BAB IV
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi yang dibangun. Tahapan ini dilakukan setelah analisis dan perancangan selesai dilakukan dan selanjutnya akan diimplementasikan kedalam bahasa pemograman. Setelah implementasi maka dilakukan pengujian terhadap aplikasi. Aplikasi yang telah dibangun akan diimplementasikan untuk mengetahui apakah sistem tersebut dapat berjalan sesuai dengan tujuannya atau tidak.

4.1 Implementasi Sistem

Tahap implementasi sistem merupakan tahap penerjemahan perancangan berdasarkan hasil analisis ke dalam suatu bahasa pemrograman yang digunakan serta penerapan perangkat lunak yang dibangun pada lingkungan yang sesungguhnya, Tujuan implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem sehingga user dapat memberi masukan demi berkembangnya sistem yang telah dibangun dari simulasi penerapan algoritma *minimax* untuk menentukan langkah NPC pada permainan domino. Adapun pembahasan untuk implementasi sistem meliputi kebutuhan perangkat keras, perangkat lunak, implementasi antarmuka, dan implementasi pesan.

4.1.1 Implementasi Perangkat Keras

Kebutuhan perangkat keras dimaksudkan untuk mengetahui spesifikasi perangkat keras yang digunakan untuk menjalankan simulasi penerapan algoritma *minimax* untuk menentukan langkah NPC pada permainan domino ini. spesifikasi perangkat keras yang digunakan akan dijelaskan pada tabel 4.1.

Tabel 4. 1. Implementasi Perangkat Keras

No	Perangkat Keras	Spesifikasi
1	Processor	Processor dengan kecepatan 1.8 Ghz
2	Monitor	Monitor dengan resolusi 1024 x 768 pixel
3	Graphic Card	VGA 64 MB

No	Perangkat Keras	Spesifikasi
4	Harddisk	Free Space 500 MB
5	Memori	RAM 256 MB
6	Mouse	Standar
7	Speaker	Standar
8	Keyboard	Standar

4.1.2 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk membangun simulasi permainan domino dapat dilihat pada spesifikasi tabel 4.2 :

Tabel 4. 2 Implementasi Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Microsoft Windows XP	Digunakan sebagai sistem operasi
2	Adobe Flash CS5	Digunakan sebagai tool pembangun
3	Adobe Photoshop CS5	Digunakan sebagai tool desain
4	Adobe Flash Player 10	Digunakan sebagai tool *compiler*

4.1.3 Implementasi Simulasi

Simulasi *game* domino merupakan sebuah simulasi yang menggambarkan bagaimana algoritma *minimax* dengan menggunakan metode *Divide* and *Conquer* dapat menentukan langkah NPC dalam mengambil keputusan pada permainan domino

4.1.4 Implementasi Class

Class merupakan deskripsi sekelompok objek dari property(atribut), sifat (operasi), relasi antar objek dan semantik yang umum.

1. *Class Home*

   Nama Class : home.as

   Keterangan : *Class* yang digunakan untuk mengulang permainann menu utama pada permainan domino

2. *Class Latar*

   Nama *Class* : latar.as

   Keterangan : *Class* yang digunakan untuk mengulang permainan domino.

3. *Class main domino*

   Nama *Class* : mainDomino.as

   Keterangan : *Class* yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

4. *Class mulai permainan*

   Nama : mulaiPermainan.as

   Keterangan : *Class* yang memiliki fungsi untuk memulai permainan domino.

4.1.5 Implementasi Antarmuka

Implementasi antarmuka dilakukan dengan setiap tampilan program yang dibangun dan pengkodeannya dalam bentuk file program. Implementasi antarmuka pada simulasi permainan domino akan dijelaskan pada tabel :

Tabel 4. 3. Implementasi Antarmuka *Game*

No	Antarmuka	Keterangan
1	Menu Utama	Merupakan tampilan awal permainan yang menampilkan menu utama permainan
2	Menu Mulai	Merupakan tampilan permainan yang menampilkan *Gameplay* dari simulasi permainan domino
3	Menu About	Merupakan tampilan antarmuka menu *about* yang berisi tentang profil pembuat simulasi permainan domino

4.1.6 Implementasi Pesan

Implementasi pesan merupakan bagian tampilan pada *game* yang membutuhkan penggunanya untuk melakukan suatu keputusan. Implementasi pesan pada simulasi permainan domino akan dijelaskan pada tabel 4.4.

Tabel 4. 4. Implementasi Pesan

No	Pesan	Keterangan
1	Keluar Permainan	Merupakan sebuah pesan yang memberikan pengguna suatu pilihan untuk melakukan keputusan apakah ingin meninggalkan permainan atau tidak

4.2 Pengujian Sistem

Pengujian sistem merupakan hal terpenting yang dilakukan untuk menemukan kekurangan atau kesalahan pada perangkat lunak yang diuji. Pengujian dimaksudkan untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan perancangan perangkat lunak tersebut. Adapun pengujian yang akan dilakukan adalah pengujian alpha.

4.2.1 Pengujian Alpha

Pengujian alpha merupakan pengujian program yang dilakukan oleh pembuat game ataupun orang - orang yang terlibat didalamnya. pengujian alpha hanya untuk sirkulasi internal dan masalah *(error)* atau ketidaklengkapan yang terdapat dalam game. Proses pengujian terhadap aplikasi akan menggunakan *black box testing* dan *white box testing*.

Pengujian yang menggunakan teknik *white box testing* lebih menitikberatkan pada struktur internal (*source code*) program, dimana 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 pada program.

4.2.1.1 Skenario Pengujian

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada permainan Domino. Adapun skenario pengujian yang ada pada permainan Domino dapat dilihat pada tabel 4.5.

Tabel 4. 5. Skenario Pengujian

No	Komponen Yang Diuji	Butir Uji	Jenis Uji
1	Menu Utama	Tampilkan Menu Utama	*Black Box*
		Tampilkan Menu Mulai	*Black Box*
		Tampilkan Menu Tentang	*Black Box*
		Tampilkan Menu Keluar	*Black Box*
2	Menu Mulai	Tampilkan Halaman Utama Permaian	*Black Box*
		Menggerakan Karakter Untuk Melempar Kartu	*Black Box*
		Menekan Tombol Reset	*Black Box*
		Menekan Tombol Keluar	*Black Box*
3	Menu Tentang	Tampilkan Menu Tentang	*Black Box*
		Menekan Tombol Keluar	*Black Box*
4	Menu Keluar	Memilih Menu Keluar	*Blax Box*
5	Algoritma *Minimax*	Algoritma *Minimax* Pada Permainan Domino	*White Box*

4.2.1.2 Kasus Dan Hasil Pengujian (*Black Box Testing*)

Berikut ini adalah kasus untuk menguji perangkat lunak yang telah dibangun denga menggunakan metode *black box* akan dijelaskan pada tabel 4.6.

Tabel 4. 6. Kasus Dan Hasil Uji

Kasus dan Hasil Uji				
No	Kasus/Diuji	Skenario Uji	Hasil Yang Diharapkan	Hasl Pengujian
1	Menu Utama	Tampilkan Menu Utama	Halaman Menu Utama Ditampilkan	[√] Berhasil [ ] Tidak Berhasil

		Tampilkan Menu Mulai	Menampilkan Arena Permainan	[√] Berhasil [ ] Tidak Berhasil
		Tampilkan Menu Tentang	Menampilkan Menu Tentang Yang Berisi Profil Pembuat *Game*	[√] Berhasil [ ] Tidak Berhasil
		Menekan Tombol Keluar	Keluar Dari Halaman Menu Utama Permaianan	[√] Berhasil [ ] Tidak Berhasil
2	Mulai Permainan	Tampilkan Halaman Utama Permaian	Menampilkan Halaman Utama Permainan Domino	[√] Berhasil [ ] Tidak Berhasil
		Menggerakan Karakter Untuk Melempar Kartu	Kartu Dilempar	[√] Berhasil [ ] Tidak Berhasil
		Menekan Tombol Reset	Permainan Direset	[√] Berhasil [ ] Tidak Berhasil
		Menekan Tombol Keluar	Keluar Dari Permainan Dan	[√] Berhasil [ ] Tidak Berhasil
			Kembali Ke Halaman Menu Utama	[√] Berhasil [ ] Tidak Berhasil
3	Menu Tentang	Menampilkan Menu Tentang	Tampilkan Menu Tentang	[√] Berhasil [ ] Tidak Berhasil
		Menekan Tombol Keluar	Keluar Dari Menu Tentang Dan Kembali Ke Halaman Menu Utama	[√] Berhasil [ ] Tidak Berhasil
4	Menu Keluar	Memilih Menu Keluar	Keluar Dari Halaman Menu Utama Permaianan	[√] Berhasil [ ] Tidak Berhasil

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

Pengujian yang digunakan untuk menguji simulasi permainan domino ini adalah metode pengujian *white box*. Pengujian *white box* yaitu menguji perangkat lunak kode program apakah algoritma *minimax* dengan metode pencarian *Divide and Conquere* mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan.

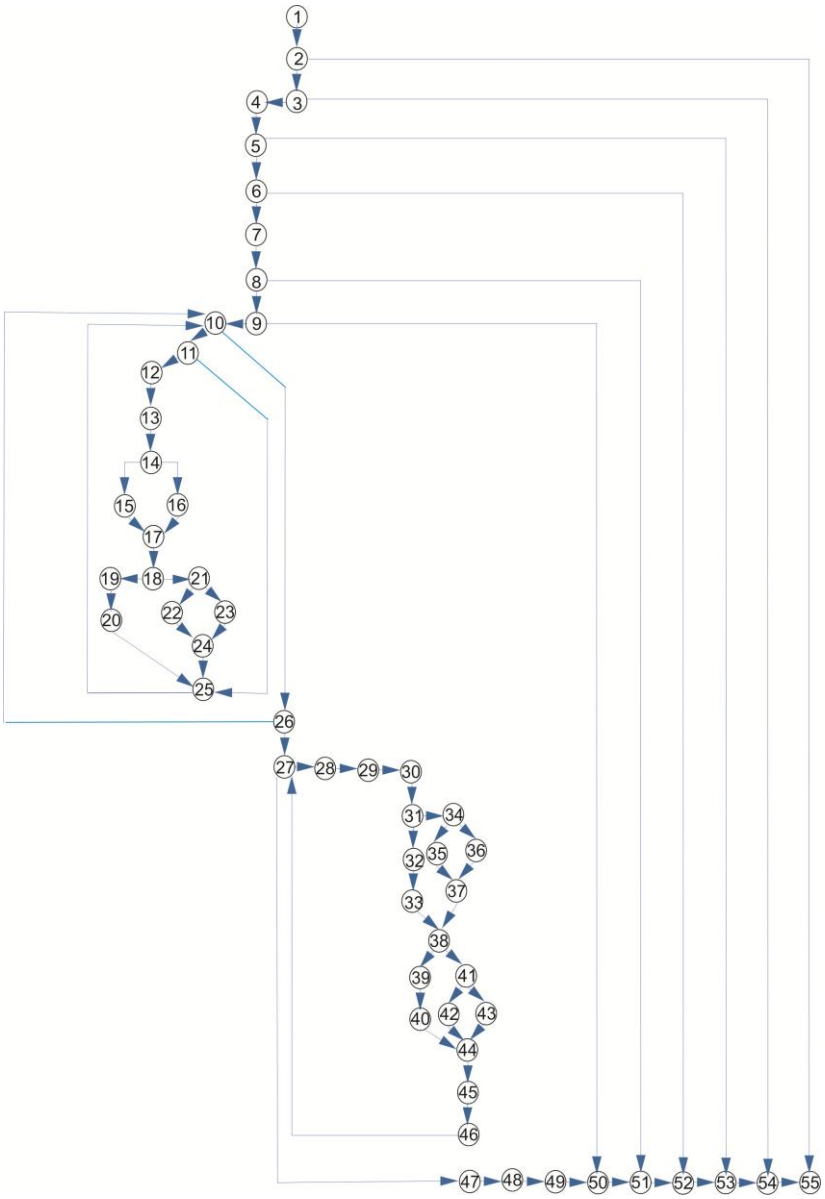
a. Pengubahan *pseudocode* menjadi menjadi flowgraph.

```
1.  solusi = [];
2.  if (solusi[0] == "solusi1")
    {
    trace("Solusi 1");
3.  for (var l:int = 0; l < tempJmlhIdx.length; l++)
        {
4.  l1.push(l);
5.  if (tempJmlhIdx.length == 7)
   {
6.  if (l == (tempJmlhIdx.length)-1)
    {
7.  var solve1:int = (l1[l]+l1[l-l])/2;// hasilnya 3
8.  for (var l2:int = 0; l2 < l1.length; l2++)
    {
9.  if (l2 == solve1)
    {
    //Blok Kiri [2]
10. for (var l3:int = 0; l3 < l1.length; l3++)
    {
11. if (l3 == 2)
    {
12. var solve21:int = (l1[l3-l3]+l1[l3])/2;//hasilnya 1
13. tempSolveL.push(solve21-1,solve21,l3);//hasilna [0,1,2];
    trace(tempSolveL);
14. if (int(p1[tempSolveL[0]].angkaKartuP.text) == kiriArr[0] ||
    int(p1[tempSolveL[0]].angkaKartuP.text) == kananArr[0] ||
    int(p1[tempSolveL[0]].angkaKartuQ.text) == kiriArr[0] ||
    int(p1[tempSolveL[0]].angkaKartuQ.text) == kananArr[0])
    {
15. tempCombine.push(tempSolveL[0]);
    }//end
    else{
16. sol.push(true);
17. }//endif
18. if (int(p1[Math.max(tempSolveL[1],tempSolveL[2])].angkaKartuP.text) ==
    kiriArr[0] || int(p1[Math.max(tempSolveL[1],tempSolveL[2])].angkaKartuP.text)
    == kananArr[0] ||
    int(p1[Math.max(tempSolveL[1],tempSolveL[2])].angkaKartuQ.text) ==
    kiriArr[0] || int(p1[Math.max(tempSolveL[1],tempSolveL[2])].angkaKartuQ.text)
    == kananArr[0])
19. {
    tempCombine.push(Math.max(tempSolveL[1],tempSolveL[2]));
20. }//endif
21. else if (int(p1[Math.min(tempSolveL[1],tempSolveL[2])].angkaKartuP.text) ==
    kiriArr[0] || int(p1[Math.min(tempSolveL[1],tempSolveL[2])].angkaKartuP.text)
    == kananArr[0] ||
```

```
int(p1[Math.min(tempSolveL[1],tempSolveL[2])].angkaKartuQ.text) ==
kiriArr[0] || int(p1[Math.min(tempSolveL[1],tempSolveL[2])].angkaKartuQ.text)
== kananArr[0])
{
22. tempCombine.push(Math.min(tempSolveL[1],tempSolveL[2]));
}//end
else{
23. sol.push(true);
24. }//endif
25. }//endif
26. }//end For //Blok Kanan [3]
27. for (var l4:int = l2; l4 < l1.length; l4++)
28. if (l4 == l1.length - 1)
{
29. var solve22:int = Math.floor((l1[l4]+l1[l4-3])/2);//hasilnya 4
30. tempSolveR.push(solve22-1,solve22,solve22+1,solve22+2); //hasilna [3,4,5,6];
trace(tempSolveR);
31. if (int(p1[Math.max(tempSolveR[0],tempSolveR[1])].angkaKartuP.text) ==
kiriArr[0] || int(p1[Math.max(tempSolveR[0],tempSolveR[1])].angkaKartuP.text)
== kananArr[0] ||
int(p1[Math.max(tempSolveR[0],tempSolveR[1])].angkaKartuQ.text) ==
kiriArr[0] || int(p1[Math.max(tempSolveR[0],tempSolveR[1])].angkaKartuQ.text)
== kananArr[0])
{
32. tempCombine.push(Math.max(tempSolveR[0],tempSolveR[1]));
33. }//endif
34. else if (int(p1[Math.min(tempSolveR[0],tempSolveR[1])].angkaKartuP.text) ==
kiriArr[0] || int(p1[Math.min(tempSolveR[0],tempSolveR[1])].angkaKartuP.text)
== kananArr[0] ||
int(p1[Math.min(tempSolveR[0],tempSolveR[1])].angkaKartuQ.text) ==
kiriArr[0] || int(p1[Math.min(tempSolveR[0],tempSolveR[1])].angkaKartuQ.text)
== kananArr[0])
{
35. tempCombine.push(Math.min(tempSolveR[0],tempSolveR[1]));
}//end Max
else{
36. sol.push(true);
37. }//endif
38. if (int(p1[Math.max(tempSolveR[2],tempSolveR[3])].angkaKartuP.text) ==
kiriArr[0] || int(p1[Math.max(tempSolveR[2],tempSolveR[3])].angkaKartuP.text)
== kananArr[0] ||
int(p1[Math.max(tempSolveR[2],tempSolveR[3])].angkaKartuQ.text) ==
kiriArr[0] || int(p1[Math.max(tempSolveR[2],tempSolveR[3])].angkaKartuQ.text)
== kananArr[0])
{
39. tempCombine.push(Math.max(tempSolveR[2],tempSolveR[3]));
40. }//endif
41. else if (int(p1[Math.min(tempSolveR[2],tempSolveR[3])].angkaKartuP.text) ==
kiriArr[0] || int(p1[Math.min(tempSolveR[2],tempSolveR[3])].angkaKartuP.text)
== kananArr[0] ||
int(p1[Math.min(tempSolveR[2],tempSolveR[3])].angkaKartuQ.text) ==
kiriArr[0] || int(p1[Math.min(tempSolveR[2],tempSolveR[3])].angkaKartuQ.text)
== kananArr[0])
{
42. tempCombine.push(Math.min(tempSolveR[2],tempSolveR[3]));
}//end Max
else{
43. sol.push(true);
```

```
44.  }//endif
45.  }//endif
46.  }//end For
47.  if (sol.length == 4)
     {
48.  solusi.splice(1,1,"solusi2");
49.  }//endif
50.  }//endif
51.  }//endfor
52.  }//end if
53.  }//end if
54.  }//endfor
55.  }//endif
```

b. Flowgraph *Minimax* menggunakan metode *Divide and Conquer*



Gambar 4. 1. Flowgraph *Minimax* menggunakan metode *Divide and Conquer*

Keterangan :

◯ = Menggambarkan kondisi.

→ = Menggambarkan aksi.

c. *Cyclomatic complexity* dari *edge* ke *node* V(G)

E = 70, N = 55

V(G) = E – N + 2

V(G) = 70 – 55 + 2

V(G) = 17

Keterangan :

E = Jumlah Aksi

N = Jumlah Kondisi

d. Jalur *Independent Path*

Jalur 1 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-17-18-19-20-25-10-26-27-28-29-30-31-32-33-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 2 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-17-18-19-20-25-10-26-27-28-29-30-31-32-33-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 3 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-17-18-21-22-24-25-10-26-27-28-29-30-31-32-33-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 4 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-17-18-21-23-24-25-10-26-27-28-29-30-31-32-33-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 5 : 1-2-3-4-5-6-7-8-9-10-11-25-10-26-27-28-29-30-31-32-33-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 6 : 1-2-3-4-5-6-7-8-9-10-26-27-28-29-30-31-32-33-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 7 : 1-2-3-4-5-6-7-8-9-10-26-27-28-29-30-31-34-35-37-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 8 : 1-2-3-4-5-6-7-8-9-10-26-27-28-29-30-31-34-36-37-38-39-40-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 9 : 1-2-3-4-5-6-7-8-9-10-26-27-28-29-30-31-34-36-37-38-41-42-44-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 10 : 1-2-3-4-5-6-7-8-9-10-26-27-28-29-30-31-34-35-37-38-39-40-44-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 11 : 1-2-3-4-5-6-7-8-9-10-26-27-28-29-30-31-34-35-37-38-39-40-44-45-46-27-47-48-49-50-51-52-53-54-55

Jalur 12 : 1-2-3-4-5-6-7-8-9-50-51-52-53-54-55

Jalur 13 : 1-2-3-4-5-6-7-8-51-52-53-54-55

Jalur 14 : 1-2-3-4-5-6-52-53-54-55

Jalur 15 : 1-2-3-4-5-53-54-55

Jalur 16 : 1-2-3-54-55

Jalur 17 : 1-2-55

e. *Graph Matrix*

Tabel 4. 7. Implementasi grap matrix

	Connect To Node																																																			E
N	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	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	
1		1																																																	1	1
2			1																																																	0
3				1																																													1		1
4					1																																													0
5						1																																										1			1
6							1																																											0
7								1																																											0
8									1																																							1			1
9										1																																					1					1
10											1														1																												1
11												1												1																													1
12													1																																							0
13														1																																						0
14															1	1																																					1
15																	1																																			0
16																	1																																			0
17																		1																																		0
18																			1		1																															1
19																				1																																0
20																									1																											0
21																						1	1																													1
22																								1																													0
23																								1																													0
24																									1																												0
25										1																																										0
26										1																																										0

27																												1																			1									1
28																													1																											0
29																														1																										0
30																															1																									0
31																																1		1																						1
32																																	1																							0
33																																						1																		0
34																																			1	1																				1
35																																					1																			0
36																																					1																			0
37																																						1																		0
38																																							1		1															1
39																																								1																0
40																																												1												0
41																																											1													0
42																																												1												0
43																																												1												0
44																																													1											0
45																																														1										0
46																											1																													0
47																																																1								0
48																																																	1							0
49																																																		1						0
50																																																			1					0
51																																																				1				0
52																																																					1			0
53																																																						1		0
54																																																							1	0
CC																																																								16+1=17

Keterangan :

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

P = 7

V(G) = P + 1

V(G) = 16 + 1

V(G) = 17

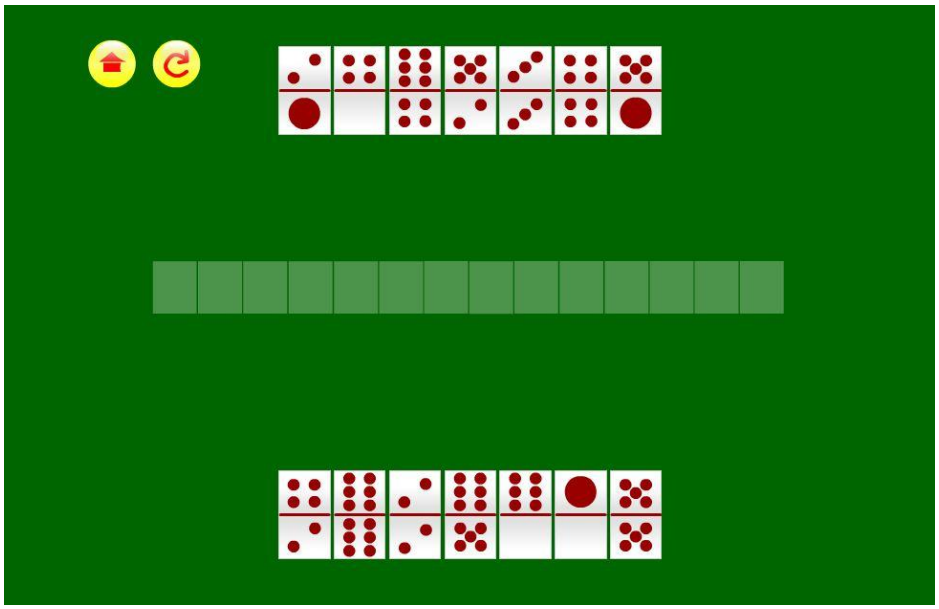
Keterangan :

P = Simpul Predikat

Dari hasil pengujian *white box* dapat disimpulkan bahwa jumlah E pada *graph matrix* sama dengan jumlah *Cyclomatic complexity* V(G) yaitu 17, sehingga tidak terjadi *error* pada setiap baris *source code.*

4.2.1.3 Kasus Pengujian Game Domino

Ini merupakan contoh kasus pengujian pada permainan domino yang terdiri dari kartu pemain, kartu player, dan kartu di papan. Dapat dilihat pada gambar 4.2.



Gambar 4. 2. Contoh kasus pengujian

1. Giliran Pemain Melempar Kartu

   1.1 Kartu Pemain

   Kartu awal pemain dapat dilihat pada tabel 4.8.

Tabel 4. 8. Kartu Pemain

No	Kiri	Kanan	Jumlah	MAX
1	4	2	6	
2	6	6	12	✓
3	2	2	4	
4	6	5	11	
5	6	0	6	
6	1	0	1	
7	5	5	10	

a. Pemain memulai permainan dengan memilih nilai MAX untuk dilempar ke papan permainan.

Tabel 4. 9. Pemain Melempar Kartu Nilai Max Ke papan

7	6	5	4	3	2	1	1	2	3	4	5	6	7
Kiri							**Kanan**						
						6	6						

b. Kartu di papan

Tabel 4. 10. Kartu di papan

No	Kiri	Kanan
1	-	-

c. Kondisi setelah pemain memilih nilai MAX yang dilempar ke papan

Tabel 4. 11. Kondisi kartu yang dilempar pemain ke papan

No	Kiri	Kanan	Jumlah	MAX
1	4	2	6	
2	6	6	12	✓
3	2	2	4	
4	6	5	11	
5	6	0	6	
6	1	0	1	
7	5	5	10	

d. Papan kartu telah terisi nilai kiri dan kanan dari kartu pemain

Tabel 4. 12 Papan Kartu Yang Telah Diisi dari kartu pemain

7	6	5	4	3	2	1	1	2	3	4	5	6	7
Kiri							**Kanan**						
						6	6						

e. Di papan setelah pemain memilih kartu

Tabel 4. 13. Kondisi papan setelah pemain memilih kartu

No	Kiri	Kanan
1	6	6

2. Giliran Player Melempar Kartu

2.1 Kartu Player

a. Cek kartu di papan untuk menentukan kartu yang akan dilempar

Tabel 4. 14. Pencocokan Kartu

Kartu Di Player				
No	Kiri	Kanan	Jumlah	MAX
1	2	1	3	
2	4	0	4	
3	6	0	6	
4	5	2	7	
5	3	3	6	
6	4	4	8	✓
7	5	1	5	

→

Kartu Di papan		
No	Kiri	Kanan
1	6	6

b. Hasil kartu yang telah dicek

Tabel 4. 15. Hasil Pencocokan Kartu

Kartu Di Player				
No	Kiri	Kanan	Jumlah	MAX
1	2	1	3	
2	4	0	4	
3	6	0	6	
4	5	2	7	
5	3	3	6	

→

Kartu Di papan		
No	Kiri	Kanan
1	6	6

6	4	4	8	✓				
7	5	1	5					

c. Papan Kartu telah terisi oleh nilai kanan dari player, karena nilai kiri dari kartu player sama dengan nilai kiri dari kartu di papan.

Tabel 4. 16. Kondisi kartu di papan setelah terisi

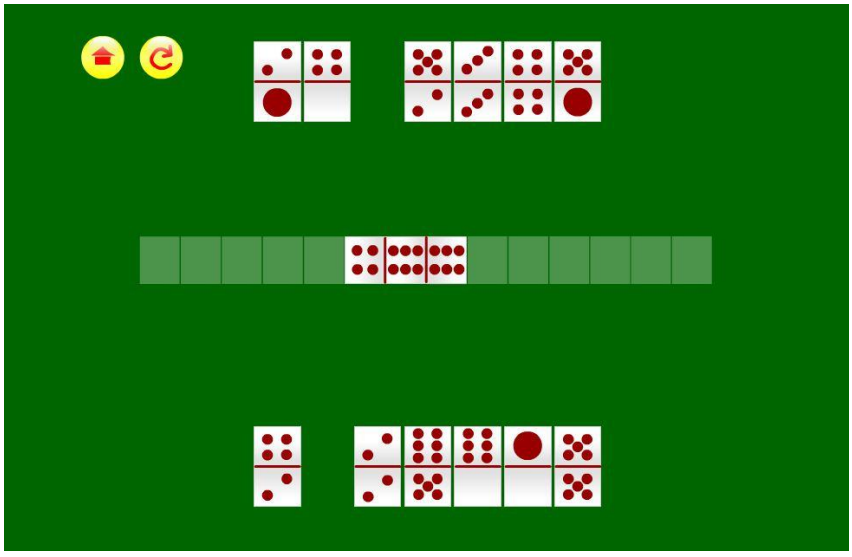
	6	6
↓		
0	6	6

d. Kartu di papan setelah player memilih kartu

Tabel 4. 17. Kartu di papan

No	Kiri	Kanan
1	6	6
2	6	-

Maka hasil dari kasus pengujian permainan domino dapat dilihat pada gambar 4.3.



Gambar 4. 3. Kartu di papan

4.2.1.4 Kesimpulan Pengujian Black Box dan White box

Hasil pengujian *black-box* dengan pengujian diatas dapat ditarik kesimpulan bahwa dalam *game* Domino tidak terdapat kesalahan proses dan secara fungsional mengeluarkan hasil yang sesuai dengan yang diharapkan.

Dari hasil pengamatan dan pengujian *white-box* dan *black-box.* Pengujian *white-box* dapat disimpulkan bahwa penentuan langkah NPC dalam mengambil keputusan sudah sesuai dengan yang diharapkan.