BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi yang dibangun yaitu *Game Tower Defense* Selamatkan Hutan Indonesia Berbasis *Desktop*. 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. Analisis sistem ini merupakan cakupan dari analisis yang berhubungan dengan *game* yangakan dibangun diantaranya analisis masalah terhadap *game* yang sejenis, analisis kebutuhan fungsional, dan analisis non fungsional.

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 2.13 GHz.
2. RAM 4GB.
3. Hard Disk 320 GB
4. Mouse, Keyboard, Sound Card, dan LCD.

Sedangkan spesifikasi perangkat lunak yang digunakan selama pengujian aplikasi ini mengguakan Sistem operasi Microsoft Windows 8.

4.1.2 Implementasi Instalasi *Game*

Sebelum dapat memainkan aplikasi *game Tower defense* Selamatkan Hutan Indonesia , *Game* harus tersimpan di direktori komputer. Petunjuk dalam instalasi *game* ini adalah sebagai berikut :

1. *Copy file* aplikasi *game Tower defense* Selamatkan Hutan Indonesia.*jar* pada direktori komputer.
2. Buka dan klik pada *file* tersebut.

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

4.1.3 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.

1. Menu Utama

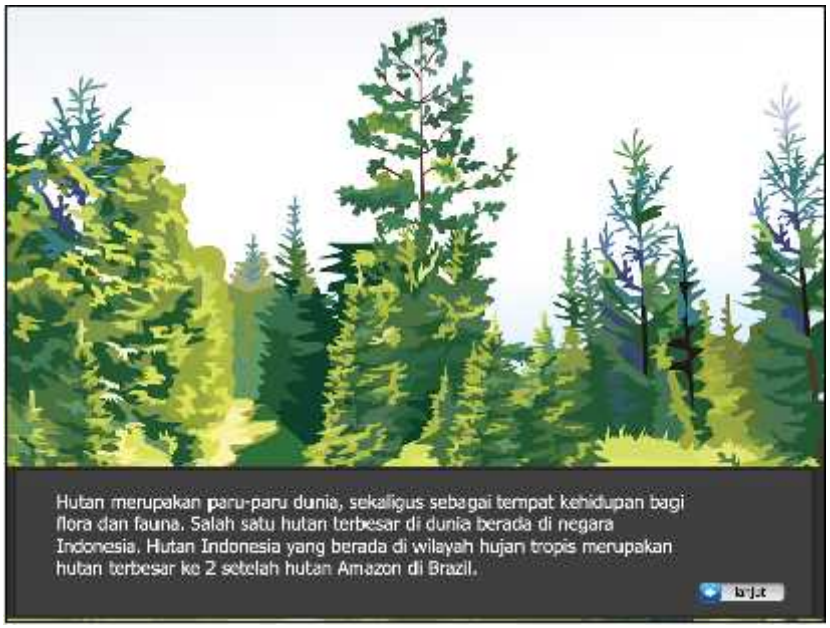
   Saat pengguna membuka aplikasi pertama kali, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka menu utama. Antar muka menu utama pada *game Tower defense* Selamatkan Hutan Indonesia adalah sebagai berikut.



Gambar 4. 1 Tampilan Menu Utama

Gambar 4.1 merupakan menu utama pada *game Tower defense* Selamatkan Hutan Indonesia. Terdapat 3 sub menu di dalamnya, yaitu menu main yang berfungsi untuk memulai permainan, menu Bantuan yang berfungsi untuk melihat petunjuk bermain dan menu Credit yang berfungsi latar belakang pembuatan *game*, dan menu.

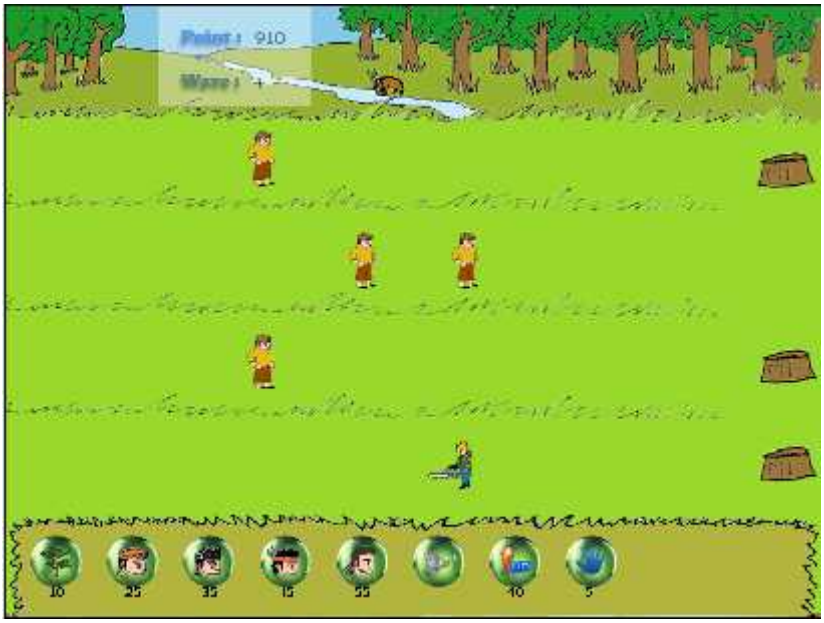
2. Intro



Gambar 4. 2 Tampilan *Intro*

Ketika pemain menekan tombol mulai baru, maka pemain akan masuk ketampilan intro yang menceritakan tentang keadaan Hutan Indonesia saat ini dan mengenai pembalakan yang sering terjadi di Indonesia.

3. *Level* 1



Gambar 4. 3 Tampilan *Level* 1

Ketika Pemain Menekan tombol Selanjutnya Pada menu intro maka System akan menampilkan *Gameplay Level* 1. Pada *Level* ini terdapat Tombol pilihan Tanaman, *tower*, siram, *upgrade* sama pindah *tower*. Tugas pemain adalah mencegah musuh masuk dan merusak area hutan.

4. *Level* 2



Gambar 4. 4 Tampilan *Level* 2

Setelah Pemain menyelesaikan *Level* 1 maka System akan menampilkan *Gameplay Level* 2. Pada *Level* ini terdapat Tombol pilihan Tanaman, *tower*, siram, *upgrade* sama pindah *tower*. Tugas pemain adalah mencegah musuh masuk dan merusak area hutan.

5. *Level* 3



Gambar 4. 5 Tampilan *Level* 3

Setelah Pemain menyelesaikan *Level* 2 maka System akan menampilkan *Gameplay Level* 3. Pada *Level* ini terdapat Tombol pilihan Tanaman, *tower*, siram, *upgrade* sama pindah *tower*. Tugas pemain adalah mencegah musuh masuk dan merusak area hutan.

6. Tampilan Bantuan



Gambar 4. 6 Tampilan Bantuan

Layar ini berfungsi menampilkan tampilan cara bermain. Tugas pemain adalah memperhatikan langkah-langkah dalam menu cara bermain ini.

7. Tampilan Credit



Gambar 4. 7 Tampilan Credit

Layar ini berfungsi menampilkan tampilan *credit*, terdapat informasi mengenai Aplikasi *game Tower Defense* Selamatkan Hutan Indonesia dan pembuatnya.

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 Tower defense* Selamatkan Hutan Indonesia. Skenario pengujian yang akan dilakukan pada ini selengkapnya dapat dilihat pada table 4.1.

Tabel 4. 1 Skenario Pengujian

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1.	Menu Utama	Memilih menu Main	*Black box*
		Memilih menu *Credit*	*Black box*
		Memilih Menu Bantuan	*Black box*
	Main	Menampilkan Intro	*Black box*
2.	Intro	Memilih Tombol Selanjutnya dan Memulai *Gameplay*	*Black box*
3.	*GamePlay*	Pilih Tanaman	*Black box*
		Pilih *Tower*	*Black box*
		Uilih siram	*Black box*
		Pilih *Upgrade*	*Black box*
		Pilih Pindah *Tower*	*Black box*
		Tampil sprites Tanaman	*Black box*
		Tampil sprites Tower	*Black box*
		Tampil sprites Musuh	*Black box*
		Tampil serangan Tower	*Black box*
		Musuh menyerang Tower	*Black box*
		Poin Bertambah	*Black box*
		Poin Berkurang	*Black box*
	Credit	Pilih Tombol Kembali	*Black box*
	Bantuan	Pilih Tombol Kembali	*Black box*
4.	Pemilihan Jalur Musuh	Penelusuran dengan Algoritma Fuzzy	*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	Memilih Tombol Main	Sistem menampilkan Halaman Intro	[√] Berhasil [ ] Tidak Berhasil
		Memilih Tombol Credit	Sistem menampilkan halaman menu *Credit*	[√] Berhasil [ ] Tidak Berhasil
		Memilih Tombol Bantuan	Sistem menampilkan halaman menu Bantuan	[√] Berhasil [ ] Tidak Berhasil
2.	Intro	Memilih Tombol Selanjutnya	Sistem Menampilkan *Gameplay*	[√] Berhasil [ ] Tidak
3.	*Gameplay*	Pilih Tanaman	Sistem dapat memilih tanaman dan menempatkannya di arena	[√] Berhasil [ ] Tidak Berhasil
		Pilih *Tower*	Sistem dapat memilih tanaman dan menempatkannya di arena	[√] Berhasil [ ] Tidak Berhasil
		Pilih Siram	Sistem dapat memperbesar tanaman	[√] Berhasil [ ] Tidak Berhasil
		Pilih *Upgrade*	Sistem Meningkatkan Jangkauan Tembakan *Tower*	[√] Berhasil [ ] Tidak Berhasil

		Pilih Pindah *Tower*	Sistem Memindahkan *Tower* di arena Permainan	[√] Berhasil [ ] Tidak Berhasil
		Tampil *sprites* Tanaman	Sistem dapat menampilkan *Sprites* Tanaman di arena	[√] Berhasil [ ] Tidak Berhasil
		Tampil *sprites Tower*	Sistem dapat menampilkan *Sprites* Tower di arena	[√] Berhasil [ ] Tidak Berhasil
		Tampil *sprites* Musuh	Sistem dapat menampilkan *Sprites* Musuh di arena	[√] Berhasil [ ] Tidak Berhasil
		Tampil serangan *tower*	Sistem dapat menampilkan Serangan *Tower* Saat Musuh mendekat	[√] Berhasil [ ] Tidak Berhasil
		Musuh menyerang *Tower*	Sistem dapat menampilkan Serangan Musuh Saat mendekati *Tower*	[√] Berhasil [ ] Tidak Berhasil
		Poin Bertambah	Sistem Menambah poin saat *tower* mengalahkan musuh dan saat terdapat taman di arena	[√] Berhasil [ ] Tidak Berhasil
		Poin Berkurang	Sistem mengurangi poin saat pemain memilih tanaman, *Tower*, *upgrade tower*, siram, dan Memindahkan *Tower*.	[√] Berhasil [ ] Tidak Berhasil
6.	*Credit*	Memilih menu kembali	Sistem Menampilkan Halaman Menu Utama	[√] Berhasil [ ] Tidak Berhasil
7.	Bantuan	Memilih Menu Kembali	Sistem Menampilkan Halaman Menu Utama	[√] Berhasil [ ] Tidak Berhasil

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

Pada pengujian *White Box* dilakukan proses pengujian pada proses jalur musuh yang merupakan proses pengujian dari pengimplementasian algoritma Fuzzy, dan menguji apakah algoritma Fuzzy berjalan dengan seharusnya pada musuh.

a. Algoritma Fuzzy pada musuh

```
1.  int total1 = 0;
    int total2 = 0;
    int total3 = 0;
    int total4 = 0;
    int jarak1 = 0;
    int jarak2 = 0;
    int jarak3 = 0;
    int jarak4 = 0;
    int kekuatan1 = 0;
    int kekuatan2 = 0;
    int kekuatan3 = 0;
    int kekuatan4 = 0;
    int all1= -1;
    int all2= -1;
    int all3= -1;
    int all4= -1;
    int countVeryEasy = 0;
    int countEasy = 0;
    int countNormal = 0;
    int countHard = 0;
    int jalur[] = new int[4];
    int fuzzyE[] = new int[4];
    int k = 0;
    int lokasiA = 150;
    int lokasiB = 500;
2.  if(this.getX()<=625)
            {
3.  lokasiA = this.getY();
4.  if(lokasiA==150)
            {
            lokasiB = lokasiA+100;
            }
5.  else if(lokasiA==450)
            {
            lokasiB = lokasiA;
            lokasiA -= 100;
            }
            else
            {
            lokasiB = lokasiA+100;
            lokasiA -= 100;
            }
6.  }
            System.out.println("Y : "+lokasiA);
```

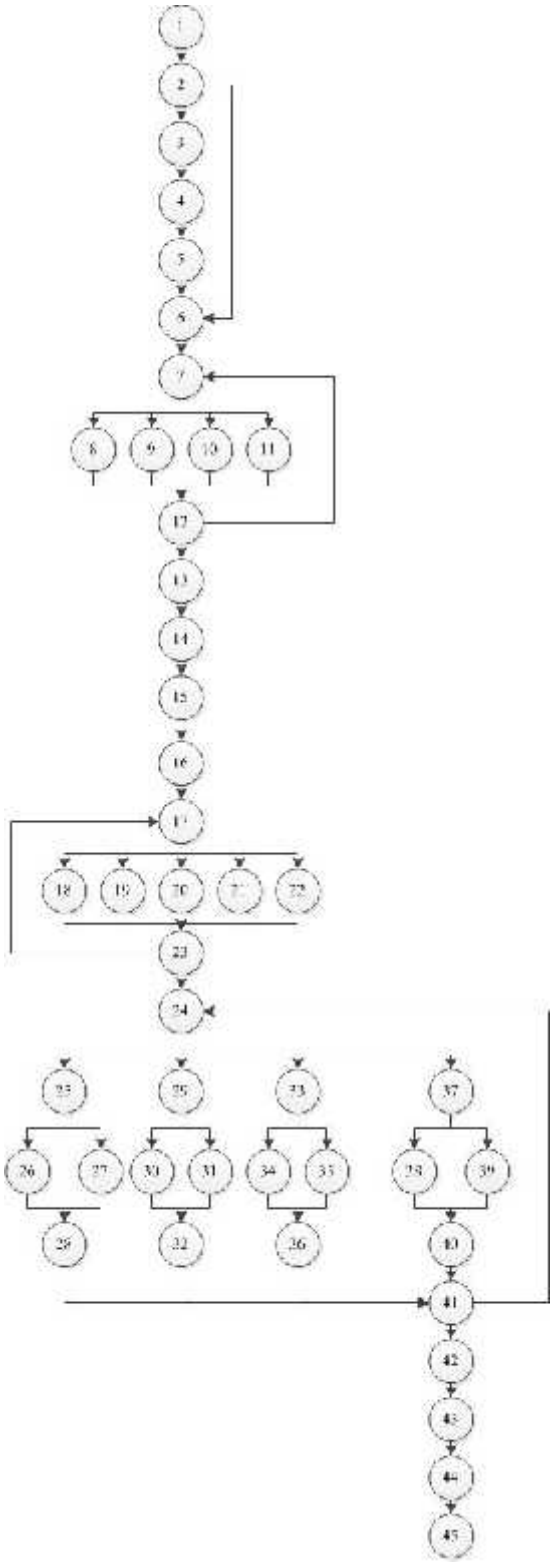
```
        System.out.println("X : "+this.getX());
7.  for(int j=lokasiA;j<=lokasiB;j+=100)
        {
8.  if(j==150)
        {
        total1 = jumlahFuzzy(j);
        jarak1 = jarakFuzzy(j);
        kekuatan1 = kekuatanFuzzy(j);
        all1 = total1+jarak1+kekuatan1;
        }
9.  else if(j==250)
        {
        total2 = jumlahFuzzy(j);
        jarak2 = jarakFuzzy(j);
        kekuatan2 = kekuatanFuzzy(j);
        all2 = total2+jarak2+kekuatan2;
            }
10. else if(j==350)
        {
        total3 = jumlahFuzzy(j);
        jarak3 = jarakFuzzy(j);
        kekuatan3 = kekuatanFuzzy(j);
        all3 = total3+jarak3+kekuatan3;
        }
11. else if(j==450)
        {
        total4 = jumlahFuzzy(j);
        jarak4 = jarakFuzzy(j);
        kekuatan4 = kekuatanFuzzy(j);
        all4 = total4+jarak4+kekuatan4;
        }
12. }
        System.out.println("all1 : "+all1);
        System.out.println("all2 : "+all2);
        System.out.println("all3 : "+all3);
        System.out.println("all4 : "+all4);
13. if(all1==0)
        {
        jalur[0] = VERYEASY;
        countVeryEasy += 1;
        }
        else if(all1>=1 && all1<=4)
        {
        jalur[0] = EASY;
        countEasy += 1;
        }
        else if(all1>4 && all1<7)
        {
        jalur[0] = NORMAL;
        countNormal += 1;
        }
        else if(all1>=7)
        {
        jalur[0] = HARD;
        countHard += 1;
        }
```

```
        else if(all1==-1)
        {
        jalur[0] = -1;
        }
14. if(all2==0)
        {
        jalur[1] = VERYEASY;
        countVeryEasy += 1;
        }
        else if(all2>=1 && all2<=4)
        {
        jalur[1] = EASY;
        countEasy += 1;
        }
        else if(all2>4 && all2<7)
        {
        jalur[1] = NORMAL;
        countNormal += 1;
        }
        else if(all2>=7)
        {
        jalur[1] = HARD;
        countHard += 1;
        }
        else if(all2==-1)
        {
        jalur[1] = -1;
        }
15. if(all3==0)
        {
        jalur[2] = VERYEASY;
        countVeryEasy += 1;
        }
        else if(all3>=1 && all3<=4)
        {
        jalur[2] = EASY;
        countEasy += 1;
        }
        else if(all3>4 && all3<7)
        {
        jalur[2] = NORMAL;
        countNormal += 1;
        }
        else if(all3>=7)
        {
        jalur[2] = HARD;
        countHard += 1;
        }
        else if(all3==-1)
        {
        jalur[2] = -1;
        }
16. if(all4==0)
        {
        jalur[3] = VERYEASY;
        countVeryEasy += 1;
```

```
        }
        else if(all4>=1 && all4<=4)
        {
        jalur[3] = EASY;
        countEasy += 1;
        }
        else if(all4>4 && all3<7)
        {
        jalur[3] = NORMAL;
        countNormal += 1;
        }
        else if(all4>=7)
        {
        jalur[3] = HARD;
        countHard += 1;
        }
        else if(all4==-1)
        {
        jalur[3] = -1;
              }
17. for(int i=0;i<4;i++)
        {
18. if(jalur[i]==-1)
        {
        System.out.println("Jalur"+(i+1)+" : NULL");
        }
19. else if(jalur[i]==0)
        {
        System.out.println("Jalur"+(i+1)+" : KOSONG");
        }
20. else if(jalur[i]==1)
        {
        System.out.println("Jalur"+(i+1)+" : AMAN");
        }
21. else if(jalur[i]==2)
        {
        System.out.println("Jalur"+(i+1)+" : NORMAL");
           }
22. else if(jalur[i]==3)
        {
        System.out.println("Jalur"+(i+1)+" : BERBAHAYA");
        }
23. }
24. for(int i=0;i<4;i++)
        {
25. if(countVeryEasy>=1)
        {
26. if(jalur[i]==VERYEASY)
        {
        fuzzyE[k] = (i+1)*100+50;
        k++;
        }
27. else if(jalur[i]!=VERYEASY)
        {
        }
28. }
```

```
29.  else if(countEasy>=1 && countVeryEasy==0)
                {
30.  if(jalur[i]==EASY)
                {
                fuzzyE[k] = (i+1)*100+50;
                k++;
                }
31.  else if(jalur[i]!=EASY)
                {
                }
32   }
33.  else if(countNormal>=1 && countEasy==0)
                {
34.  if(jalur[i]==NORMAL)
                {
                fuzzyE[k] = (i+1)*100+50;
                k++;
                }
35.  else if(jalur[i]!=NORMAL)
                {
                }
36.  }
37.  else if(countHard==4)
                {
38.  if(jalur[i]==HARD)
                {
                fuzzyE[k] = (i+1)*100+50;
                k++;
                }
39.  else if(jalur[i]!=HARD)
                {
                }
40.  }
41.  }
42.  k-=1;
43.  if(k!=0)
                {
                k=Greenfoot.getRandomNumber(k);
                }
44.  this.setLocation(this.getX(), fuzzyE[k]);
45.  }
```

b. *Flowgraph*



Gambar 4. 8 *Flowgraph* Jalur Musuh (*Fuzzy*)

c. *Cyclomatic Complexity* V(G)

V(G) = E – N + 2

Di mana :

E adalah edge = Jumlah Aksi, dan

N adalah node = Jumlah Kondisi. Jadi :

V(G) = 62 edge – 45 node + 2 = 19

d. *Independent Path*

Dari hasil perhitungan *cyclomatic complexity* terdapat 19 independent path, yaitu :

Path 1 = 1-2-3-4-5-6-7

Path 2 = 1-2-6-7-8-12

Path 3 = 1-2-6-7-9-12

Path 4 = 1-2-6-7-10-12

Path 5 = 1-2-6-7-11-12

Path 6 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-23

Path 7 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-19-23

Path 8 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-20-23

Path 9 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-21-23

Path 10 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-22-23

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

Path 12 = 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- 28

Path 13 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-27-28

Path 14 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-29-30-32

Path 15 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-29-31-32

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

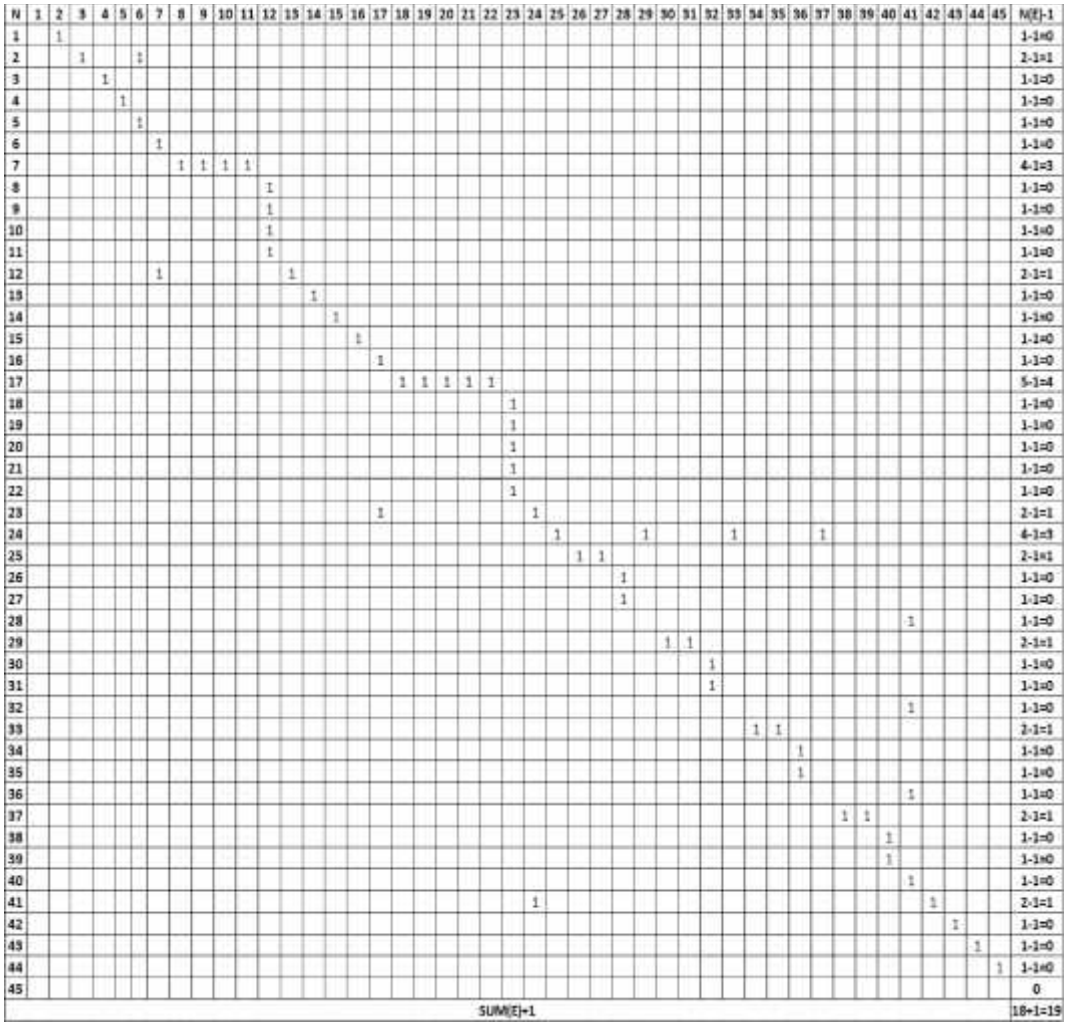
33-34-36

Path 17 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-33-35-36

Path 18 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-37-38-40-41-42-43-44-45

Path 19 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-37-39-40-41-42-43-44-45

Graph Matriks Jalur Musuh (Fuzzy)



Gambar 4. 9 Tabel *Graph* Matriks Jalur Musuh (*Fuzzy*)

Berikit ini adalah hasil dari pengujian *whitebox* penelusuran algoritma Fuzzy jalur musuh pada aplikasi *game Tower Defense* Selamatkan Hutan Indonesia.

Tabel 4. 3 Hasil Pengujian *White Box*

NO	PATH	YANG DIHARAPKAN	HASIL YANG DI CAPAI	KESIMPULAN
1	1-2-3-4-5-6-7	Inisialisasi lokasi Musuh dari ke empat jalur ketika berjalan	Sebelum musuh keluar, musuh meninisialisasi lokasi dari ke empat jalur	[√] Diterima [ ] Ditolak
2	1-2-6-7-8-12	Musuh mendapatkan lokasi dari ke empat jalur ketika berjalan	Setelah musuh keluar, musuh dapat lokasi dari ke empat jalur	[√] Diterima [ ] Ditolak
3	1-2-6-7-9-12	Musuh mendapatkan lokasi dari ke empat jalur ketika berjalan	Setelah musuh keluar, musuh dapat lokasi dari ke empat jalur	[√] Diterima [ ] Ditolak
4	1-2-6-7-10-12	Musuh mendapatkan lokasi dari ke empat jalur ketika berjalan	Setelah musuh keluar, musuh dapat lokasi dari ke empat jalur	[√] Diterima [ ] Ditolak
5	1-2-6-7-11-12	Musuh mendapatkan lokasi dari ke empat jalur ketika berjalan	Setelah musuh keluar, musuh dapat lokasi dari ke empat jalur *tower*	[√] Diterima [ ] Ditolak
6	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-23	Musuh mendapatkan lokasi dari ke empat jalur ketika berjalan	Setelah musuh keluar, musuh dapat lokasi dari ke empat jalur	[√] Diterima [ ] Ditolak
7	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-19-23	Musuh dapat mengecek ke empat jalur berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	Setelah musuh keluar, musuh mengecek ke empat *tower* berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	[√] Diterima [ ] Ditolak
8	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-20-23	Musuh dapat mengecek ke empat jalur berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	Setelah musuh keluar, musuh mengecek ke empat *tower* berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	[√] Diterima [ ] Ditolak
9	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-21-23	Musuh dapat mengecek ke empat jalur berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	Setelah musuh keluar, musuh mengecek ke empat *tower* berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	[√] Diterima [ ] Ditolak
10	1-2-3-4-5-6-7-8-9-10-	Musuh dapat mengecek ke empat	Setelah musuh keluar, musuh	[√] Diterima [ ] Ditolak

	11-12-13-14-15-16-17-22-23	jalur berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	mengecek ke empat *tower* berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	
11	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-2443-44	Musuh dapat mengecek ke empat jalur berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	Setelah musuh keluar, musuh mengecek ke empat *tower* berdasarkan jumlah *tower*, jarak *tower*, kuat *tower*	[√] Diterima [ ] Ditolak
12	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- 28	Musuh dapat menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	Setelah musuh keluar. Musuh menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	[√] Diterima [ ] Ditolak
13	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-27-28	Musuh dapat menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	Setelah musuh keluar. Musuh menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	[√] Diterima [ ] Ditolak
14	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-29-30-32	Musuh dapat menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	Setelah musuh keluar. Musuh menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	[√] Diterima [ ] Ditolak
15	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-29-31-32	Musuh dapat menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	Setelah musuh keluar. Musuh menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	[√] Diterima [ ] Ditolak
16	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-33-34-36	Musuh dapat menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	Setelah musuh keluar. Musuh menentukan status defuzzyfikasi ke empat jalur tersebut berdasarkan aman, normal, berbahaya	[√] Diterima [ ] Ditolak
17	1-2-3-4-5-6-7-8-9-10-	Menampilkan musuh di lokasi berdasarkan	Setelah musuh keluar, musuh	[√] Diterima [ ] Ditolak

	11-12-13-14-15-16-17-18-19-20-21-22-23-24-33-35-36	defuzzyfikasi	mengatur lokasi defuzzyfikasi dan berpindah	
18	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-37-38-40-41-42-43-44-45	Menampilkan musuh di lokasi berdasarkan defuzzyfikasi	Setelah musuh keluar, musuh mengatur lokasi defuzzyfikasi dan berpindah	[√] Diterima [ ] Ditolak
19	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-37-39-40-41-42-43-44-45	Menampilkan musuh di lokasi berdasarkan defuzzyfikasi	Setelah musuh keluar, musuh mengatur lokasi defuzzyfikasi dan berpindah	[√] Diterima [ ] Ditolak

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 Kuisioner

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 kemudian dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian penerapan sistem yang telah dibangun. Kuesioner terdiri dari 6 pertanyaan 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.

skor ideal = Nilai tertinggi dikalikan dengan jumlah sampel.

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

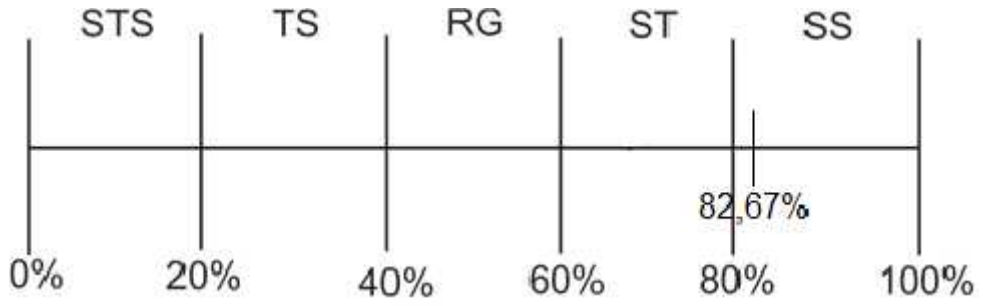
1. Apakah game ini membantu anda mengetahui kondisi Hutan Indonesia pada saat ini?

Tabel 4. 4 Perhitungan persentase pertanyaan ke-1

Pertanyaan	Keterangan	Skor	Frekuensi	S
1	Sangat Setuju	5	10	50
	Setuju	4	14	56
	Ragu-ragu	3	6	18
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**124**

$$P = \frac{124}{150} \times 100\% = 82{,}67\%$$

Dapat disimpulkan bahwa penilaian terhadap pengetahuan pemain mengenai kondisi hutan Indonesia setelah memainkan *game Tower defense* selamatkan Hutan Indonesia adalah 82,67% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.10.



Gambar 4. 10 Interpretasi Skor Pertanyaan ke-1

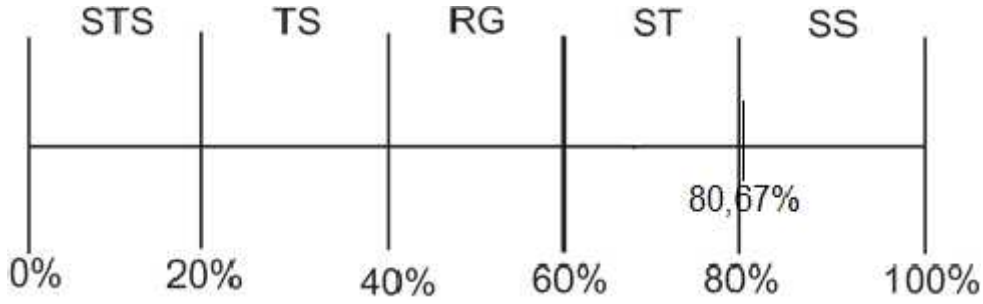
2. Apakah menurut anda *Game* ini mudah dimainkan?

Tabel 4. 5 Perhitungan persentase pertanyaan ke-2

Pertanyaan	Keterangan	Skor	Frekuensi	S
2	Sangat Setuju	5	6	30
	Setuju	4	19	76
	Ragu-ragu	3	5	15
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**121**

$$P = \frac{121}{150} \times 100\% = 80.67\%$$

Dapat disimpulkan bahwa penilaian terhadap mudah tidaknya dimainkan pada *game Tower defense* selamatkan Hutan Indonesia adalah 80,67% 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-2

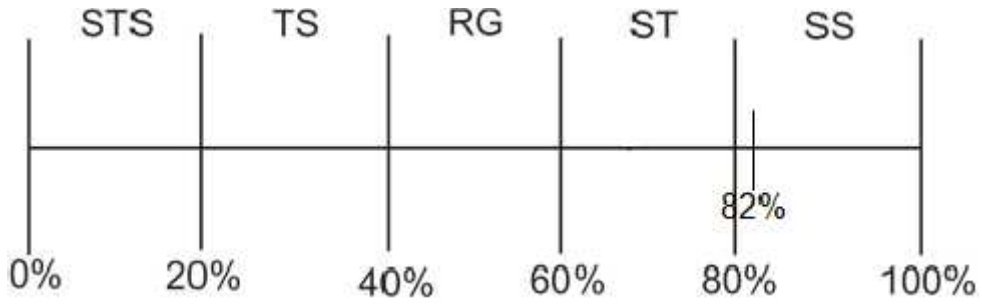
3. Apakah tema selamatkan Hutan Indonesia pada *game* ini sudah baik?

Tabel 4. 6 Perhitungan persentase pertanyaan ke-3

Pertanyaan	Keterangan	Skor	Frekuensi	S
3	Sangat Setuju	5	8	40
	Setuju	4	17	68
	Ragu-ragu	3	5	15
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**123**

$$P = \frac{123}{150} \times 100\% = 82\%$$

Dapat disimpulkan bahwa penilaian terhadap kesesuaian tema pada *game Tower defense* selamatkan Hutan Indonesia adalah 82% 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-3

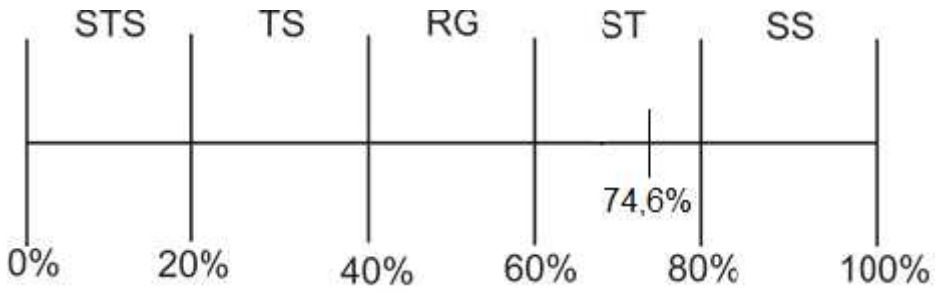
4. Dengan Musuh yang cerdas apakah tingkat kesulitan pada *game* ini cukup menantang?

Tabel 4. 7 Perhitungan persentase pertanyaan ke-4

Pertanyaan	Keterangan	Skor	Frekuensi	S
4	Sangat Setuju	5	1	5
	Setuju	4	20	80
	Ragu-ragu	3	9	27
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**112**

$$P = \frac{112}{150} \times 100\% = 74{,}6\%$$

Dapat disimpulkan bahwa penilaian terhadap tingkat kesulitan *game* dengan kecerdasan yang dimiliki musuh pada *game Tower defense* selamatkan Hutan Indonesia adalah 74,6% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.13.



Gambar 4. 13 Interpretasi Skor Pertanyaan ke-4

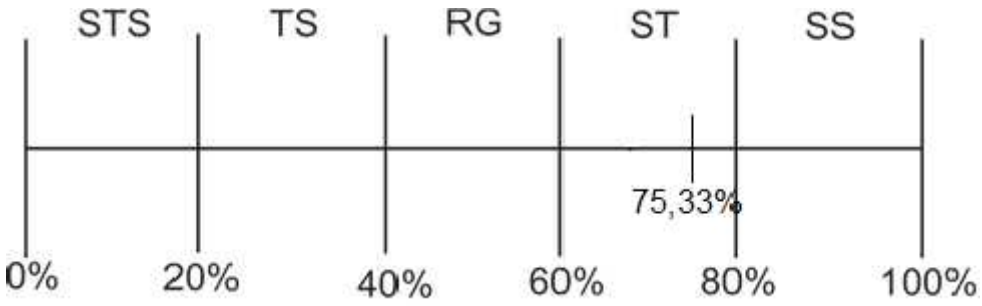
5. Apakah *gameplay* pada *game* ini secara umum menarik bagi anda?

Tabel 4. 8 Perhitungan persentase pertanyaan ke-5

Pertanyaan	Keterangan	Skor	Frekuensi	S
5	Sangat Setuju	5	3	15
	Setuju	4	17	68
	Ragu-ragu	3	10	30
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**113**

$$P = \frac{113}{150} \times 100\% = 75.33\%$$

Dapat disimpulkan bahwa penilaian terhadap *gameplay* pada pada *game Tower defense* selamatkan Hutan Indonesia adalah 75,33% 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-5

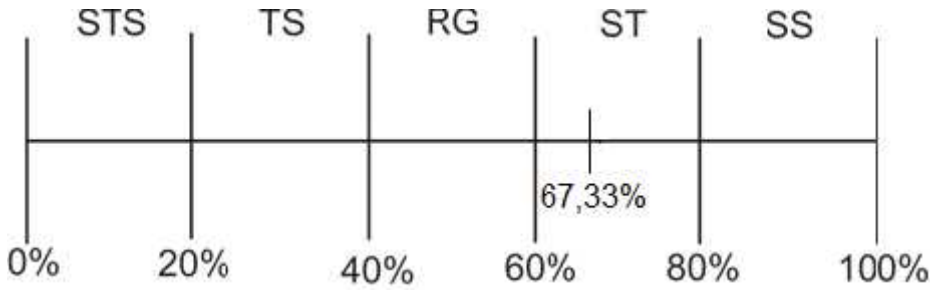
6. Apakah menurut anda tampilan antarmuka *game* ini menarik?

Tabel 4. 9 Perhitungan persentase pertanyaan ke-6

Pertanyaan	Keterangan	Skor	Frekuensi	S
6	Sangat Setuju	5	0	0
	Setuju	4	15	60
	Ragu-ragu	3	11	33
	Tidak Setuju	2	4	8
	Sangat Tidak Setuju	1	0	0
Jumlah			**30**	**101**

$$P = \frac{101}{150} \times 100\% = 67{,}33\%$$

Dapat disimpulkan bahwa penilaian terhadap tampilan antar muka game pada *game Tower defense* selamatkan Hutan Indonesia adalah 67,33% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.15.



Gambar 4. 15 Interpretasi Skor Pertanyaan ke-6

4.2.2.2 Kesimpulan hasil kuisioner

Hasil dari pengujian beta (kuisioner) dapat disimpulkan bahwa :

1. Pemain merasa lebih mengetahui kondisi hutan Indonesia pada saat ini setelah bermain game ini.
2. Permain merasa sangat mudah dalam memainkan *game* ini.
3. Kesesuaian tema selamatkan Hutan Indonesia dengan *game* dinilai sudah sangat baik oleh pemain.
4. Pemain merasa dengan kecerdasan yang dimiliki musuh, tingkat kesulitan *game* ini cukup menantang.
5. Pemain merasa *gameplay* pada *game* ini sudah menarik.
6. Pemain merasa tampilan *game* ini menarik dan sudah bagus.