BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Implementasi adalah penerapan cara kerja sistem berdasarkan hasil analisa dan juga perancangan yang telah dibuat sebelumnya ke dalam suatu bahasa pemrograman tertentu.

4.2 Perangkat Kebutuhan Aplikasi

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

4.2.1 Perangkat Keras (*Hardware*)

Berikut adalah spesifikasi perangkat keras yang digunakan untuk menguji *game* "Ternak 2D" ini, yaitu :

1. Processor : 1,8 Ghz ke atas.
2. Hardisk : 40 Gb.
3. Memory : 512 Mb.
4. VGA : 128 Mb.
5. Monitor : CRT/LCD 15 inch.
6. Mouse & Keyboard.

4.2.2 Perangkat Lunak (*Software*)

Untuk menjalankan *game* "Ternak 2D" ini ada beberapa tahap yang harus dilakukan, yaitu :

1. Download *Flash Player* versi 10.
2. Install *Flash Player* untuk menjalankan program.
3. Download *Adobe Flash CS4/CS5*.
4. Install *Adobe Flash* untuk membuka *source code* program.
5. Klik file Ternak 2D.swf untuk menjalankan *game*.

4.3 Identifikasi Kelas

Identifikasi kelas bisa dilihat pada tabel 4.1 :

Tabel 4.1 Identifikasi kelas

Nama Class	Nama File Fisik
Ternak2D	Ternak2D.as
Node	Node.as
INode	INode.as
Pathfinder	Pathfinder.as
Grid	Grid.as
Panel	Panel.as
Actor	Actor.as
ternak Layer	ternak Layer.as
Food layer	Food layer.as
Musuh	Musuh.as
Kandang	Kandang.as

4.4 Implementasi Antar Muka

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.

4.4.1 Tampilan Pilih Karakter

Pada tampilan pilih karakter ini terdapat dua pilihan yaitu karakter laki-laki dan karakter perempuan, setelah itu isi nama untuk membuat player baru. Tombol simpan untuk menyimpan karakter dan nama yang telah di isi, sedangkan tombol batal untuk membatalkan pengisian nama dan pemilihan karakter. Tampilan pilih karakter dapat di lihat pada Gambar 4.1:



Gambar 4.1 Tampilan pilih karakter

4.4.2 Tampilan Menu Utama *Game*

Pada tampilan menu utama *game* ini terdapat 3 menu utama yaitu Mulai Beternak untuk memulai permainan, Pengaturan untuk melihat pengaturan *game* diantaranya pengaturan suara, musik dan tampilan layar, dan Petunjuk untuk melihat apa yang harus dikerjakan. Menu Buat baru untuk membuat karakter baru, menu Bantuan untuk melihat panduan permainan, dan tombol untuk keluar dari aplikasi. Tampilan menu utama *game* dapat dilihat pada Gambar 4.2:



Gambar 4.2 Tampilan menu utama g*ame*

4.4.3 Tampilan Pengaturan

Pada tampilan Pengaturan ini terdapat 3 menu yaitu Musik untuk mengaktifkan dan menonaktifkan musik, Efek suara untuk mengaktifkan dan menonaktifkan efek suara, dan *Fullscreen* untuk membesar atau mengecilkan tampilan layar. Dan tombol kembali yaitu untuk kembali ke tampilan menu utama *game*. Tampilan Pengaturan dapat dilihat pada Gambar 4.3 :



Gambar 4.3 Tampilan Pengaturan

4.4.4 Tampilan Petunjuk

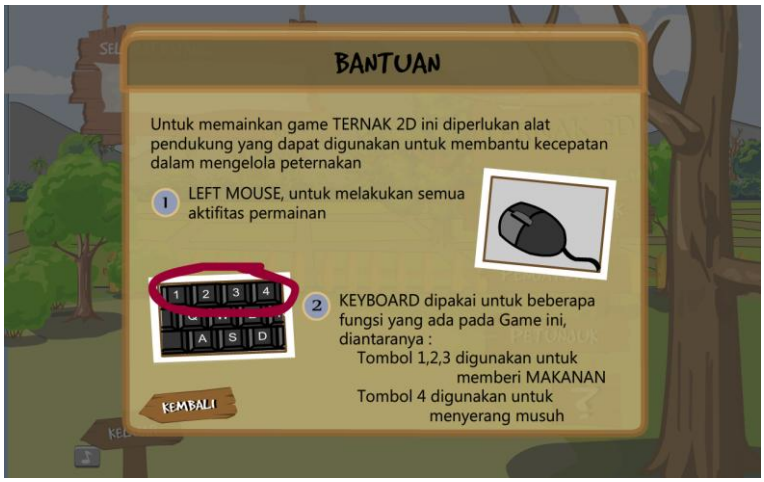
Pada tampilan Petunjuk berisi mengenai Apa saja yang harus dilakukan. Tampilan Petunjuk dapat dilihat pada gambar 4.4 :



Gambar 4.4 Tampilan Petunjuk

4.4.5 Tampilan Bantuan

Pada tampilan Bantuan berisi mengenai panduan bermain *game*. Tampilan Bantuan dapat dilihat pada gambar 4.5 :



Gambar 4.5 Tampilan Bantuan

4.4.6 Tampilan Permainan

Pada tampilan Permainan berisi mengenai tampilan permainan dari level 1 sampai level 4, didalamnya terdapat Toko untuk pembelian, fungsi-fungsi untuk peemberian makanan, dan juga fungsi untuk menhyerang musuh. menu main menu yaitu untuk kembali ke tampilan menu utama *game*. Tampilan Permainan dapat dilihat pada gambar 4.6:



Gambar 4.6 Tampilan Permainan

4.4.7 Tampilan Menu Toko

Pada tampilan Menu Toko berisi mengenai daftar pembelian. Tampilan Menu Toko dapat dilihat pada Gambar 4.7:



Gambar 4.7 Tampilan Menu Toko

4.4.8 Tampilan Pesan Naik Level

Pada tampilan pesan Naik level berisi mengenai pesan ketika sudah mencapai pada poin tertentu. Tampilan Pesan Naik Level dapat dilihat pada Gambar 4.8:



Gambar 4.8 Tampilan Pesan Naik Level

4.4.9 Tampilan Pesan Menang

Pada tampilan Pesan Menang berisi mengenai pesan selesainya permainan. Tampilan pesan Menang dapat dilihat pada Gambar 4.9:



Gambar 4.9 Tampilan Pesan Menang

4.4.10 Tampilan Pesan Kalah

Pada tampilan pesan Kalah berisi mengenai pesan permainan kalah. Tampilan Pesan Kalah dapat dilihat pada Gambar 4.10:



Gambar 4.10 Tampilan Pesan Kalah

4.4.11 Tampilan Pesan Keluar ke Menu Utama

Pada tampilan pesan Keluar ke menu utama berisi mengenai pesan jika ingin keluar dari permainan dengan menyimpan permainan sebelumnya. Tampilan Pesan Keluar ke menu utama dapat dilihat pada Gambar 4.11:



Gambar 4.11 Tampilan Pesan Keluar ke Menu Utama

4.4.12 Tampilan Pesan Hapus Karakter

Pada tampilan pesan hapus karekter berisi mengenai pesan jika ingin menghapus karakter yang telah disimpan. Tampilan Pesan hapus karakter dapat dilihat pada Gambar 4.11:



Gambar 4.11 Tampilan Pesan Hapus Karakter

4.5 Pengujian Sistem

Pengujian sistem dilakukan untuk menemukan kesalahan pada perangkat lunak yang akan diuji. Pengujian dimaksudkan untuk mengetahui apakah perangkat lunak yang telah dibuat sudah memenuhi kriteria yang sesuai dengan tujuan ataukah belum.

4.6 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.

4.6.1 Rencana Pengujian

4.6.1.1 *Black Box*

Black box ini bertujuan untuk menunjukkan fungsi perangkat lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan. Rencana pengujian yang akan dilakukan pada aplikasi ini selengkapnya bisa di lihat pada tabel 4.2 :

Tabel 4.2 Rencana Pengujian Aplikasi

Kelas Uji	Detail Pengujian	Jenis Uji
Ternak2D	Menampilkan semua tampilan pada permainan, dari mulai pilih karakter, pilihan menu utama, dan permainan, termasuk isi dari menu toko.	*Black box*
Node	Menggunakan *node* atau *grid* untuk menentukan jarak pada permainan.	*Black box*
INode	Memanggil fungsi yang ada di kelas Node.as.	*Black box*
Pathfinder	Menampilkan fungsi algoritma A* (*A star*) *pathfinding* dengan *euclidean distance* pada permainan.	*White box*
Grid	Menentukan nilai *grid* dan membedakan *grid* untuk hewan darat dan hewan air.	*Black box*
Panel	Menampilkan fungsi pada permainan, diantaranya fungsi layar, *pause* dan fungsi	*Black box*

	penggunaan tombol pada *keyboard*.	
Actor	Menampilkan tampilan *actor* untuk menjalankan aktifitas peternakan yaitu memberi makan ternak	*Black box*
ternak Layer	Menampilkan tampilan hewan ternak pada permainan dengan menerapkan beberapa fungsi yang ada pada hewan ternak tersebut. Diantaranya harga ternak, harga jual, pergerakan hewan ternak termasuk penerapan algoritma A* (*A star*) *pathfinding* dengan *euclidean distance*, *collision* pada ternak, waktu lapar, waktu produksi, dan waktu mati.	*Black box*
Food layer	Menampilkan tampilan makanan hewan ternak pada permainan dengan menerapkan banyaknya makanan, waktu habis makanan, pembedaan jenis makanan, dan berkurangnya makanan.	*Black box*
Musuh	Menampilkan tampilan musuh pada permainan dengan menerapkan beberapa fungsi pada musuh. Diantaranya jumlah munculnya musuh pada setiap level, pergerakan musuh dengan menerapkan	*Black box*

	algoritma A* (*A star*) *pathfinding euclidean distance* pada pencarian hewan ternak terdekatnya.	
Kandang	Menampilkan tampilan kandang pada permainan dengan menerapkan kapasitas kandang untuk tempat persembunyian hewan ternak terhadap musuh.	*Black box*

4.6.1.2 *White Box*

White Box adalah meramalkan cara kerja perangkat lunak secara rinci, karenanya *logical path* (jalur logika) perangkat lunak akan ditest dengan menyediakan *test case* yang akan mengerjakan kumpulan kondisi dan atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

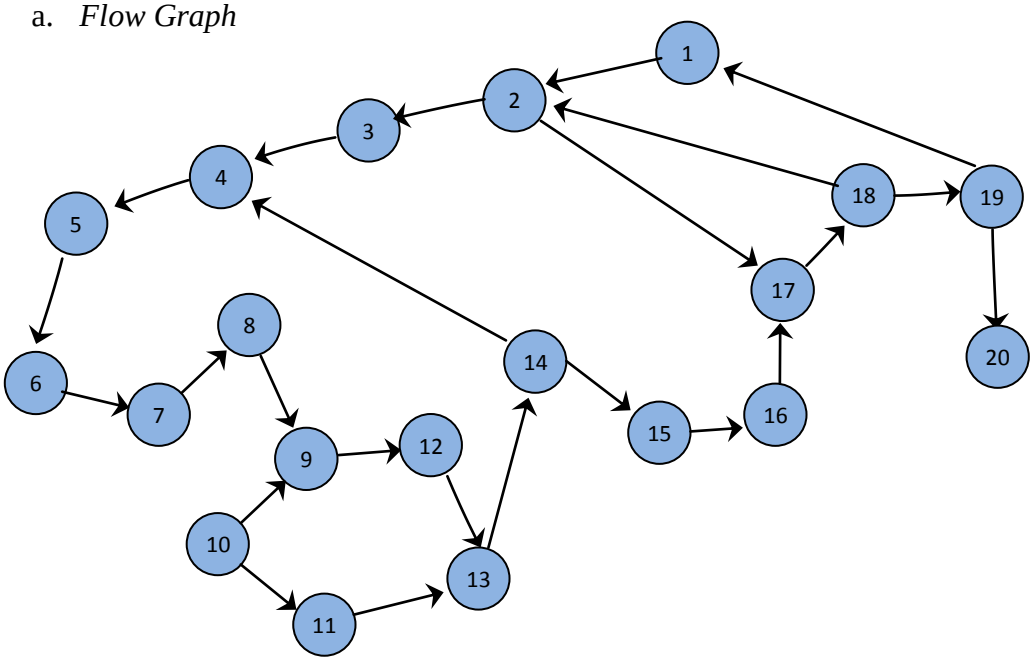
4.6.1.3 *Algoritma A* (A Star)*

Berikut ini algoritma deskriptif algoritma A* (*A Star*) :

```
1 :   1.1. inisialisasi list
          openNodes := kosong
          closedNodes := kosong
      1.2. masukkan node awal ke list OPEN
          currentNode := INode = firstNode
      1.3. menentukan perhitungan A* (A star)
          currentNode.g = 0
          currentNode.h = Pathfinder.euclidianHeuristic(currentNode,
          destinationNode)
          currentNode.f = currentNode.g + currentNode.h
2 :   while currentNode tidak sama dengan destinationNode
3 :       connectedNodes = connectedNodeFunction( currentNode )
4 :       for i = 1 to jumlah connectedNodes
                testNode = connectedNodes[i]
5 :             if testNode = currentNode dan testNode.traversable = false
6 :             g = currentNode.g + travelCost
                h = Pathfinder.euclidianHeuristic( testNode, destinationNode)
                f = g + h
7 :             end if
8 :             if Pathfinder.isOpen(testNode, openNodes) dan
                Pathfinder.isClosed( testNode, closedNodes)
```

```
9 :                         if testNode.f > f
10 :                            testNode.f = f
                                testNode.g = g
                                testNode.h = h
                                testNode.parentNode = currentNode
                                openNodes. parentNode (testNode)
11 :                        end if
12 :                else
                            testNode.f = f
                            testNode.g = g
                            testNode.h = h
                            testNode.parentNode = currentNode
                            openNodes.push(testNode)
13:                 end if
14:                 end for
15:           if  openNodes.length = 0
                        return null
16:           end if
17:           openNodes.sortOn('f', Array.NUMERIC);
              currentNode = openNodes.shift() as INode;
18:    end while
19:    return Pathfinder.buildPath(destinationNode, firstNode)
20:    end
```

a. *Flow Graph*



Gambar 4.7 *Flow graph* pengujian algoritma A* (*A Star*)

Keterangan :

= Menggambarkan kondisi

= Menggambarkan aksi

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

Dari diagram alir di atas, dapat dihitung *cyclomatic complexity*, yakni:

Region = 6

V(G) = E – N + 2

V(G) = 24 – 20 + 2

V(G) = 4 + 2

V(G) = 6

Di mana :

E = jumlah *edge* pada grafik alir

N = jumlah *node* pada grafik alir

c. *Independent Path*

Path 1 = 1 – 2 – 17 – 18 – 19 – 20

Path 2 = 1 – 2 – 17 – 18 – 2 – 17 – 18 – 19 – 20

Path 3 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20

Path 4 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20

Path 5 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20

Path 6 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 2 – 17 – 18 – 19 – 20

d. *Graph Matrices*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Koneksi n(E) - 1
1		1																			1 – 1 = 0
2			1														1				2 – 1 = 1
3				1																	1 – 1 = 0
4					1																1 – 1 = 0
5						1															1 – 1 = 0
6							1														1 – 1 = 0
7								1													1 – 1 = 0
8									1												1 – 1 = 0
9										1		1									2 – 1 = 1
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																1					1 – 1 = 0
16																	1				1 – 1 = 0
17																		1			1 – 1 = 0
18		1																	1		2 – 1 = 1
19	1																			1	2 – 1 = 1
																			Sum (E) + 1		5 + 1 = 6

Keterangan :

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

4.6.2 Hasil Pengujian

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

4.6.2.1 Hasil Pengujian *Black Box*

1. Pengujian Kelas Ternak2D

   Pengujian ini bertujuan untuk menampilkan semua tampilan pada permainan, dari mulai pilih karakter, pilihan menu utama, dan permainan, termasuk isi dari menu toko.

Tabel 4.3 Pengujian kelas Ternak

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Ternak2D	Jika Pilih karakter telah dipilih dan di isi maka simpan data karakter tersebut	Simpan karakter yang dipilih	[√] Simpan [ ] Tidak
	Tombol-tombol pada tampilan	Menampilkan tampilan	[√] Tampil [ ] Tidak

	menu utama bisa mengacu pada tampilan berikutnya	berikutnya	
	Jika Toko dipilih maka bisa menampilkan menu pembelian	Menampilkan menu pembelian	[√] Tampil [ ] Tidak

2. Pengujian Kelas Node

Pengujian ini bertujuan untuk menentukan jarak pada permainan.

Tabel 4.4 Pengujian kelas Node

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Node	Menentukan jarak node satu ke node yang lainnya	Jarak sudah ditentukan	[√] Ditentukan [ ] Tidak

3. Pengujian INode

   Pengujian ini bertujuan untuk memanggil fungsi yang ada di kelas Node.

Tabel 4.5 Pengujian kelas INode

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
INode	Memanggil fungsi yang ada di kelas Node	Memanggil fungsi dari kelas Node	[√] Dipanggil [ ] Tidak

4. Pengujian Grid

   Pengujian ini bertujuan untuk menentukan nilai *grid* dan membedakan *grid* untuk hewan darat dan hewan air.

Tabel 4.6 Pengujian kelas Grid

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Grid	Tidak ada hewan darat ke air dan hewan air ke darat	Tidak bisa melewati grid yang berbeda	[√] Tampil [ ] Tidak

5. Pengujian Panel

Pengujian ini bertujuan untuk menampilkan fungsi pada permainan, diantaranya fungsi layar, *pause* dan fungsi penggunaan tombol pada *keyboard*.

Tabel 4.7 Pengujian kelas Panel

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Panel	Jika pilih tombol pause maka permainan bisa berhenti	Permainan berhenti	[√] Berhenti [ ] Tidak
	Jika pilih tombol layar maka ukuran layar permainan bisa kecil atau besar	Layar mengecil / membesar	[√] Mengecil / Membesar [ ] Tidak
	Fungsi untuk memberi makan atau menhajar musuh bisa menggunakan fungsi keyboard : num(1) sampai num(4)	Fungsi berjalan	[√] Jalan [ ] Tidak

6. Pengujian Actor

   Pengujian ini bertujuan untuk menampilkan tampilan *actor* untuk menjalankan aktifitas peternakan yaitu memberi makan ternak

Tabel 4.8 Pengujian kelas ternak Layer

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Actor	Dapat mencari jalur terpendek ketika memberi makan ternak dengan algoritma A* (*A star*) *pathfinding euclidean distance*	Actor memberi makan	[√] Memberi makan [ ] Tidak

7. Pengujian ternak Layer

   Pengujian ini bertujuan untuk menampilkan tampilan hewan ternak pada permainan dengan menerapkan beberapa fungsi yang ada pada hewan ternak tersebut. Diantaranya harga ternak, harga jual, pergerakan hewan ternak termasuk penerapan algoritma A* (*A star*) *pathfinding* dengan *euclidean distance*, *collision* pada ternak, waktu lapar, waktu produksi, dan waktu mati.

Tabel 4.9 Pengujian kelas ternak Layer

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
ternak Layer	Dapat mencari makanan dengan jalur terdekat menggunakan algoritma A* (*A star*) *pathfinding euclidean distance*	Hewan ternak mencari makanan	[√] Mencari makanan [ ] Tidak

8. Pengujian Food Layer

Pengujian ini bertujuan untuk menampilkan tampilan makanan hewan ternak pada permainan dengan menerapkan banyaknya makanan, waktu habis makanan, pembedaan jenis makanan, dan berkurangnya makanan.

Tabel 4.10 Pengujian kelas Food Layer

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Food Layer	Makanan bisa berkurang dan habis jika dimakan 3 hewan ternak	Makanan habis	[√] Makanan habis [ ] Tidak

	yang sedang lapar		

9. Pengujian Musuh

Pengujian ini bertujuan untuk menampilkan tampilan musuh pada permainan dengan menerapkan beberapa fungsi pada musuh. Diantaranya jumlah munculnya musuh pada setiap level, pergerakan musuh dengan menerapkan algoritma A* (*A star*) *pathfinding euclidean distance* pada pencarian hewan ternak terdekatnya.

Tabel 4.11 Pengujian kelas Musuh

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Musuh	Dapat memangsa musuh dengan mencari jalur terdekat menggunakan algoritma A* (*A star*) *pathfinding euclidean distance*	Memangsa ternak	[√] Memangsa [ ] Tidak

10. Pengujian Kandang

Pengujian ini bertujuan untuk menampilkan tampilan kandang pada permainan dengan menerapkan kapasitas kandang untuk tempat persembunyian hewan ternak terhadap musuh.

Tabel 4.12 Pengujian kelas Kandang

Hasil Uji (Data Normal)			
Data masukkan	Yang di harapkan	Pengamatan	Kesimpulan
Kandang	Jika kandang dibeli maka kandang bisa di isi 10 hewan ternak ketika dikejar musuh	Menampilkan kandang	[√] Tampil [ ] Tidak

4.6.2.2 Hasil Pengujian *White Box*

1. Algoritma A* (*Star*)

Tabel 4.13 Hasil pengujian algoritma A* (*A Star*)

Path no.	*Node (n)*	*Expected result*	*Test Result*	*Conclusion*
1	1 – 2 – 17 – 18 – 19 – 20	Menampilkan rute ke tujuan	Menampilkan rute ke tujuan	*Path* dikerjakan
2	1 – 2 – 17 – 18 – 2 – 17 – 18 – 19 – 20	Menampilkan rute ke tujuan	Menampilkan rute ke tujuan	*Path* dikerjakan

3	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20	Menampilkan rute ke tujuan	Menampilkan rute ke tujuan	*Path* dikerjakan
4	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20	Menampilkan rute ke tujuan	Menampilkan rute ke tujuan	*Path* dikerjakan
5	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20	Menampilkan rute ke tujuan	Menampilkan rute ke tujuan	*Path* dikerjakan
6	1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 2 – 17 – 18 – 19 – 20	Menampilkan rute ke tujuan	Menampilkan rute ke tujuan	*Path* dikerjakan

Kesimpulan Hasil Pengujian *White Box*

Dari hasil pengujian *white box* yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan , serta tidak terjadi pengulangan tak terhingga.

4.7 Pengujian Beta

Pada pengujian beta ini, pengujian akan dilakukan secara objektif dengan cara dicoba langsung oleh beberapa responden. Sampel yang diambil adalah 30 orang, nantinya sampel akan dipersilahkan mencoba aplikasi dan mengisi kuesioner yang telah di sediakan. Teknik pengujian yang digunakan dalam pengujian beta ini adalah teknik skala likert.

1. Penafsiran bobot pengujian

   Pengujian menggunakan 3 kategori jawaban dengan bobot yang berbeda untuk setiap jawabannya seperti pada tabel berikut.

Tabel 4.14 Bobot Jawaban Pernyataan Positif

Jawban	Setuju	Biasa Saja	Tidak Setuju
Bobot	2	1	0

Tabel 4.15 Bobot Jawaban Pernyataan Negatif

Jawban	Tidak Setuju	Biasa Saja	Setuju
Bobot	2	1	0

2. Kualifikasi Skor

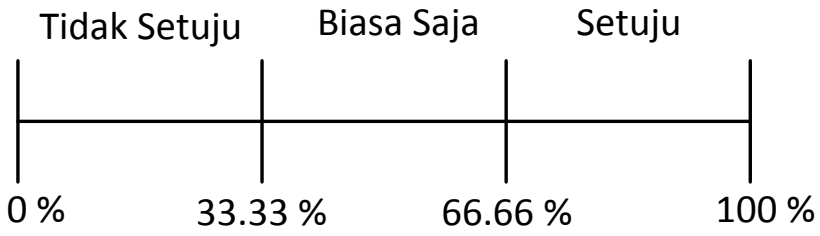
   Persentase skor=(jumlah skor / jumlah bobot nilai tertinggi) x100%

   Berikut adalah kriteria penilaian intrepetasi skor, kualifikasi pesentase = (bobot jawaban / bobot jawaban mak) x 100%;

a. Persentase nilai Setuju = (3 / 3) x 100% = 100%

b. Persentase nilai Ragu-Ragu= (2 / 3) x 100% = 66.66%

c. Persentase nilai Biasa Saja= (1 / 3) x 100% = 33.33%



Gambar 4.14 Kualifikasi Skala Pengukuran

Hasil pengujian *Beta* adalah sebagai berikut:

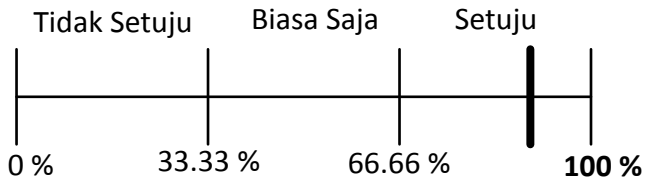
1. Pertanyaan Nomor 1

   (1) Pemainan ini sangat menarik untuk dimainkan ?

Tabel 4.16 Perhitungan Pertanyaan Nomor 1

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
1	Setuju	2	21	42
	Biasa Saja	1	9	9
	Tidak Setuju	0	-	-
Jumlah				51

Persentase = (51 / 60) x 100% = 85%



Gambar 4.15 Hasil Pengukuran Pertanyaan Nomor 1

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian ketertarikan responden saat memainkan game ini adalah 85% atau bisa dikategorikan sebagian besar responden setuju.

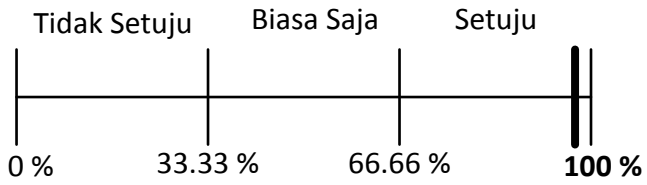
2. Pertanyaan Nomor 2

(2) Apakah dengan adanya game ini saya bisa belajar beternak ?

Tabel 4.17 Perhitungan Pertanyaan Nomor 2

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
2	Setuju	2	28	56
	Biasa Saja	1	2	2
	Tidak Setuju	0	-	-
Jumlah				58

Persentase = (58 / 60) x 100% = 96%



Gambar 4.16 Hasil Pengukuran Pertanyaan Nomor 2

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian dapat belajar peternakan dari game ini adalah 96% atau bisa dikategorikan sebagian besar responden setuju.

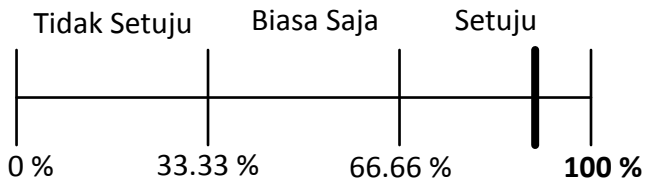
3. Pertanyaan Nomor 3

(3) Apakah setuju jika pada game ini di terapkan tokoh karakter peternak ?

Tabel 4.18 Perhitungan Pertanyaan Nomor 3

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
3	Setuju	2	22	44
	Biasa Saja	1	5	5
	Tidak Setuju	0	3	0
Jumlah				49

Persentase = (49 / 60) x 100% = 82%



Gambar 4.17 Hasil Pengukuran Pertanyaan Nomor 3

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian adanya tokoh karakter peternak pada game ini adalah 82% atau bisa dikategorikan sebagian besar responden setuju.

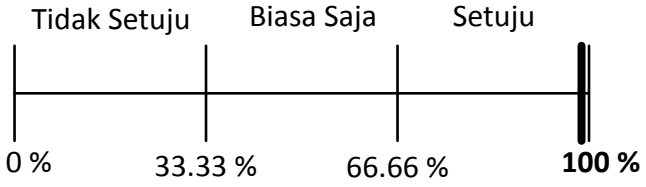
4. Pertanyaan Nomor 4

(4) Apakah setuju hewan dalam *game* ini bisa mencari makanan yang paling dekat ?

Tabel 4.19 Perhitungan Pertanyaan Nomor 4

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
4	Setuju	2	29	58
	Biasa Saja	1	1	1
	Tidak Setuju	0	-	-
Jumlah				59

Persentase = (60 / 60) x 100% = 98%



Gambar 4.18 Hasil Pengukuran Pertanyaan nomor 4

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian hewan ternak dapat mencari makanan yang paling dekat adalah 98% atau bisa dikategorikan sebagian besar responden setuju.

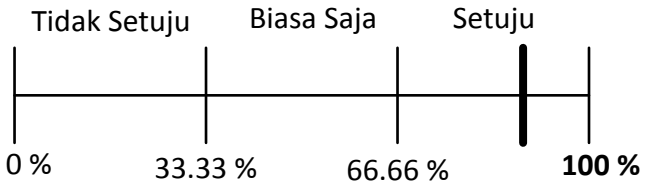
5. Pertanyaan Nomor 1

(5) Apakah game ini sudah menggambarkan lingkugan peternakan ?

Tabel 4.20 Perhitungan Pertanyaan Nomor 5

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
5	Setuju	2	20	40
	Biasa Saja	1	9	9
	Tidak Setuju	0	1	0
Jumlah				49

Persentase = (49 / 60) x 100% = 81%



Gambar 4.19 Hasil Pengukuran Pertanyaan Nomor 5

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian lingkungan peternakan pada game ini adalah 81% atau bisa dikategorikan sebagian besar responden setuju.

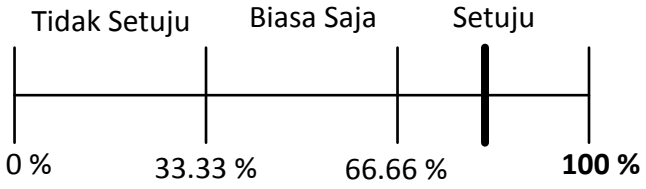
6. Pertanyaan Nomor 6

(6) Apakah setuju jika musuhnya pintar dalam mengejar ternak ?

Tabel 4.21 Perhitungan Pertanyaan Nomor 6

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
6	Setuju	2	17	34
	Biasa Saja	1	13	13
	Tidak Setuju	0	-	-
Jumlah				47

Persentase = (47 / 60) x 100% = 78%



Gambar 4.20 Hasil Pengukuran Pertanyaan Nomor 6

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian kepintaran musuh dalam mengejar hewan ternak adalah 78% atau bisa dikategorikan sebagian besar responden setuju.

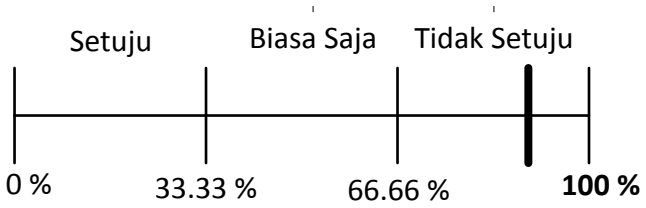
7. Pertanyaan Nomor 7

(7) Saya kurang setuju dengan adanya musuh ?

Tabel 4.22 Perhitungan Pertanyaan Nomor 7

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
7	Setuju	0	2	0
	Biasa Saja	1	10	12
	Tidak Setuju	2	18	36
Jumlah				48

Persentase = (48 / 60) x 100% = 80%



Gambar 4.21 Hasil Pengukuran Pertanyaan Nomor 7

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian ketidak setujuan adanya musuh pada game ini adalah 80% atau bisa dikategorikan sebagian besar responden tidak setuju.

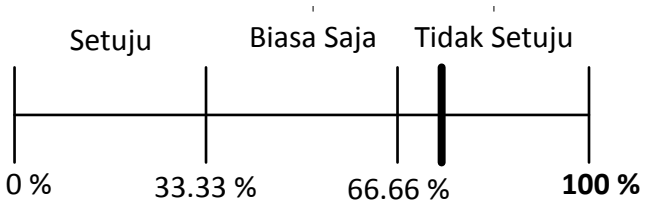
8. Pertanyaan nomor 8

(8) Kesulitan dalam menggunakan tombol fungsi untuk memberi makan?

Tabel 4.23 Perhitungan Pertanyaan Nomor 8

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
8	Setuju	0	2	0
	Biasa Saja	1	12	12
	Tidak Setuju	2	16	32
Jumlah				44

Persentase = (44 / 60) x 100% = 73%



Gambar 4.22 Hasil Pengukuran Pertanyaan Nomor 8

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian kesulitan dalam menggunakan tombol-tombol yang terdapat pada game ini adalah 73% atau bisa dikategorikan sebagian besar responden tidaksetuju.

4.7.1 Kesimpulan Pengujian Beta

Berdasarkan pengujian beta yang telah didapat dari hasil kuesioner yang telah dijawab oleh para responden, maka bisa diambil kesimpulan bahwa *game* ”Ternak” ini :

1. Menarik responden dalam memainkannya
2. Dapat memberikan pembelajaran tentang tata cara beternak
3. Diterapkan karakter peternak
4. Hewan ternak dapat mencari makanan yang paling dekat
5. Sudah menggambarkan lingkungan peternakan
6. Musuh diberi kecerdasan buatan
7. Ada musuh
8. Tombol fungsi memudahkan dalam bermain