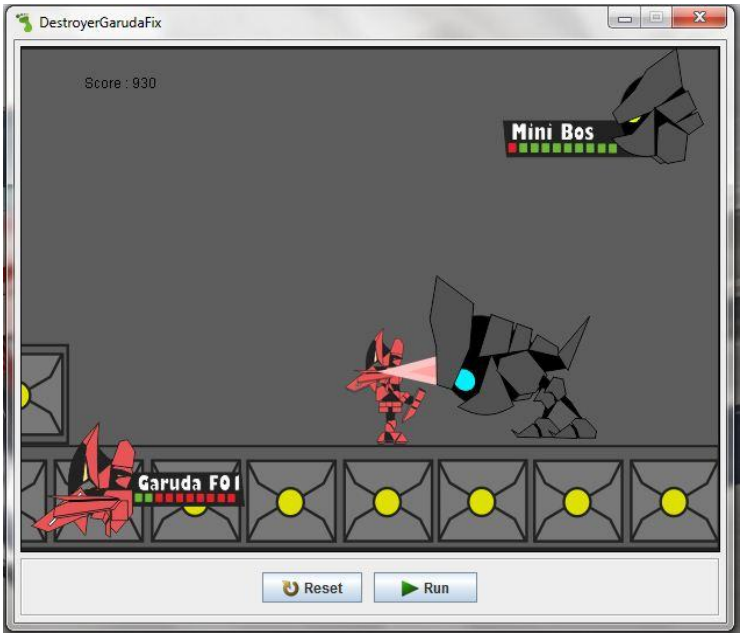
Pada tampilan ini berisi mengenai tampilan *level* 2 *game,* yang di mana di *level* ini *player* harus menghadapi musuh ufo dan juga *shooter* ufo. Untuk lebih jelasnya dapat di lihat pada gambar 4.4 :



Gambar 4.4 Tampilan *level* 2

4.1.3.5 Tampilan *Level* 3

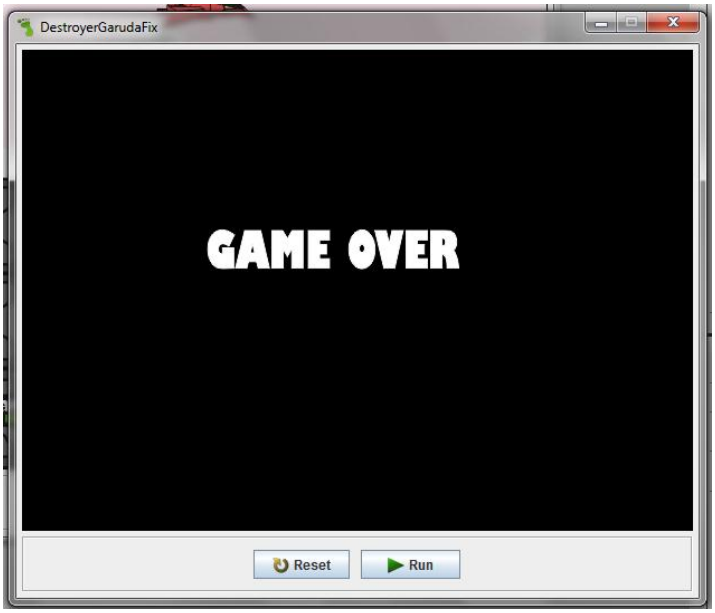
Pada tampilan ini berisi mengenai tampilan *level* 3 *game,* yang di mana di *level* ini *player* harus menghadapi musuh ufo, robot ufo, *shooter* ufo dan juga bos. Untuk lebih jelasnya dapat di lihat pada gambar 4.5 :



Gambar 4.5 Tampilan *level* 3

4.1.3.6 Tampilan *Game Over*

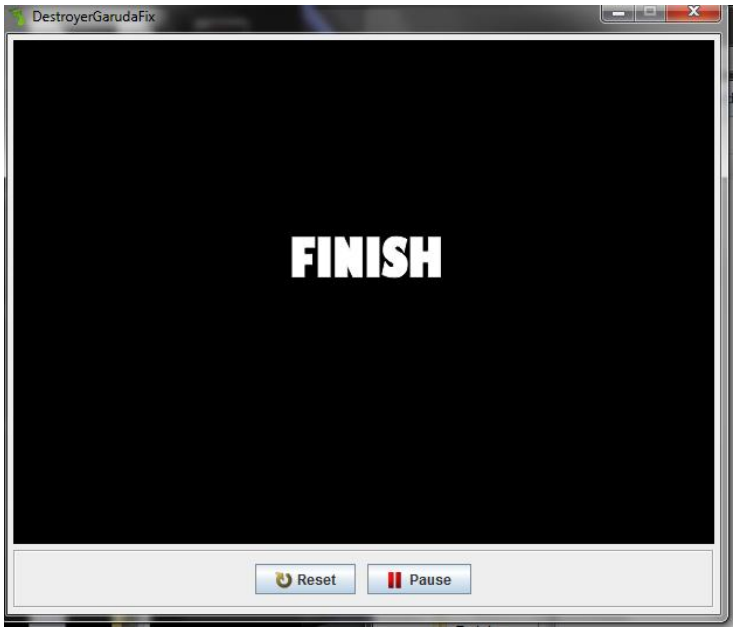
Pada tampilan ini menandakan berakhirnya permainan karena *player* mati. Untuk lebih jelasnya dapat di lihat pada gambar 4.6 :



Gambar 4. 6 Tampilan *game over*

4.1.3.7 Tampilan *Finish*

Pada tampilan ini menandakan akhir dari permainan karena *player* berhasil mengalahkan bos. Untuk lebih jelasnya dapat di lihat pada gambar 4.7 :



Gambar 4.7 Tampilan *finish*

4.1 Pengujian Sistem

Pengujian sistem merupakan hal terpenting yang dilakukan untuk menemukan kekurangan atau kesalahan pada perangkat lunak yang diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

4.2.1 Pengujian *Alpha* (fungsional)

Pengujian fungsional yang digunakan untuk menguji sistem yang baru adalah dengan menggunakan metode pengujian *alpha*. Pengujian *alpha* berfokus pada persyaratan fungsional perangkat lunak.

a. Rencana Pengujian

Renacana pengujian yang akan dilakukan pada aplikasi ini selengkapnya terlihat pada tabel 4.2:

Tabel 4.2 Rencana Pengujian Aplikasi

Kelas Uji	Detail Pengujian	Jenis Uji
Play	Memulai permainan	*Black Box*
Control	Menampilkan tampilan petunjuk/control	*Black Box*
Karakter Bergerak	Pergerakan karakter menggunakan tombol-tombol dalam aplikasi	*Black Box*
Musuh	Pergerakan musuh dalam aplikasi	*Black Box*
Suara	Menampilkan suara dalam aplikasi	*Black Box*
Gerakan musuh AI	Penelusuran algorima Djikstra	*White Box*

b. Kasus dan hasil pengujian

Berdasarkan hasil pengujian yang telah disusun, maka dapat dilakukan pengujian sebagai berikut :

1. Pengujian Mulai

Pengujian ini dilakukan untuk memulai aplikasi *game* yang telah dibuat

Tabel 4.3 Pengujian Menu Mulai

Kasus dan hasil uji (data normal)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Pilih Menu "Play"	Memulai permainan level 1	Permainan dimulai	[√] Tampil [ ] Tidak

2. Pengujian Petunjuk

Pengujian ini dilakukan untuk melihat informasi dan petunjuk yang ada di dalam *game*.

Tabel 4.4 Pengujian Menu Petunjuk

Kasus dan hasil uji (data normal)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Pilih Menu "Control"	Melihat informasi dan petunjuk untuk memainkan permainan	Petunjuk permainan	[√] Tampil [ ] Tidak

3. Pengujian Karakter Bergerak

Pengujian ini dilakukan untuk menampilkan informasi pergerakan karakter yang terdapat dalam *game*.

Tabel 4.5 Pengujian Karakter Bergerak

Kasus dan hasil uji (data normal)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Tombol →	Garuda bergerak ke arah kanan	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol ←	Garuda bergerak ke arah kiri	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol ↑	Garuda lompat	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol 1	Garuda berubah menjadi mode pesawat	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol 2	Garuda berubah menjadi mode robot	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol R	Garuda mode pesawat menembak	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol Q	Garuda mode robot menyerang	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol W	Garuda mode robot menyerang	Pergerakan garuda	[√] Bergerak [ ] Tidak
Tombol E	Garuda mode robot menyerang	Pergerakan garuda	[√] Reset [ ] Tidak

Karakter jatuh	Menampilkan tombol reset	Tampil tombol reset	[√] Tampil [ ] Tidak

4. Pengujian Musuh

Pengujian ini dilakukan untuk menampilkan pergerakan musuh yang terdapat dalam *game*.

Tabel 4.6 Pengujian Musuh

Kasus dan hasil uji (data normal)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Karakter menghancurkan musuh	Karakter mati, tampil tombol reset	Pergerakan musuh	[√] Tampil [ ] Tidak

5. Pengujian Suara

Pengujian ini dilakukan untuk menampilkan bunyi suara atau musik yang terdapat dalam *game*

Tabel 4.7 Pengujian Suara

Kasus dan hasil uji (data normal)			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Karakter menyerang musuh	Menampilkan bunyi suara serangan	Tampil bunyi suara atau musik serangan	[√] Tampil [ ] Tidak
Musuh menyerang karakter	Menampilkan bunyi suara serangan	Tampil bunyi suara atau musik serangan	[√] Tampil [ ] Tidak
Musuh mati	Menampilkan bunyi suara	Tampil bunyi suara atau musik	[√] Tampil

	ledakan	ledakan	[ ] Tidak
Karakter berubah mode	Menampilkan bunyi suara berubah mode	Tampil bunyi suara atau musik berubah mode	[√] Tampil [ ] Tidak
Mengklik tombol start	Menampilkan bunyi suara atau musik awal	Tampil bunyi suara atau musik awal	[√] Tampil [ ] Tidak
Masuk *stage* pertama	Menampilkan bunyi suara atau *backcound*	Tampil *backsound*	[√] Tampik [ ] Tidak

4.2.2 Pengujian *White Box*

White box adalah meramalkan cara kerja perangkat lunak secara rinci, karenanya *logikal path* (jalur logika) perangkat lunak akan ditest dengan menyediakan *test case* yang akan mengerjakan kumpulan kondisi dan atau pengulangan secara spesifik. Penentuan kasus uji disesuaikan dengan struktur sistem, pengetahuan aplikasi yang digunakan untuk mengindentifikasikan kasus uji tambahan.

1. Algoritma Dijkstra

Menguji apakah algoritma ini berjalan dengan baik pada musuh yang berjalan di darat pada *game* ini.

a. Source code

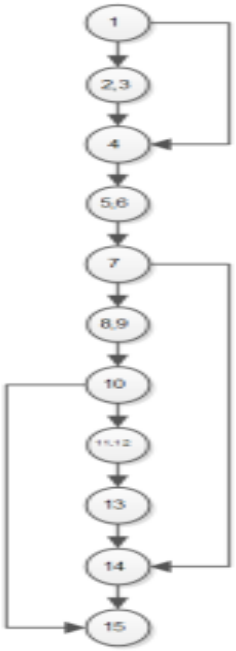
Berikut ini merupakan algoritma djikstra pada musuh dengan menggunakan bahasa pemrograman Java:

```
1.  for i ← 1 to n do      //langkah 0 inisialisai
2.  si ← 0
3.  di ← mai
4.  endfor
```

```
5.  sa ← 1                  // langkah 1 simpul a adalah
    asal lintasan terpendek, simpul a terpilih dalam
    lintasan terpendek
6.  da ← ∞                 // tidak ada lintasan terpendek
    dari simpul a ke a
7.  for k ← 2 to n-1 do    // langkah 2,3…,n-1
8.  j ← simpul dengan sj= 0 dan dj minimal
9.  sj ← 1b               // simpul j sudah terpilih kedalam
    lintasan terpendek
10. for semua simpul i dengan si= 0 do //perbarui table d
11. if dj+ mji < di then
12. di ← dj + mji
13. endif
14. endfor
15. endfor
```

---

b. Flow Graph



Gambar 4. 8 Flow graph pengujian algoritma djikstra

Keterangan :

 = Menggambarkan kondisi

 = Menggambarkan aksi

c. *Cyclomatic Complexity* (V(G))

V(G) = E – N + 2

V(G) = 13 – 11 + 2

V(G) = 2 + 2

V(G) = 4

Di mana :

E = jumlah *edge* pada grafik alir

N = jumlah *node* pada grafik alir

d. *Independent Path*

Path 1 = 1 – 4 – 5 – 6 – 7 – 14 – 15

Path 2 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 15

Path 3 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 14 – 15

Path 4 = 1 – 4 – 5 – 6 – 7 – 14 – 13 – 11 – 12– 10 –15

e. *Graph Matrices*

Berikut ini adalah *graph matrices* dari algoritma Djikstra yang dapat di lihat pada table 4.2.

Tabel 4.8 Graph Matrices Djikstra

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**N(E)-1**
1		1		1												2 – 1 = 1
2			1													1– 1 = 0
3				1												1 – 1 = 0
4					1											1 – 1 = 0
5						1										1 – 1 = 0
6							1									1 – 1 = 0
7								1						1		2 – 1 = 1
8									1							1 – 1 = 0

9										1						1 – 1 = 0
10															1	1 – 1 = 0
11												1				1 – 1 = 0
12										1						1 – 1 = 0
13												1				1 – 1 = 0
14													1		1	2 – 1 = 1
15																0 – 1 = 0
Sum(E) + 1																3 + 1 = 4

V(G) = Jumlah Graph Matric + 1

V(G) = 3 + 1

V(G) = 4

Berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 4, maka dapat disimpulkan bahwa pada proses pencarian dengan algoritma Djikstra berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.3 Hasil Pengujian

Berdasarkan rencana pengujian aplikasi yang telah disusun, maka dapat dilakukan beberapa tahap pengujian sebagai berikut :

4.2.3.1 Hasil Pengujian White Box

1. Algoritma Dijkstra

Berikut ini adalah hasil dari pengujian *white box* penelusuran algoritma Djikstra pada objek musuh yang terdapat pada game Destroyer Garuda. Dapat dilihat pada table 4.9.

Tabel 4.9 Hasil pengujian white box

Path no.	***Node (n)***	***Expected result***	***Test Result***	***Conclusion***
1	1 – 4 – 5 – 6 – 7 – 14 – 15	Menampilkan rute ke *player*	Menampilkan rute ke *player*	*Path* dikerjakan

2	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 15	Menampilkan rute ke *player*	Menampilkan rute ke *player*	*Path* dikerjakan
3	1 – 2 – 3 – 4 – 5 – 6 – 7 – 14– 15	Menampilkan rute ke *player*	Menampilkan rute ke *player*	*Path* dikerjakan
4	1 – 4 – 5 – 6 – 7 – 14 – 13 – 11 – 12– 10 –15	Menampilkan rute ke *player*	Menampilkan rute ke *player*	*Path* dikerjakan

4.2.4 Pengujian *Beta*

Pengujian *beta* merupakan pengujian yang di lakukan secara objektif, dimana pengujian dilakukan dengan mencoba aplikasi langsung ke kalangan yang memang menjadi target user atau pengguna apliksi *game* ini. Pengujian di lakukan dengan cara membagiakan kuesioner pada penguna guna mendapatkan data tentang kepuasan pengguna terhadap *game* Destroyer Garuda 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 di lakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian kepuasan terhadap aplikasi *game* Destroyer Garuda . kuesioner terdiri dari 5 peryataan dengan mengunakan skala 1 sampai 4 (skala likert).

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

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

Keterangan :

P = Nilai presentase yang dicari

S = Jumlah frekuensi dikalikan dengan skor yang ditetapkan tiap jawaban

skorideal = Skor tertinggi dikalikan dengan jumlah sample

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

1. Apakah perubahan mode pesawat ke mode robot dalam aplikasi *game Destroyer Garuda* menarik?
   a. Sangat Setuju b. Setuju c. Kurang Setuju d. Tidak Setuju

2. Apakah musuh dalam aplikasi *game Destroyer Garuda* lebih pintar?
   a. Sangat Setuju b. Setuju c. Kurang Setuju d. Tidak Setuju

3. Apakah anda kesulitan dalam menyelesaikan misi dalam aplikasi *game Destroyer Garuda*?
   a. Sangat Setuju b. Setuju c. Kurang Setuju d. Tidak Setuju

4. Apakah kontrol dalam aplikasi *game Destroyer Garuda* mudah dikendalikan?
   a. Sangat Setuju b. Setuju c. Kurang Setuju d. Tidak Setuju

5. Apakah secara umum menurut anda aplikasi *game Destroyer Garuda* sudah menarik?
   a. Sangat Setuju b. Setuju c. Kurang Setuju d. Tidak Setuju

4.6.3.1 Hasil Pengujian Beta adalah sebagai berikut:

1. Apakah perubahan mode pesawat ke mode robot dalam aplikasi *game Destroyer Garuda* menarik?

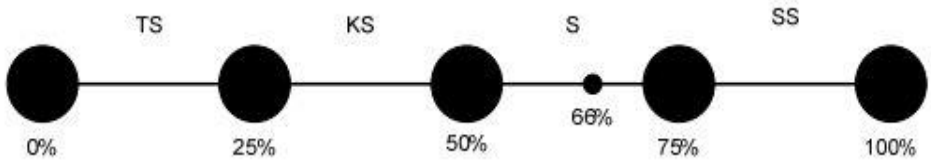
   Hasil kuesioner untuk pertanyaan ke-1 dapat di lihat pada table 4.10.

Tabel 4.10 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* Destroyer Garuda adalah 66% dari yang di harapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.9 :



Gambar 4.9 Interpretasi Skor pernyataan ke-1

2. Apakah musuh dalam aplikasi *game Destroyer Garuda* lebih pintar?

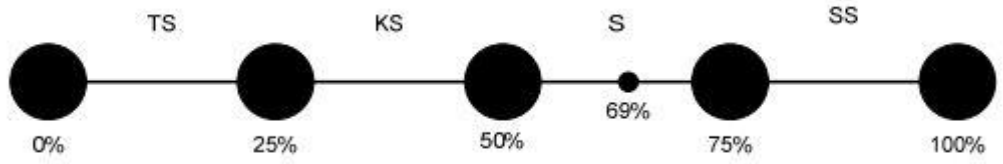
   Hasil kuesioner untuk pertanyaan ke-2 dapat di lihat pada table 4.11.

Tabel 4.11 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 Destroyer Garuda* adalah 69% dari yang di harapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.10 :



Gambar 4.10 Interpretasi skor pernyataan ke-2

3. Apakah anda kesulitan dalam menyelesaikan misi dalam aplikasi *game Destroyer Garuda*?

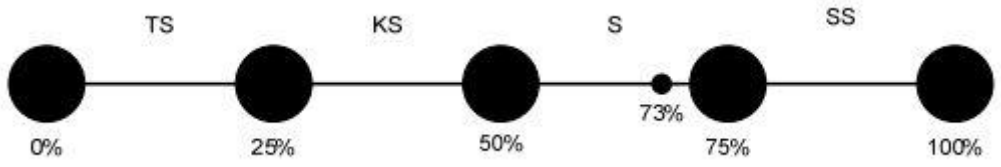
Hasil kuesioner untuk pertanyaan ke-3 dapat di lihat pada table 4.12.

Tabel 4.12 Perhitungan persentae 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 Destroyer Garuda* adalah 73% dari yang di harapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.11:



Gambar 4. 11 Interpretasi skor pernyataan ke-3

4. Apakah kontrol dalam aplikasi *game Destroyer Garuda* mudah dikendalikan?

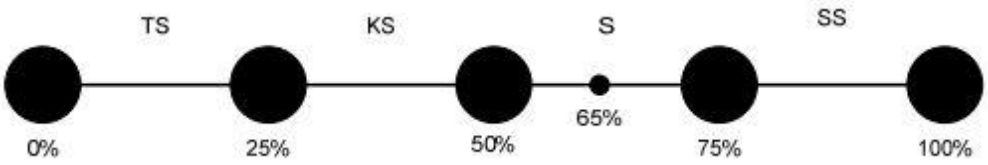
   Hasil kuesioner untuk pertanyaan ke-4 dapat di lihat pada table 4.13.

Tabel 4.13 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 Destroyer Garuda* adalah 65% dari yang di harapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.12 :



Gambar 4.12 Interpretasi skor pernyataan ke-4

5. Apakah secara umum menurut anda aplikasi *game Destroyer Garuda* sudah menarik?

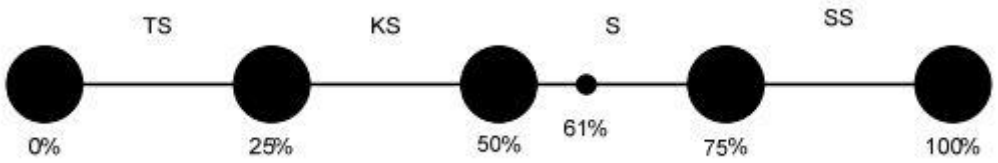
   Hasil kuesioner untuk pertanyaan ke-5 dapat di lihat pada table 4.14.

Tabel 4.14 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 Destroyer Garuda* adalah 61% dari yang di harapkan 100%, atau bisa dikategorikan sebagai setuju. Berikut kriteria interpretasi skor yang dapat dilihat pada gambar 4.13 :



Gambar 4.13 Interpretasi skor pernyataan ke-5

4.2.5 Kesimpulan Pengujuan Beta

Berdasakan pengujian beta yang telah didapat dari hasil kuesioner yang telah dijawab oleh para responden, maka bisa ditarik kesimpulan bahwa *game* Distroyer Garuda dalam perubahan mode dari mode pesawat ke mode robot membuat *game* ini menarik untuk di mainkan dan memiliki musuh yang cukup pintar sehinga sulit untuk menyelesaikan misi ditiap levelnya serta kontrol pada *game* yang mudah dikendalikan sehingga secara keseluruhan dapat di simpulkan bahwa *game* Destroyer Garuda menarik untuk di mainkan.