BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi permainan yang dibangun, yaitu *Crazy* Becak. Hasil perancangan pada tahap sebelumnya kemudian diimplementasikan ke dalam bahasa pemrograman. Setelah tahap implementasi selesai, maka dilakukan tahap pengujian terhadap aplikasi permainan *Crazy* Becak.

4.1. Implementasi

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi.Tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan, yang terdiri dari penjelasan mengenai lingkungan implementasi, baik itu lingkungan perangkat keras, maupun lingkungan perangkat lunak, serta implementasi program.

4.1.1. Implementasi Perangkat keras

Perangkat keras/device yang digunakan untuk menjalankan aplikasi permainan *Crazy* Becak tidak harus yang berspesifikasi tinggi. Spesifikasi minimum yang dapat digunakan untuk menjalankan aplikasi permainan *Crazy* Becak adalah sebagai berikut :

1. Mempunyai CLDC 1.1
2. MIDP 2.0
3. Layar ukuran 240x320 pixel

4.1.2. Implementasi perangkat lunak

Adapun spesifikasi perangkat lunak yang digunakan untuk menjalankan aplikasi permainan *Crazy* becak adalah perangkat device mobile yang sudah berbasis Java.

4.1.3. Implementasi Aplikasi

Crazy becak merupakan aplikasi permainan ber-*genre advanture maze*. Pemain mendapatkan misi setiap *level*-nya yaitu untuk mengumpulkan item dan mencari tempat atau spot yang dituju dengan melewati halangan-halangan.

Permainan dikatakan *game over* apabila *lifebar* telah habis, dan sebaliknya permainan dikatakan berhasil apabila pemain dapat menyelesaikan misi disetiap *level*-nya dengan baik.

4.1.4. Implementasi *Class*

Implementasi Class dapat dilihat pada tabel 4.1.

Tabel 4.1 implementasi Class

No.	Nama Class	Nama File Fisik
1	Becak	Becak.java
2	Splash Screen	SplashScreen.java
3	*Game* Menu	*Game*menu.java
4	Petunjuk	Petunjuk.java
5	Tentang	Tentang.java
6	*Game* Screen	*Game*Screen.java
7	*Game* Map	*Game*Map.java
8	*Game* Map 2	*Game*Map2.java
9	*Game* Map 3	*Game*Map3.java
10	Mobil	Mobil.java
11	Halangan	Halangan.java

12	Satpol	Satpol.java
13	Point	Point.java
14	Node	Node.java
15	Clock	Clock.java
16	Becak Sprite	BecakSprite.java
17	NPC tembak	NPCtembak.java
18	Bullet	Bullet.java

4.1.5. Implementasi Antarmuka

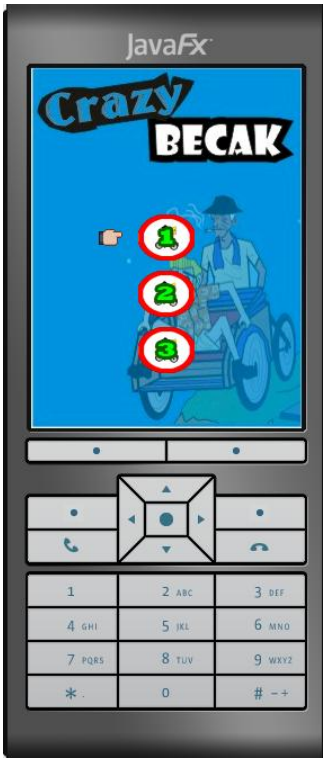
Adapun implementasi antarmuka pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

1. Antarmuka Menu utama



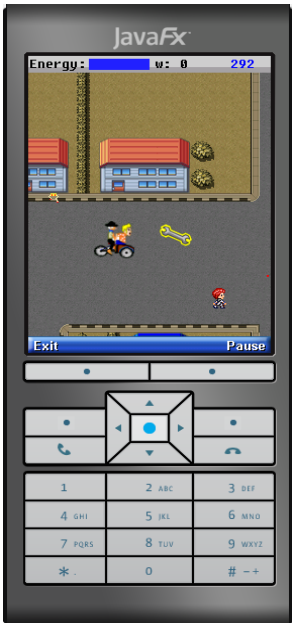
Gambar 4.1 Menu utama *crazy* becak

2. Antarmuka pilih level



Gambar 4.2 pilih level

3. Antarmuka permainan



Gambar 4.3 Area permainan *crazy* becak

4. Antarmuka petunjuk



Gambar 4.4 Petunjuk pemainan

5. Antarmuka tentang



Gambar 4.5 Tentang (*about*) Permainan

4.2. Pengujian

Dalam melakukan pengujian terhadap aplikasi permainan *Crazy* becak dilakukan dengan dua mcama teknik, yaitu pengujian *black-box* dan pengujian *white-box*.Adapun pengujian *black box* berfokus pada menemukan kesalahan program, sedangkan pengujian *white box* berfokus kepada struktur internal (*source* code) program untuk mengetahui apakah masih terjadi *error*. Pengujian dilakukan dengan dua tahap, yaitu tahap pengujian *alpha* dan tahap pengujian *bet*a.

4.2.1. Pengujian Alpha

Pada tahap ini, pengujian dilakukan untuk memastikan aplikasi permainan *Crazy* becak telah berjalan dengan benar dan sesuai dengan kebutuhan yang diinginkan.Pengujian *alpha* menitikberatkan kepada persyaratan fungsional perangkat lunak.

4.2.1.1. Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi permainan *Crazy* becak.Skenario pengujian dari aplikasi permainan *Crazy* becak dapat dilihat pada tabel 4.2.

Tabel 4.2 Skenario Pengujian Aplikasi Permainan *Crazy* Becak

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1	Menu Utama	Pilih Menu Mulai Permainan	*Black Box*
		Pilih Menu Petunjuk	*Black Box*
		Pilih Menu Tentang	*Black Box*
		Pilih Menu Keluar	*Black Box*
2	Permainan	Pergerakan pemain	*Black Box*
3	Menu Level	Pilih Level 1	*Black Box*
		Pilih Level 2	*Black Box*
		Pilih Level 3	*Black Box*
4	Menu Tentang	Pilih Menu Tentang	*Black Box*
		Pilih Kembali	*Black Box*
5	Menu Petunjuk	Pilih Menu Petunjuk	*Black Box*
		Pilih Kembali	*Black Box*

6	Menu Keluar	Pilih Menu Keluar	*Black Box*
7	Algoritma A* (A star)	Algoritma A*	*White Box*

4.2.1.2 Pengujian (Black Box)

Kasus dan hasil uji perangkat lunak yang sudah dibangun dengan menggunakan metode *black box* berdasarkan pengamatan yang dilakukan dan menghasilkan kesimpulan dari pengujian tersebut, diantaranya sebagai berikut :

A. Pengujian Permainan

Pengujian permainan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.3.

Tabel 4.3 Pengujian permainan

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Memijit *keypad* navigasi	Karakter utama bergerak sesuai pijitan keypad	Karakter utama dapat bergerak sesuai pijitan keypad	[√] Diterima [ ] Ditolak

B. Pengujian pilihan level

Pengujian pilihan level dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.4.

Tabel 4.4 Pengujian pilihan level

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih mulai permainan	Menampilkan pilihan level	Dapat menampilkan pilihan level	[√] Diterima [ ] Ditolak

pilih *level* 1	Menampilkan map *level* 1	Dapat menampilkan map *level* 1	[√] Diterima [ ] Ditolak
pilih *level* 2	Menampilkan map *level* 2	Dapat menampilkan map *level* 2	[√] Diterima [ ] Ditolak
pilih *level* 3	Menampilkan map *level* 3	Dapat menampilkan map *level* 3	[√] Diterima [ ] Ditolak

C. Pengujian Tampil Tentang

Pengujian tampilan tentang dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.5.

Tabel 4.5 Pengujian Tampil Tentang

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih pilihan tentang	Menampilkan informasi pembuat aplikasi permainan	Dapat menampilkan informasi pembuat aplikasi permainan	[√] Diterima [ ] Ditolak
Pilih kembali	Menampilkan menu utama	Dapat menampilkan menu utama	[√] Diterima [ ] Ditolak

D. Pengujian Tampil Petunjuk

Pengujian tampilan petunjuk dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.6.

Tabel 4.6 Pengujian Tampil Petunjuk

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih pilihan petunjuk	Menampilkan informasi petunjuk permainan	Dapat menampilkan informasi petunjuk permainan	[√] Diterima [ ] Ditolak
Menekan tombol kembali	Menampilkan menu utama	Dapat menampilkan menu utama	[√] Diterima [ ] Ditolak

E. Pengujian Keluar Aplikasi

Pengujian keluar aplikasi dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.7

Tabel 4.7 Pengujian keluar aplikasi

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih keluar	Keluar dari aplikasi	Dapat keluar dari aplikasi	[√] Diterima [ ] Ditolak

4.2.1.3 Pengujian (White Box)

Di bawah ini merupakan kasus algoritma A* (A *star*) untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *white box*, berdasarkan skenario pengujian aplikasi yang sudah dijabarkan.

Tabel 4.8 *Source code* Algoritma A*

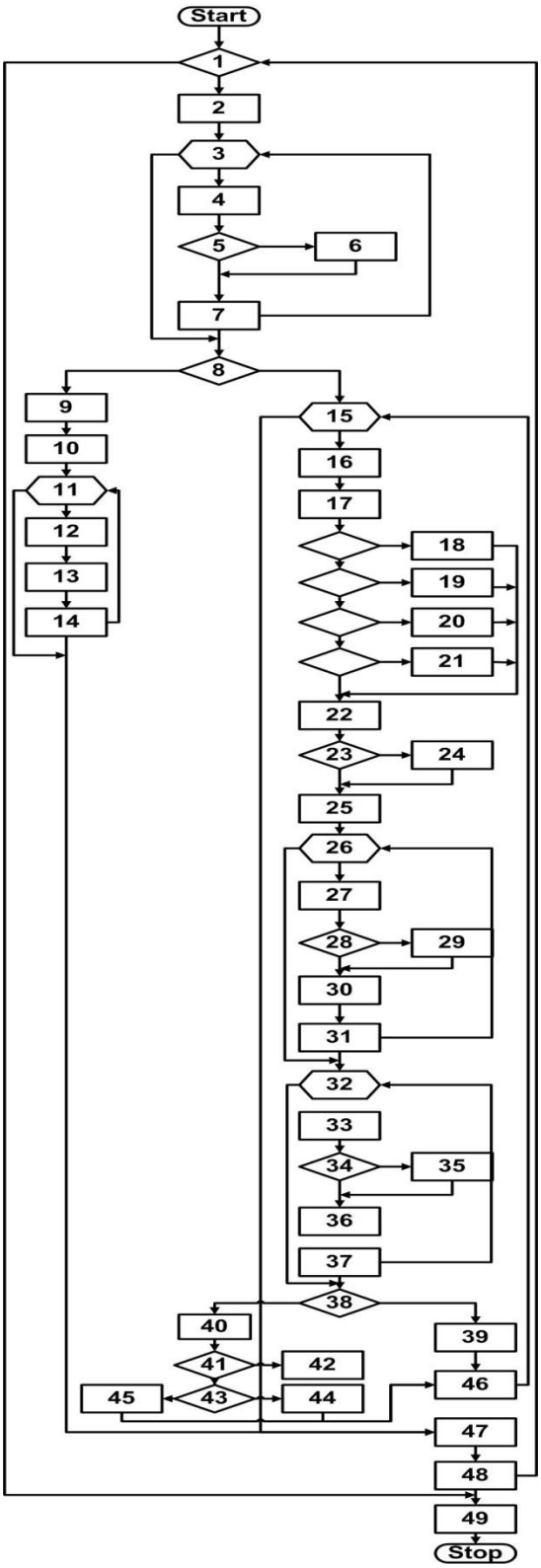
```
    while (!open.isEmpty())
1
    {
2     min =10000;
3     for (Enumeration e = open.keys() ; e.hasMoreElements() ; )
      {
4       ir = (Integer)e.nextElement();
5       if (ir.intValue() < min)
6         min = ir.intValue();
7     }
      Integer min2 = new Integer(min);
      ExtractNode = (Node)open.get(min2);
      open.remove(min2);
8     if((ExtractNode.getXLoc() <= (GoalNode.getXLoc() + 4))
       &&(ExtractNode.getXLoc() >= (GoalNode.getXLoc() - 4)))
      {
9       if((ExtractNode.getYLoc() <= (GoalNode.getYLoc() + 4)) &&
        (ExtractNode.getYLoc() >= (GoalNode.getYLoc() - 4)))
          {
10          Point[] punti = new
    Point[(ExtractNode.getCostFromStart())];
11          for (int i=0; i< punti.length; i++)
            {
12              punti[i]=new Point();
                punti[i].x = ExtractNode.getXLoc();
                punti[i].y = ExtractNode.getYLoc();
                ExtractNode = ExtractNode.getParent();
13          }
14        }
       }
       Else
```

```
    {
15   for (int i=0; i<4; i++)
     {
16    costoOpen = 0;
      costoClose = 0;
      inOpen = false;
      inClose = false;
      NewNode = new Node();
      NewNode.setXLoc(ExtractNode.getXLoc());
      NewNode.setYLoc(ExtractNode.getYLoc());
17    switch(i){
            case 0:
18            NewNode.setXLoc(-4);
              break;
            case 1:
19            NewNode.setXLoc(4);
              break;
            case 2:
20            NewNode.setYLoc(4);
              break;
            case 3:
21            NewNode.setYLoc(-4);
              break;
22          }
23    if (ostacoli[NewNode.getXLoc()][NewNode.getYLoc()] > 1)
      {
24          continue;
25    }
      cost = manhattanDistance(NewNode, GoalNode)
      NewNode.setCostFromStart(ExtractNode.getCostFromStart()+
```

```
    1)
          NewNode.setCostToGoal(cost)
          NewNode.setParent(ExtractNode)
          NewNode.setTotalCost()
          inOpen = false
          inClose = false
26        for (Enumeration e = open.elements() ;
    e.hasMoreElements() ; )
          {
27         n = (Node)e.nextElement();
28         if ( (NewNode.getXLoc() == n.getXLoc()) &&
           (NewNode.getYLoc() == n.getYLoc()) )
            {
29              costoOpen = n.getTotalCost();
                inOpen = true;
30           }
31         }
32        for (Enumeration e = close.elements() ;
    e.hasMoreElements() ; )
          {
33          n = (Node)e.nextElement();
34          if ( (NewNode.getXLoc() == n.getXLoc()) &&
            (NewNode.getYLoc() == n.getYLoc()) )
              {
35                costoClose = n.getTotalCost();
                  inClose = true;
36             }
37         }
38        if ((inOpen && NewNode.getTotalCost() >= costoOpen) ||
          (inClose && NewNode.getTotalCost() >= costoClose))
39         continue;
          Else
```

```
     {
40     if (inClose)
       {
41       ir = new Integer(costoClose);
         close.remove(ir);
         open.put(new
Integer(NewNode.getTotalCost()),NewNode);
42     }
43     if (inOpen)
       {
44       ir = new Integer(costoOpen);
         nodoDaModif = new Node();
         nodoDaModif = (Node)open.get(ir);
         open.remove(ir);
         nodoDaModif.setTotalCost(NewNode.getTotalCost());
          open.put(new Integer(nodoDaModif.getTotalCost()),
         nodoDaModif);
        }
        Else
        {
45        open.put(new
Integer(NewNode.getTotalCost()),NewNode);
        }
     }
46   }
   }
47   close.put(new
Integer(ExtractNode.getTotalCost()),ExtractNode);
48 }
49 return null;
```

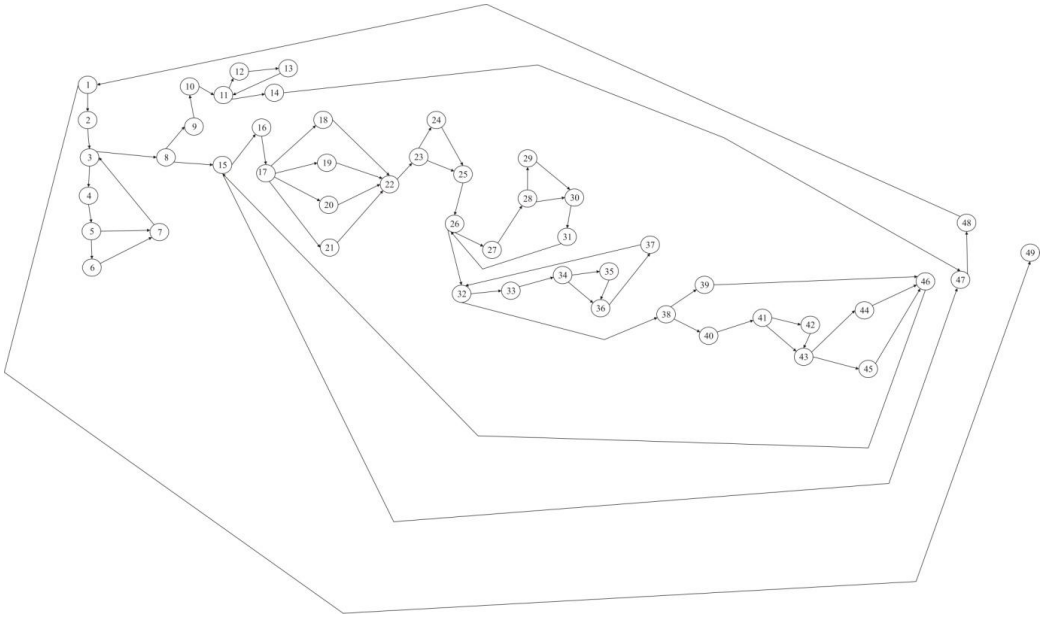
a. *Flowchart* White box A*



Gambar 4.6 *Flowchart* pengujian white box Algoritma A*

b. Flowgraph Algoritma A*

Pengujian dilakukan berdasarkan *source code* pada tabel 4.1 dan *flowchart* pada gambar , adapun *Flowgraph* pengujian pada Algoritma A* dapat dilihat pada gambar 4.7.



Gambar 4.7 *Flowgraph* pengujian algoritma A*

Keterangan :  = Menggambarkan kondisi

 = Menggambarkan aksi

c. Tahap Pengujian *Collision Detection* Peluru dengan Musuh

1. *Cyclomatic Complexity* V(G)

   Edge = 63, Node = 49

   V(G) = Edge – Node + 2

   V(G) = 63 – 49 + 2

   V(G) = 14+ 2

   V(G) = 16

2. *Independent Path*

   Path 1= 1-49

   Path 2= 1-2-3-4-5-6-7-3-8-9-10-11-12-13-11-14-47-48-1-49

Path 3= 1-2-3-4-5-7-3-8-9-10-11-12-13-11-14-47-48-1-49

Path 4= 1-2-3-4-5-6-7-3-8-15-47-48-1-49

Path 5= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-35-36-37-32-38-39-46-15-47-48-1-49

Path 6= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-35-36-37-32-38-40-41-42-43-44-46-15-47-48-1-49

Path 7= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-35-36-37-32-38-40-41-42-43-45-46-15-47-48-1-49

Path 8= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-35-36-37-32-38-40-41-43-45-46-15-47-48-1-49

Path 9= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-35-36-37-32-38-40-41-43-44-46-15-47-48-1-49

Path 10= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-40-41-43-44-46-15-47-48-1-49

Path 11= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-39-46-15-47-48-1-49

Path 12= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-40-41-42-43-44-46-15-47-48-1-49

Path 13= 1-2-3-4-5-6-7-3-8-15-16-17-18-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-40-41-43-44-45-46-15-47-48-1-49

Path 14= 1-2-3-4-5-6-7-3-8-15-16-17-19-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-40-41-43-44-45-46-15-47-48-1-49

Path 15= 1-2-3-4-5-6-7-3-8-15-16-17-20-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-40-41-43-44-45-46-15-47-48-1-49

Path 16= 1-2-3-4-5-6-7-3-8-15-16-17-21-22-23-24-25-26-27-28-29-30-31-26-32-33-34-36-37-32-38-40-41-43-44-45-46-15-47-48-1-49

3. *Predicate Node*

   V(G) = Jumlah node yang memiliki lebih dari 1 jalur + 1

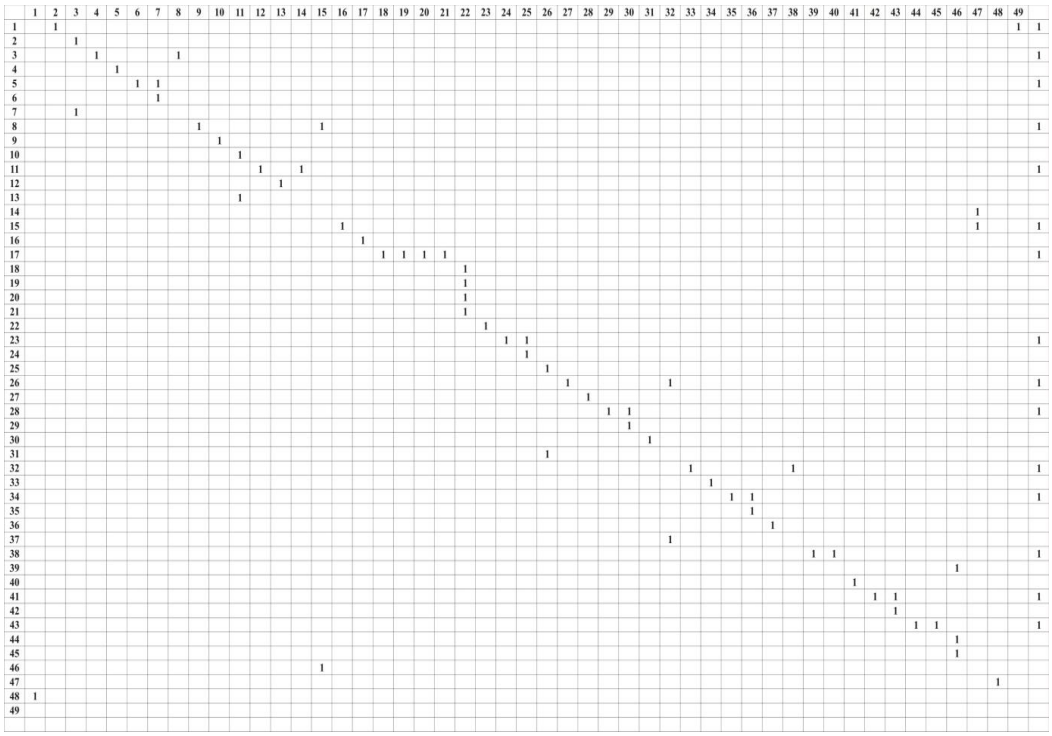
   V(G) = 15 + 1

   V(G) = 16

4. *Graph Matriks*

   Graph Matriks pengujian pada algoritma A* dapat dilihat pada tabel 4.6.

Tabel 4.9 Matriks algoritma A*



5. Kesimpulan

   Berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 17, maka dapat disimpulkan bahwa pada proses pencarian dengan algoritma A* (A *star*) berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.1.4 Kesimpulan Pengujian Alpha

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa sistem sudah berjalan seperti yang diharapkan dan secara fungsional sudah dapat menghasilkan keluaran yang diharapkan.

4.2.2 Pengujian Beta

Pengujian *beta* merupakan pengujian yang dilakukan secara objektif, dimana pengujian langsung melibatkan para pengguna aplikasi permainan secara langsung dengan menggunakan kuisioner mengenai kepuasan para pengguna terhadap aplikasi permainan yang dibangun. Pengujian dilakukan terhadap 30 responden, setiap responden diberikan 6 pernyataan untuk kemudian dilakukan penilaian dengan memilih 5 pilihan yang telah disediakan untuk setiap pernyataannya.

4.2.2.1 Kuesioner

Dari hasil kuesioner yang sudah dibagikan, didapat data-data yang kemudian diolah untuk mencari berapa besar presentase dan frekuensi dari responden (pelanggan) dengan menggunakan skala *Likert.* Skala *Likert* menggunakan bobot sebagai berikut :

a. Untuk jawaban Sangat Setuju diberikan bobot 5.

b. Untuk jawaban Setuju diberikan bobot 4.

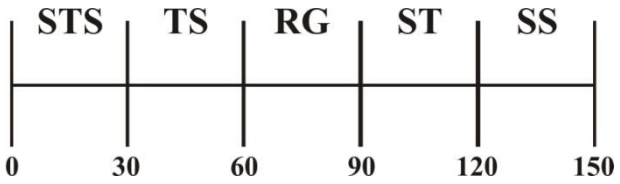
c. Untuk jawaban Ragu-Ragu diberikan bobot 3.

d. Untuk jawaban Tidak Setuju diberikan bobot 2.

e. Untuk jawaban Sangat Tidak Setuju diberikan bobot 1.

Data yang diperoleh dari pemberian kuesioner kepada responden dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan scoring setiap jawaban dari responden.Analisis yang dapat dilakukan yaitu sebagai berikut:

1. Jumlah skor ideal (kriterium) untuk seluruh item = 5 × jumlah responden
2. Rata-rata skor dalam persentase = (jumlah skor : jumlah skor ideal) × 100%
3. Jumlah skor ideal pada penelitian ini yaitu; Jumlah skor ideal = 5×30 = 150
4. Secara kontinum dapat digambarkan seperti berikut:



Gambar 4.8 Skala Kontinum

Garis diatas dapat dijelaskan pada tabel 4.10.

Tabel 4.10 Skor ideal

	Skor Ideal	Nilai	Keterangan
STS	30	1	Sangat Tidak Setuju
TS	60	2	Tidak Setuju
RG	90	3	Ragu-Ragu
ST	120	4	Setuju
SS	150	5	Sangat Setuju

Data yang didapat dari hasil kuisioner terhadap 30 orang responden adalah sebagai berikut :

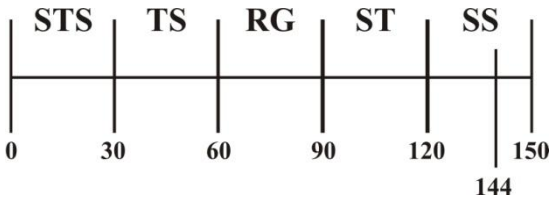
1. Cara pengoprasian dari mulai menu sampai menuju ke arena permainan pada aplikasi permainan *Crazy* becak mudah dimengerti.

   Hasil kuisioner untuk pernyataan ke-1 dapat dilihat pada tabel 4.11.

Tabel 4.11 Hasil kuisioner pernyataan ke-1

Pernyataan	**Jawaban**	**Skor**	**Responden**	**S**
1	Sangat Setuju	5	24	120
	Setuju	4	6	24
	Ragu-Ragu	3	0	0
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**144**

Setelah menghitung hasil kuesioner pernyataan 1, dapat disimpulkan bahwa penilaian terhadap pengoperasian aplikasi *game crazy* becak mudah tidaknya diperoleh rata-rata 144 maka pada garis kontinum berada pada daerah sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.8.



Gambar 4.9 Interpretasi Skor Pertanyaan ke-1

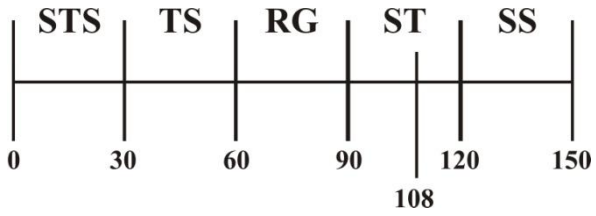
2. Tingkat kesulitan baik itu halangan dan musuh yang mengejar pemain dalam aplikasi permainan *crazy* becak tidak mudah untuk dilewati.

   Hasil kuisioner untuk pernyataan ke-2 dapat dilihat pada tabel 4.12.

Tabel 4.12 Hasil kuisioner pernyataan ke-2

Pernyataan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	1	5
	Setuju	4	16	64
	Ragu-Ragu	3	13	39
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**108**

Setelah menghitung hasil kuesioner pernyataan 2, dapat disimpulkan bahwa penilaian terhadap tingkat kesulitan aplikasi *game crazy* becak diperoleh rata-rata 108 maka pada garis kontinum berada pada daerah setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.9.



Gambar 4.10 Interpretasi Skor Pertanyaan ke-2

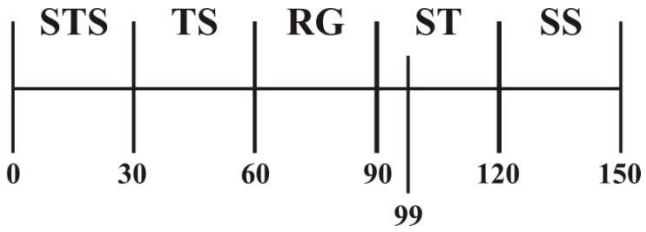
3. Musuh yang terdapat pada *game Crazy* becak sudah cerdas.

Hasil kuisioner untuk pernyataan ke-3 dapat dilihat pada tabel 4.13.

Tabel 4.13 Hasil kuisioner pernyataan ke-3

Pernyataan	Jawaban	Skor	Responden	S
3	Sangat Setuju	5	0	0
	Setuju	4	9	36
	Ragu-Ragu	3	21	63
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**99**

Setelah menghitung hasil kuesioner pernyataan 3, dapat disimpulkan bahwa penilaian terhadap cerdasnya musuh aplikasi *game crazy* becak mudah tidaknya diperoleh rata-rata 99 maka pada garis kontinum berada pada daerah setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.10.



Gambar 4.11 Interpretasi Skor Pertanyaan ke-3

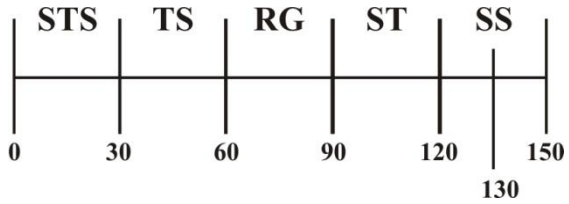
4. Gambar, animasi, tampilan dan informasi serta audio pada *game crazy* becak sudah baik dan mudah untuk dimengerti.

   Hasil kuisioner untuk pernyataan ke-4 dapat dilihat pada tabel 4.14.

Tabel 4.14 Hasil kuisioner pernyataan ke-4

Pernyataan	Jawaban	Skor	Responden	S
4	Sangat Setuju	5	11	55
	Setuju	4	18	72
	Ragu-Ragu	3	1	3
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**130**

Setelah menghitung hasil kuesioner pernyataan 4, dapat disimpulkan bahwa penilaian terhadap baik tidaknya dari gambar, animasi, tampilan dan informasi aplikasi *game crazy* becak diperoleh rata-rata 130 maka pada garis kontinum berada pada daerah sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.11.



Gambar 4.12 Interpretasi Skor Pertanyaan ke-4

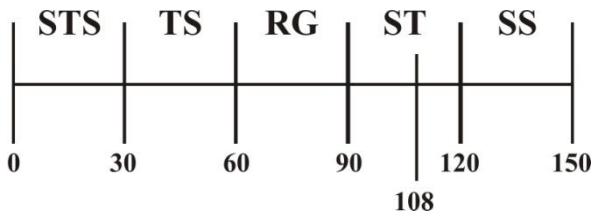
5. Aplikasi permainan *Crazy* becak menarik untuk dimainkan.

Hasil kuisioner untuk pernyataan ke-4 dapat dilihat pada tabel 4.15.

Tabel 4.15 Hasil kuisioner pernyataan ke-5

Pernyataan	Jawaban	Skor	Responden	S
5	Sangat Setuju	5	0	0
	Setuju	4	18	72
	Ragu-Ragu	3	12	36
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**108**

Setelah menghitung hasil kuesioner pernyataan 5, dapat disimpulkan bahwa penilaian terhadap menarik tidaknya aplikasi *game crazy* becak diperoleh rata-rata 108 maka pada garis kontinum berada pada daerah setuju, untuk lebih jelasnya dapat dapat dilihat pada gambar 4.12.



Gambar 4.13 Interpretasi Skor Pertanyaan ke-5

6. Karakter utama dalam aplikasi permainan *Crazy* becak adalah becak itu sendiri, setujukah anda jika becak bisa menjadi alat transportasi alternatif yang ramah lingkungan karena tidak menyebabkan polusi udara dan tidak menyebabkan kebisingan.

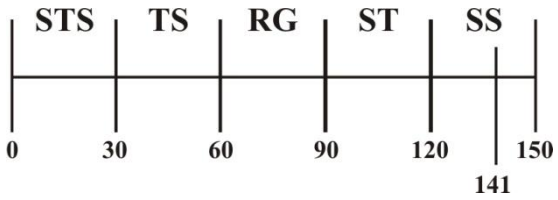
   Hasil kuisioner untuk pernyataan ke-4 dapat dilihat pada tabel 4.16.

Tabel 4.16 Hasil kuisioner pernyataan ke-6

Pernyataan	Jawaban	Skor	Responden	S
6	Sangat Setuju	5	21	105
	Setuju	4	9	36
	Ragu-Ragu	3	0	0
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**141**

Setelah menghitung hasil kuesioner pernyataan 6, dapat disimpulkan bahwa penilaian terhadap becak sebagai alat transportasi alternatif diperoleh rata-rata 141

maka pada garis kontinum berada pada daerah sangat setuju, untuk lebih jelasnya dapat dapat dilihat pada gambar 4.13.



Gambar 4.14 Interpretasi Skor Pertanyaan ke-6

4.2.2.2. Kesimpulan Pengujian Beta

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

1. Pengoperasian *game* mudah dimengerti.
2. Musuh sudah cukup cerdas.
3. Permainan cukup menarik untuk dimainkan.
4. Tingkat kesulitan baik itu halangan dan musuh yang mengejar pemain dalam aplikasi permainan *crazy* becak cukup tidak mudah untuk dilewati.