BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi, tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan. Aplikasi ini diimplementasikan dengan nama Balapan Tradisonal Egrang dan *file extension*-nya *.apk* (*application package file*).

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang dapat digunakan untuk menjalankan aplikasi *game* ini tidak harus memiliki spesifikasi *device* yang tinggi. Sebagai ilustrasi sistem minimum agar dapat menjalankan *game* balapan tradisional egrang adalah sebagai berikut :

1) *Processor* minimal 800 mhz
2) RAM minimal 128 MB
3) Menggunakan layar dengan dimensi 320x480 pixel
4) Memiliki sensor *accelerometer*

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi balapan tradisional egrang adalah sistem operasi *Android* versi 2.3 (*Gingerbread*) atau diatasnya.

4.1.3 Implementasi Instalasi *Game*

Sebelum dapat memainkan *game* balapan tradisional egrang, aplikasi harus terinstal pada *device android*. Petunjuk dalam instalasi aplikasi *game* ini adalah sebagai berikut :

1. *Copy file* Balapan Tradisional Egrang.*apk* pada *memory card device.*
2. Kemudian buka dan klik *file* Balapan Tradisional Egrang.*apk* yang terdapat pada *memory card device* setelah itu klik tombol *install*.

4.1.4 Implementasi Antarmuka

Implementasi antarmuka dilakukan untuk setiap tampilan program yang dibangun dalam bentuk *file code*. Berikut ini adalah implementasi antarmuka untuk *user* :

1. Antarmuka Menu Utama

   Saat pemain membuka aplikasi pertama kali, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka menu utama. Tampilan dari antarmuka menu utama dapat dilihat pada gambar 4.1:



Gambar 4.1 Antarmuka Menu Utama

Gambar di atas merupakan menu utama pada *game* balapan tradisional egrang. Terdapat 3 (tiga) sub menu di dalamnya, yaitu menu mulai yang berfungsi untuk memulai permainan, menu petunjuk yang berfungsi untuk melihat petunjuk bermain, menu tentang yang berfungsi untuk melihat mengenai pembuat aplikasi.

2. Antarmuka Menu Petunjuk

   Saat pemain memilih menu petunjuk, maka antarmuka yang ditampilkan oleh sistem adalah informasi mengenai petunjuk cara bermain dalam *game* balapan tradisional egrang. Tampilan dari antarmuka menu petunjuk dapat dilihat pada gambar 4.2 :



Gambar 4.2 Antarmuka Menu Petunjuk

3. Antarmuka Menu Tentang

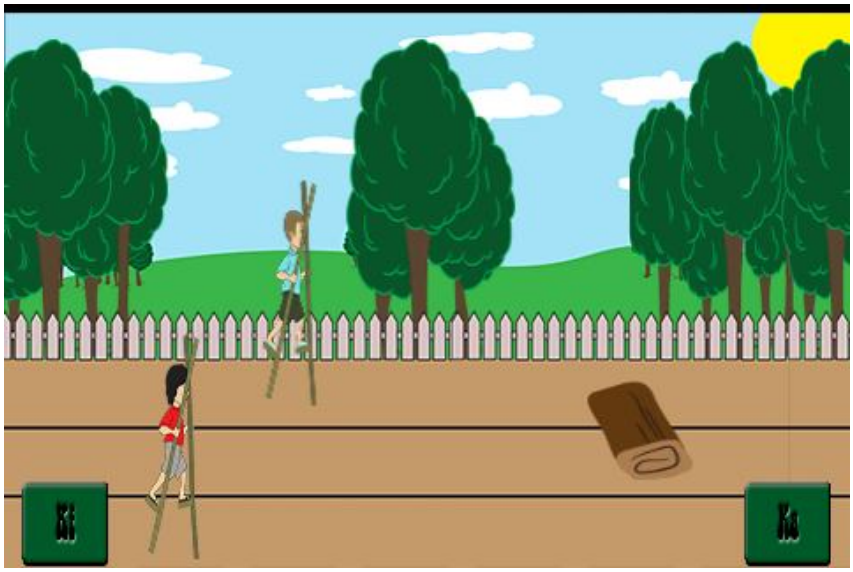
   Saat pemain memilih menu tentang, maka antarmuka yang ditampilkan oleh sistem adalah informasi mengenai versi aplikasi dan pembuat *game* balapan tradisional egrang. Tampilan dari antarmuka menu tentang dapat dilihat pada gambar 4.3:



Gambar 4.3 Antarmuka Menu Tentang

4. Antarmuka Level 1

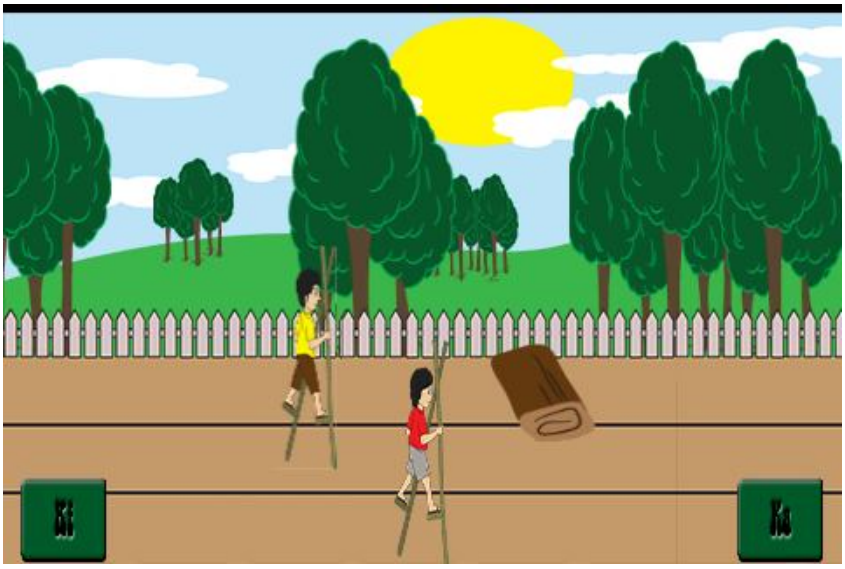
   Saat pemain memilih menu mulai maka pemain akan langsung diarahkan pada permainan *level* 1. Di *level* 1 ini rintangan masih sangat jarang, namun pemain harus menghadapi NPC bernama Asep.



Gambar 4.4 Antarmuka Level 1

5. Antarmuka Level 2

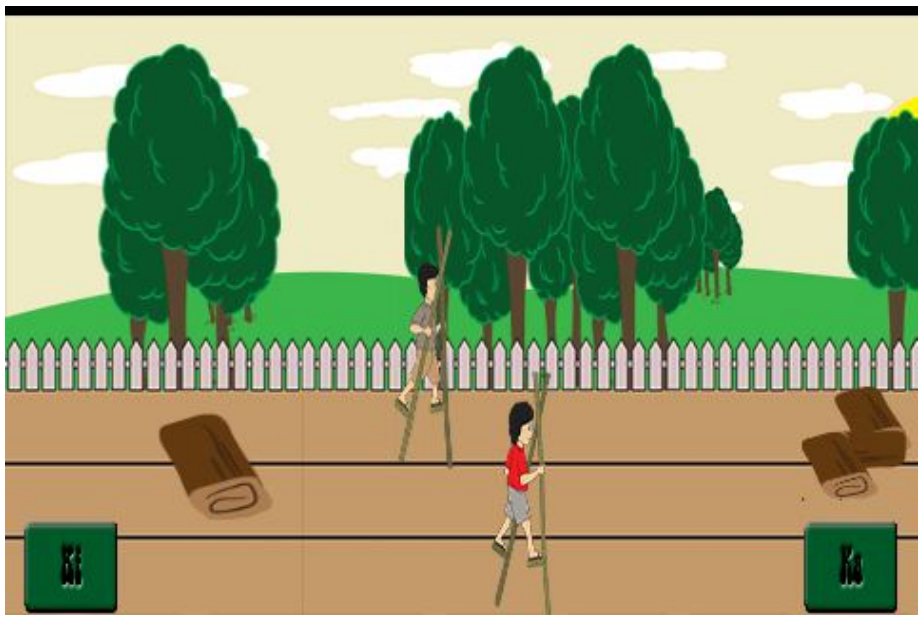
   Pemain akan mendapatkan informasi tentang egrang dan langsung diarahkan ke *level* 2 apabila berhasil mengalahkan Asep (NPC) di *level* sebelumnya.



Gambar 4.5 Antarmuka Level 2

6. Antarmuka Level 3

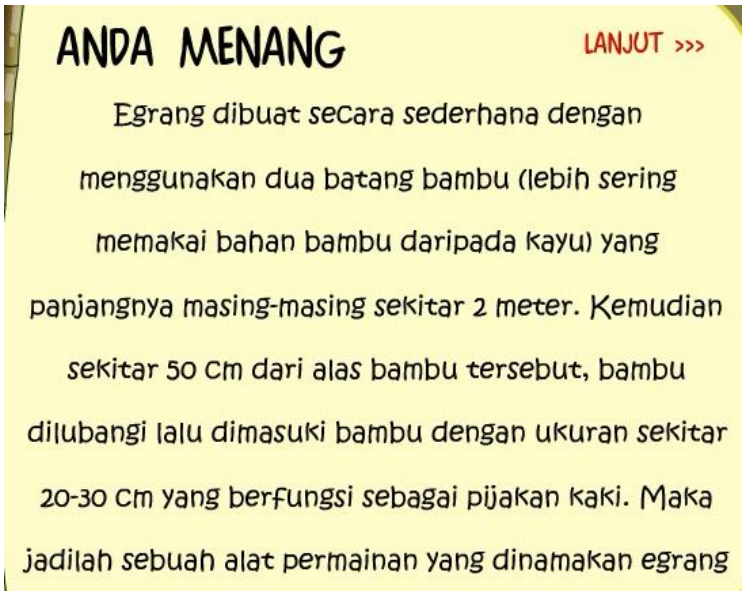
   Pemain akan mendapatkan informasi tentang egrang dan langsung diarahkan ke *level* 3 apabila berhasil mengalahkan Nana (NPC) di *level* sebelumnya.



Gambar 4.6 Antarmuka Level 3

7. Antarmuka Menang

   Informasi tentang egrang akan muncul di setiap *level* apabila pemain berhasil memenangkan perlombaan.



Gambar 4.7 Antarmuka Menang

8. Antarmuka Kalah

   Apabila pemain tidak berhasil memenangkan perlombaan maka pemain tidak akan mendapatkan informasi tentang egrang dan tampil pesan apakah akan mengulangi permainan atau keluar dari permainan.



Gambar 4.8 Antarmuka Kalah

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 Rencana Pengujian

Metode yang digunakan dalam pengujian *alpha* adalah metode *white box* dan *black box.* Rencana pengujian dapat dilihat pada tabel 4.1.

Tabel 4.1 Skenario Pengujian Aplikasi

No	Komponen yang diuji	Skenario Pengujian	Jenis Pengujian
1.	Menu Mulai	Sentuh menu mulai	*Black Box*
2.	Menu Petunjuk	Sentuh menu petunjuk	*Black Box*
		Sentuh tombol kembali	*Black Box*
3.	Menu Tentang	Sentuh menu tentang	*Black Box*
		Sentuh tombol kembali	*Black Box*
4.	Menu Keluar	Sentuh tombol keluar	*Black Box*
5.	*Accelerometer*	*Device* diam	*Black Box*
		Device cenderung miring ke bawah	*Black Box*
		Device cenderung miring ke atas	*Black Box*
6.	Tombol Ka	Sentuh tombol “Ka”	*Black Box*
7.	Tombol Ki	Sentuh tombol “Ki”	*Black Box*
8.	Jalur NPC	Penelusuran algoritma A *Star*	*White box*
9.	*Collision*	Karakter menabrak NPC atau menabrak pembatas jalan	*Black Box*

4.2.2 Pengujian *Black Box*

Pengujian *black box* dilakukan dengan tujuan untuk mengetahui apakah data yang dimasukan sudah benar atau belum, hasil *output* apakah sesuai dengan harapan atau tidak. Berdasarkan rencana pengujian yang telah disusun, maka dapat dilakukan pengujian sebagai berikut :

Tabel 4.2 Pengujian *Black Box*

Kasus Dan Hasil Uji				
No	**Kasus yang akan diuji**	**Skenario Uji**	**Hasil Yang diharapkan**	**Kesimpulan**
1.	Menu Mulai	Sentuh Menu Mulai	Sistem menampilkan	[✓] Berhasil [ ] Tidak Berhasil

			permainan	
2.	Menu Petunjuk	Sentuh Menu Petunjuk	Sistem menampilkan halaman menu petunjuk	[✓] Berhasil [ ] Tidak Berhasil
3.	Menu Tentang	Sentuh Menu Tentang	Sistem menampilkan halaman menu tentang	[✓] Berhasil [ ] Tidak Berhasil
4.	Menu Keluar	Sentuh Menu Keluar	Keluar dari permainan	[✓] Berhasil [ ] Tidak Berhasil
5.	*Accelerometer*	*Device* diam	Sistem melakukan *event* karakter menjadi tidak bergerak ke kiri ataupun ke kanan	[✓] Berhasil [ ] Tidak Berhasil
		Device cenderung miring ke bawah	Sistem melakukan *event* karakter bergerak ke bawah	[✓] Berhasil [ ] Tidak Berhasil
		Device cenderung miring ke atas	Sistem melakukan *event* karakter bergerak ke atas	[✓] Berhasil [ ] Tidak Berhasil
6.	Tombol Ka	Sentuh tombol Ka	Sistem melakukan	[✓] Berhasil [ ] Tidak Berhasil

			gerak kaki kanan	
7.	Tombol Ki	Sentuh tombol Ki	Sistem melakukan gerak kaki kiri	[✓] Berhasil [ ] Tidak Berhasil
8.	*Collision*	Karakter menabrak NPC atau menabrak pembatas jalan	Sistem terjatuh	[✓] Berhasil [ ] Tidak Berhasil

4.2.3 Pengujian Algoritma A Star Jalur NPC (*White Box*)

Pengujian pada proses jalur musuh merupakan proses pengujian dari pengimplementasian algoritma *A Star* menggunakan pengujian *whitebox*, dan menguji apakah algoritma *A Star* berjalan dengan seharusnya pada NPC.

1. **Algoritma *A Star***

```
1. openNodes.put(fromNodeID, fromNode);
   sortedOpenNodes.enter(fromNode);
   Node currentNode = null;
   while(openNodes.size() > 0) {
2. currentNode = sortedOpenNodes.poll();
          final long currentNodeID = currentNode.mID;
          if(currentNodeID == toNodeID) {
                break;
          }
          visitedNodes.put(currentNodeID, currentNode);
3. for(int dX = -1; dX <= 1; dX++) {
          for(int dY = -1; dY <= 1; dY++) {
                if((dX == 0) && (dY == 0)) {
                      continue;
                      }
       if(!allowDiagonalMovement && (dX != 0) && (dY != 0)) {
                                    continue;
                              }
```

```
        final int neighborNodeX = dX + currentNode.mX;
        final int neighborNodeY = dY + currentNode.mY;
        final long neighborNodeID = Node.calculateID(neighborNodeX,
        neighborNodeY);
        if(!IntBoundsUtils.contains(pXMin, pYMin, pXMax, pYMax,
        neighborNodeX, neighborNodeY) ||
        pPathFinderMap.isBlocked(neighborNodeX, neighborNodeY,
        pEntity)) {
                    continue;
                          }
        if(visitedNodes.indexOfKey(neighborNodeID) >= 0) {
                          continue;
                          }
        Node neighborNode = openNodes.get(neighborNodeID);
        final boolean neighborNodeIsNew;
4.      if(neighborNode == null) {
        neighborNodeIsNew = true;
        neighborNode = new Node(neighborNodeX, neighborNodeY,
        pAStarHeuristic.getExpectedRestCost(pPathFinderMap,
        pEntity, neighborNodeX, neighborNodeY, pToX, pToY));
                          } else {
                          neighborNodeIsNew = false;
                          }
5.      final float costFromCurrentToNeigbor =
        pCostFunction.getCost(pPathFinderMap, currentNode.mX,
        currentNode.mY, neighborNodeX, neighborNodeY, pEntity);

        final float neighborNodeCost = currentNode.mCost +
        costFromCurrentToNeigbor;
              if(neighborNodeCost > pMaxCost) {
6.            if(!neighborNodeIsNew) {
              openNodes.remove(neighborNodeID);
                                    }
                                    } else {
           neighborNode.setParent(currentNode,
           costFromCurrentToNeigbor);
              if(neighborNodeIsNew) {
              openNodes.put(neighborNodeID, neighborNode);
                                          } else {
7.      sortedOpenNodes.remove(neighborNode);
                                          }
              sortedOpenNodes.enter(neighborNode);
```

```
                    if(pPathFinderListener != null) {
                    pPathFinderListener.onVisited(pEntity,
          neighborNodeX, neighborNodeY);
                                        }
                                  }
                              }
                    }
            }
8.  visitedNodes.clear();

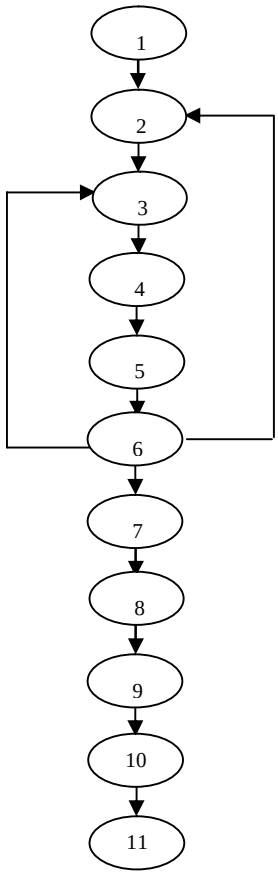
          openNodes.clear();

          sortedOpenNodes.clear();

9.  if(currentNode.mID != toNodeID) {
                    return null;
            }
10.         int length = 1;
            Node tmp = currentNode;
            while(tmp.mID != fromNodeID) {
                    tmp = tmp.mParent;
                    length++;
            }
11.         final Path path = new Path(length);
            int index = length - 1;
            tmp = currentNode;
            while(tmp.mID != fromNodeID) {
            path.set(index, tmp.mX, tmp.mY);
            tmp = tmp.mParent;
            index--;
            }
            path.set(0, pFromX, pFromY);
            return path;
            }
```

Gambar 4.9 *Source code* Algoritma A *Star*

2. *Flowgraph*



Gambar 4.10 *Flowgraph* Algoritma A *Star*

Keterangan :

= Menggambarkan kondisi

= Menggambarkan aksi

3. *Cyclomatic Complexity* V(G)

Berikut ini adalah perhitungan Cyclomatic complexity yang nantinya akan digunakan sebagai jumlah aksi pada independent path.

V(G) = E – N + 2

Di mana E adalah edge, dan N adalah node. Jadi :

V(G) = 12 edge – 11 node + 2 = 3

4. *Independent Path*

Dari hasil perhitungan *cyclomatic complexity* terdapat 4 *independent path*, yaitu :

Path 1 = 1-2-3-4-5-6-7-8-9-10-11

Path 2 = 1-2-3-4-5-6-3-4-5-6-7-8-9-10-11

Path 3 = 1-2-3-4-5-6-2-3-4-5-6-7-8-9-10-11

Tabel 4.3 *Graph* Matriks Jalur NPC (A *Star*)

N	1	2	3	4	5	6	7	8	9	10	11	N(E)-1
1		1										1-1=0
2			1									1-1=0
3				1								1-1=0
4					1							1-1=0
5						1						1-1=0
6		1	1				1					3-1=2
7								1				1-1=0
8									1			1-1=0
9										1		1-1=0
10											1	1-1=0
11												0
											SUM(E)+1	2+1=3

Keterangan:

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

Tabel 4.4 Hasil Pengujian *White Box*

No	**Path**	**Yang diharapkan**	**Hasil Sesuai Kasus Uji**	**Keterangan**
1	1-2-3-4-5-6-7-8-9-10-11	Mencari F dengan nilai H	Mencari F dengan nilai H	[√] Alur terlewati [ ] Alur tidak terlewati
2	1-2-3-4-5-6-3-4-5-6-7-8-9-10-11	Masukkan semua *neighbor node* ke dalam open dan hitung nilai F dan H.	*Current node* dan nilai F,H masuk ke dalam open	[√] Alur terlewati [ ] Alur tidak terlewati

No	Path	Yang diharapkan	Hasil Sesuai Kasus Uji	Keterangan
3	1-2-3-4-5-6-2-3-4-5-6-7-8-9-10-11	Nilai H yang didapat lebih kecil dari nilai H sebelumnya. Update ulang nilai F,H	Nilai F,H, diupdate ulang	[√] Alur terlewati [ ] Alur tidak terlewati

Berdasarkan pengujian yang dilakukan, maka bisa diambil kesimpulan bahwa masing-masing *independent path* yang dihasilkan bisa dijalankan dan dieksekusi.

4.2.4 Pengujian Beta

Pengujian beta merupkan pengujian yang dilakukan secara objektif dimana diuji secara langsung ke lapangan, dengan menggunakan kuesioner mengenai kepuasan *user* terhadap aplikasi yang telah dibangun, dan apakah sesuai dengan yang diharapkan. Adapun metode penilaian yang digunakan yaitu metode kuantitatif berdasarkan data sampel dari *user*.

4.2.4.1 Skenario Pengujian Beta

Kuesioner disebarkan menggunakan teknik *Simple Random Sampling* dimana anggota *sample* dari populasi dilakukan secara acak tanpa memperhatikan strata yang ada dalam populasi.

Kuesioner disebarkan kepada 30 orang. Dari hasil kuisioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian kepuasan pemain terhadap aplikasi ini. Kuisioner terdiri dari 6 pertanyaan dengan skala 1 sampai 5 menggunakan skala Likert.

1. **Rumus Skala Likert**

   Skala Likert digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau kelompok orang tentang fenomena sosial. Dalam penelitian, fenomena sosial ini telah ditetapkan secara spesifik oleh peneliti, yang selanjutnya disebut sebagai variabel penelitian.

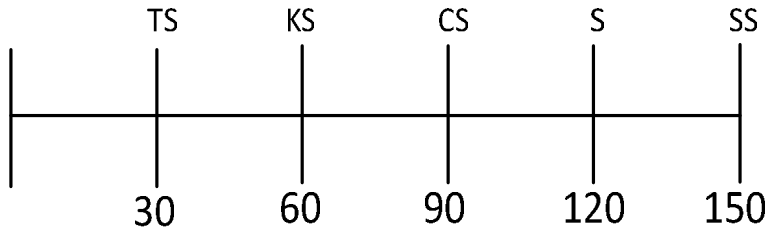
Dengan skala Likert, maka variabel yang akan diukur dijabarkan menjadi indicator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrument yang dapat berupa pernyataan atau pertanyaan. Jawaban setiap item instrumen yang menggunakan skala Likert, mempunyai gradasi dari sangat positif sampai sangat negative.

2. Bobot Nilai

Untuk keperluan analisis kuantitatif, skala Likert memberikan skor untuk jawaban-jawaban yang disediakan dengan bobot yang diberikan pada tabel 4.5.

Tabel 4.5 Bobot Nilai Pengujian

Skala Jawaban	Keterangan	Bobot
SS	Sangat Setuju	5
S	Setuju	4
CS	Cukup Setuju	3
KS	Kurang Setuju	2
TS	Tidak Setuju	1



Gambar 4.11 Skala Kontinu

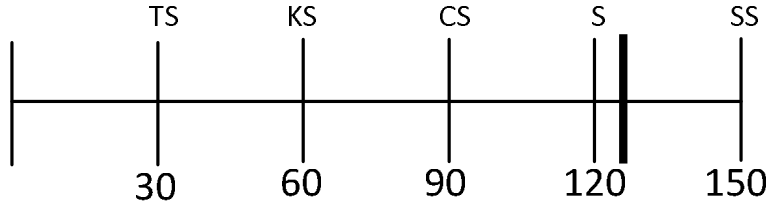
4.2.4.2 Hasil Kuisioner

Kuisioner yang dibuat terdiri dari enam buah pertanyaan (kuisioner dapat dilihat di lampiran). Berikut ini adalah hasil persentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan rumus di atas, dengan pertanyaan sebagai berikut.

1. Apakah kontrol karakter pada game ini mudah?

Tabel 4.6 Perhitungan Pertanyaan Nomor 1

No	Jawaban	Total Skor	Persentase
1	Sangat Setuju	5 x 8 = 40	(124 / 150) * 100 = 82,67%
2	Setuju	4 x 18 = 72	
3	Cukup Setuju	3 x 4 = 12	
4	Kurang Setuju	2 x 0 = 0	
5	Tidak Setuju	1 x 0 = 0	



Gambar 4.12 Skala Persentase Pertanyaan 1

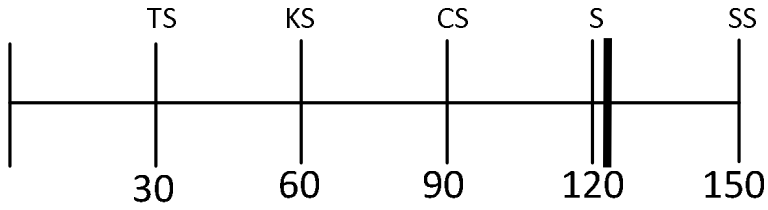
Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian terhadap kemudahan dalam kontrol karakter yang menggunakan sensor *accelerometer* pada *game* balapan tradisional egrang adalah 82,67% dari yang diharapkan 100%.

2. Apakah menurut anda *game* ini mudah untuk dimainkan?

Tabel 4.7 Perhitungan Pertanyaan Nomor 2

No	Jawaban	Total Skor	Persentase
1	Sangat Setuju	5 x 6 = 30	(121 / 150) * 100 = 80,67 %
2	Setuju	4 x 19 = 76	
3	Cukup Setuju	3 x 5 = 15	

4	Kurang Setuju	2 x 0 = 0	
5	Tidak Setuju	1 x 0 = 0	



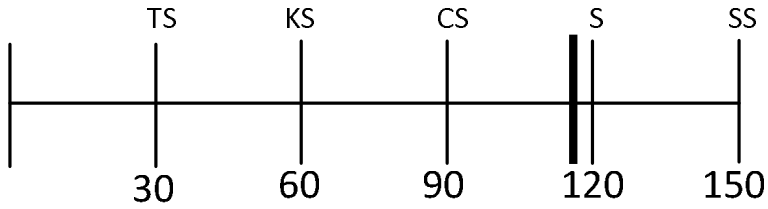
Gambar 4.13 Skala Persentase Pertanyaan 2

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian terhadap kemudahan pada permainan balapan tradisional egrang adalah 80,67% dari yang diharapkan 100%.

3. Apakah *game* ini dapat membantu anda untuk kembali mengingat permainan tradisional egrang?

Tabel 4.8 Perhitungan Pertanyaan Nomor 3

No	**Jawaban**	**Total Skor**	**Persentase**
1	Sangat Setuju	5 x 10 = 50	(119 / 150) * 100 = 79,33 %
2	Setuju	4 x 12 = 48	
3	Cukup Setuju	3 x 7 = 21	
4	Kurang Setuju	2 x 0 = 0	
5	Tidak Setuju	1 x 0 = 0	



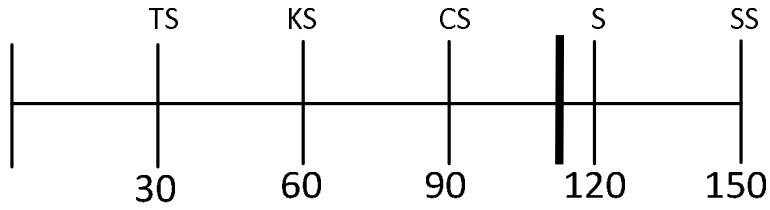
Gambar 4.14 Skala Persentase Pertanyaan 3

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian untuk membantu pemainn dalam mengingat permainan tradisional egrang adalah 79,33% dari yang diharapkan 100%.

4. Apakah tantangan yang diberikan di dalam *game* ini cukup menantang?

Tabel 4.9 Perhitungan Pertanyaan Nomor 4

No	Jawaban	Total Skor	Persentase
1	Sangat Setuju	5 x 7 = 35	(117 / 150) * 100 = 78 %
2	Setuju	4 x 13 = 52	
3	Cukup Setuju	3 x 10 = 30	
4	Kurang Setuju	2 x 2 = 4	
5	Tidak Setuju	1 x 0 = 0	



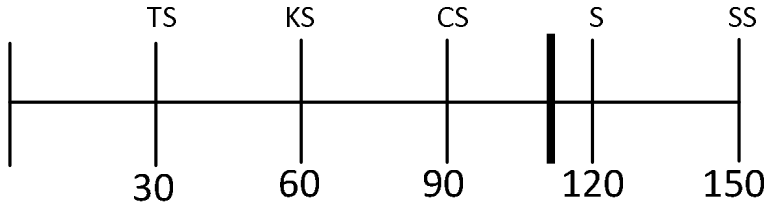
Gambar 4.15 Skala Persentase Pertanyaan 4

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian terhadap tantangan yang ada pada *game* balapan tradisional egrang adalah 78% dari yang diharapkan 100%.

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

Tabel 4.10 Perhitungan Pertanyaan Nomor 5

No	Jawaban	Total Skor	Persentase
1	Sangat Setuju	5 x 4 = 20	(116 / 150) * 100 = 73,33 %
2	Setuju	4 x 17 = 68	
3	Cukup Setuju	3 x 6 = 18	
4	Kurang Setuju	2 x 2 = 4	
5	Tidak Setuju	1 x 0 = 0	



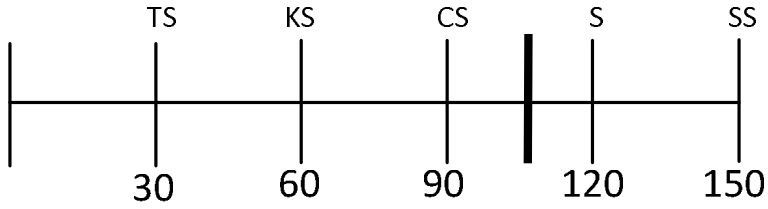
Gambar 4.16 Skala Persentase Pertanyaan 5

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian terhadap grafik 2D yang dipakai adalah 73,33% dari yang diharapkan 100%.

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

Tabel 4.11 Perhitungan Pertanyaan Nomor 6

No	**Jawaban**	**Total Skor**	**Persentase**
1	Sangat Setuju	5 x 1 = 5	(113 / 150) * 100 = 75,33 %
2	Setuju	4 x 21 = 84	
3	Cukup Setuju	3 x 8 = 24	
4	Kurang Setuju	2 x 0 = 0	
5	Tidak Setuju	1 x 0 = 0	



Gambar 4.17 Skala Persentase Pertanyaan 6

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian terhadap ketertarikan pemain terhadap *game* balapan tradisional egrang adalah 75,33% dari yang diharapkan 100%.

4.2.4.3 Kesimpulan Pengujian Beta

Hasil dari pengujian beta dapat disimpulkan bahwa :

1. Pemain dapat mengontrol karakter dengan mudah menggunakan senor *accelerometer*.
2. Pemain merasa mudah dalam memainkan *game* ini.
3. Pemain merasa terbantu dalam mengingat kembali permainan tradisional egrang.
4. Pemain merasa tertantang dengan adanya musuh yang cerdas.
5. Pemain beranggapan grafik pada *game* ini bagus.
6. Pemain beranggapan bahwa *game* ini menarik untuk dimainkan berulang-ulang.