BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi. Tahap ini merupakan tahap di mana 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 yang digunakan untuk melakukan implementasi dan pengujian game ini yaitu sebagai berikut :

Tabel 4.1 Implementasi perangkat keras

No	PerangkatKeras	Keterangan
1	Prosesor	Kecepatan 2.4 GHz
2	Monitor	Monitor 17'', resolusi 1024x768
3	VGA On-board	Kecepatan 512 MB
4	Optical Drive DVD-ROM	Standard
5	Keyboard	Standard
6	Mouse	Standard
7	Memori	1GB

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk mengimplementasikan perangkat lunak diantara lain sebagai berikut :

Tabel 4.2 Implementasi perangkat lunak

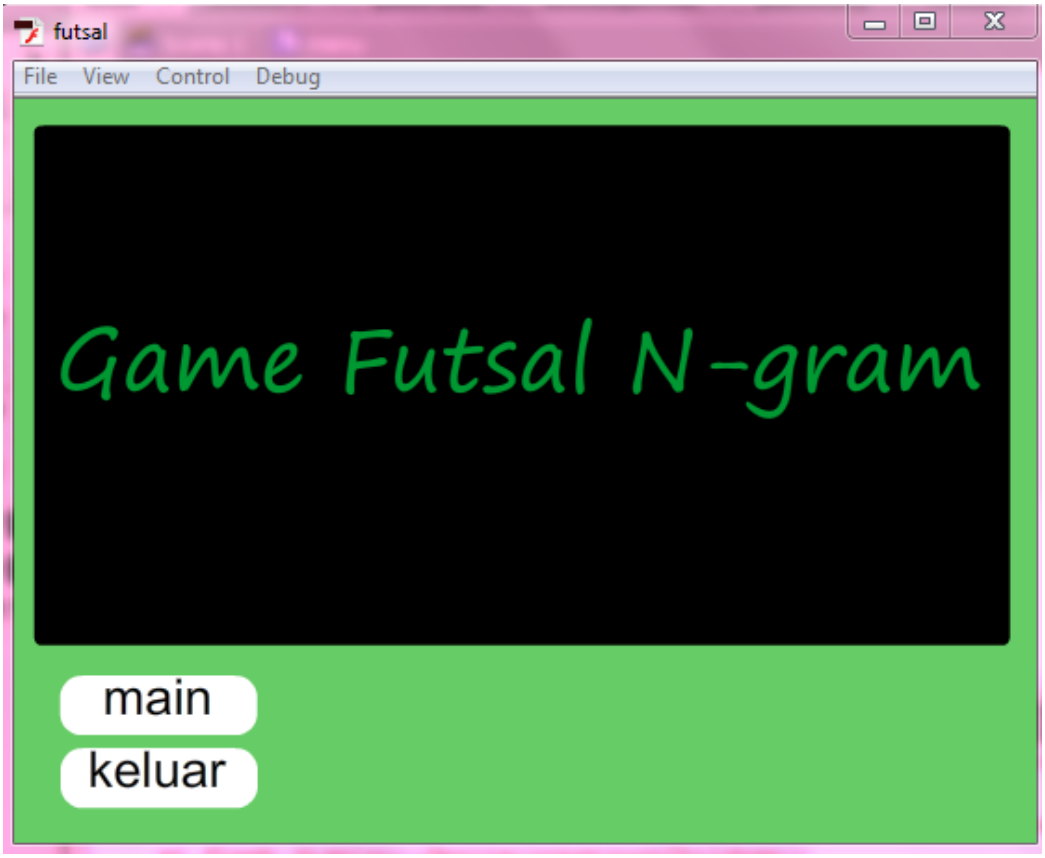
No	PerangkatLunak	Keterangan
1	SistemOperasi	*Windows* 7
2	*Tool*s Program	*Adobe Flash* CS5

4.1.3 Implementasi Antarmuka

Implementasi antar muka adalah bagian yang menunjuk bentuk tampilan setiap antar muka pada aplikasi yang sudah dibangun. Pada bagian ini akan ditampilkan implementasi anatrmuka menu utama dan menu main.

4.1.3.1 Implementasi Antarmuka Menu utama

Berikut adalah implementasi antarmuka menu utama yang ditunjukan oleh gambar 4.1.



Gambar 4.1 Antarmuka Menu Utama

4.1.3.2 Implementasi Antarmuka Menu main

Berikut adalah implementasi antarmuka menu main yang ditunjukan oleh gambar 4.2.



Gambar 4.2 Antarmuka Menu Main

4.2 Pengujian Sistem

Dalam melakukan pengujian terhadap aplikasi permainan futsal dilakukan dengan dua macam 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*.

4.2.1 Skenario Pengujian

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada game futsal. Skenario pengujian dari aplikasi permainan ini dapat dilihat pada tabel berikut.

Tabel 4.3 Skenario pengujian

Pengujian	Skenario Pengujian	Jenis Pengujian
Menu Utama	klik menu main	Blackbox
	klik menu keluar	Blackbox
Menu Main	klik tombol serang 1 untuk serang 1	Blackbox
	klik tombol serang 2 untuk serang 2	Blackbox
	klik tombol serang 3 untuk serang 3	Blackbox

	Klik kiri tombol bertahan 1 untuk bertahan 1	Blackbox
	Klik kiri tombol bertahan 2 untuk bertahan 2	Blackbox
	Klik kiri tombol bertahan 3 untuk bertahan 3	Blackbox
Metode N-Gram	Prediksi formasi	Whitebox

4.2.2 Pengujian Blackbox

Tabel berikut adalah hasil pengujian blackbox yang dilakukan sesuai skenario pengujian yang telah dirancang sebelumnya :

Tabel 4.4 Pengujian Blackbox

Menampilkan menu utama		
Masukan	Hasil yang diharapkan	Hasil pengujian
Memilih menu main	Menampilkan menu main	[√]diterima [ ]ditolak
Memilih menu keluar	Keluar dari permainan	[√]diterima [ ]ditolak
Menampilkan menu main		
klik tombol serang 1	Menjalankan formasi serang 1	[√]diterima [ ]ditolak
klik tombol serang 2	Menjalankan formasi serang 1	[√]diterima [ ]ditolak
klik tombol serang 3	Menjalankan formasi serang 1	[√]diterima [ ]ditolak
klik tombol bertahan 1	Menjalankan formasi bertahan 1	[√]diterima [ ]ditolak
klik tombol bertahan 2	Menjalankan formasi bertahan 2	[√]diterima [ ]ditolak
klik tombol bertahan 3	Menjalankan formasi bertahan 3	[√]diterima [ ]ditolak

4.2.3 Pengujian *Whitebox*

Di bawah ini merupakan kasus metode N-Gram untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan *whitebox*, berdasarkan skenario pengujian aplikasi yang sudah dijawabarkan.

a. Metode N-Gram

```
1   public class nGramPredictor

    // Menyimpan data frekuensi
2   private var data:Array = new Array();
3   private var nLength:int;
4   private var maxDataNum:int;

    // Menyimpan ukuran window + 1
5   public function nGramPredictor(nLength:int = 3, maxDataNum:int = 10){

    // Mengambil data dan menyimpannya ke array yang nGramList
6   var dataLength:int = data.length - (nLength - 1);

7   for (var dataIndex:int = 0; dataIndex < dataLength; dataIndex++){
```

```
    // Menghapus duplikat & menyimpan ke array baru
8   var tempArray:Array = Basic.getUniqueArray(nGramList, true);

    // Buat KeyData baru dari tempArray dan menyimpannya ke array yang
    nGramFreq
9   var tempArrayLength:int = tempArray.length;

10  for (var tempIndex:int = 0; tempIndex < tempArrayLength; tempIndex++){
11  nGramFreq.push(new KeyData(tempArray[tempIndex], 0));
12  }

    // Menghitung frekuensi setiap tombol
13  var nGramFreqLength:int = nGramFreq.length;
14  var nGramListLength:int = nGramList.length;

15  for (var i:int = 0; i < nGramFreqLength; i++){
16  for (var j:int = 0; j < nGramListLength; j++){

17  public function getNextAction():*{
18  var nGramArray:Array = new Array();
19  var actionArray:Array = new Array();
20  var bestAction:KeyData;
21  var lastData:Array;

    // Mendapatkan pola data terakhir
22  lastData = data.slice(data.length - (nLength - 1));

    // Pencocokan pola insert ke actionArray
23  var nGramLength:int = nGramArray.length;

24  for (var i:int = 0; i < nGramLength; i++){

    // Mendapatkan pola yang paling sering dari actionArray
25  for (var j:int = 0; j < actionArray.length; j++){

26  if (bestAction == null){
27  bestAction = actionArray[j];
28  }
30  else{
31  if (bestAction.getCount() < actionArray[j].getCount()){
32  bestAction = actionArray[j];
33  }
34  else if (bestAction.getCount() == actionArray[j].getCount()){
35  var rand:int = Basic.generateRandomSign();

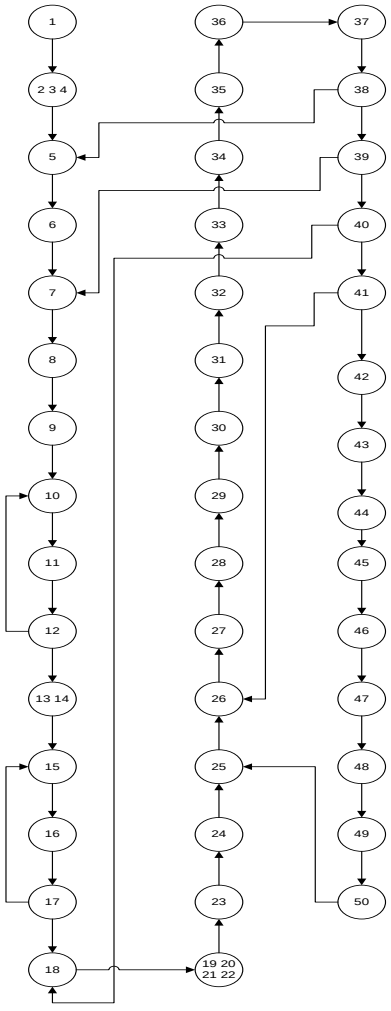
36  if (rand < 0){
37  bestAction = actionArray[j];
38  }
39  }
40  }
41  }

    //kembali ke aksi prediksi akhir
42  var predictedAction:Array = new Array();
```

```
43  if (bestAction != null){
44  predictedAction  =  bestAction.getKey().slice(bestAction.getKey().length
    - 1);
45  }
46  else{
47  predictedAction[0] = null;
48  }
49  return predictedAction[0];
50  }
```

Gambar 4.3 *Pseudocode* Metode N-Gram

b. Flowgraph



Gambar 4.4 *Flowgraph* Metode N-Gram

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

E (jumlah busur) = 50, N (jumlah simpul) = 44

V(G) = E – N + 2

V(G) = 50 – 44 + 2

V(G) = 8

d. Independent Path

Path 1 = 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

Path 2 = 1-2-3-4-5-6-7-8-9-10-11-12-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44

Path 3 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44

Path 4 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44

Path 5 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44

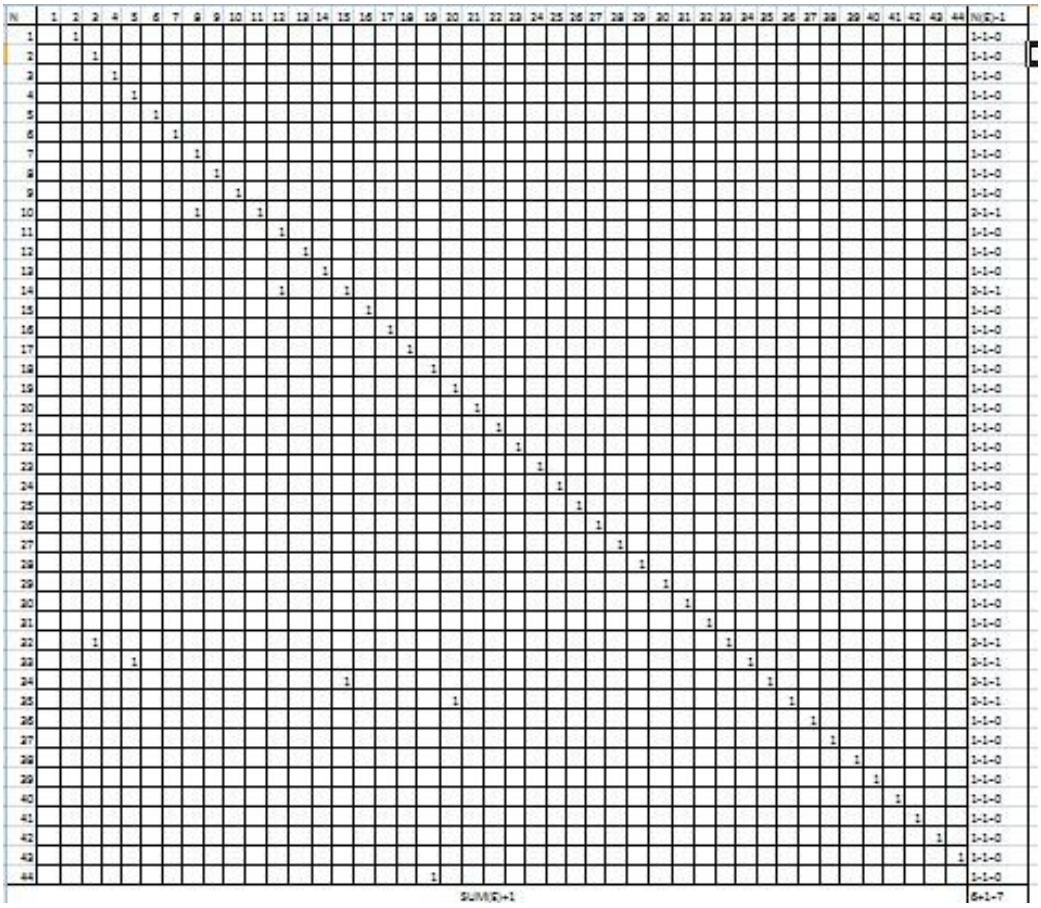
Path 6 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44

Path 7 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-26-27-28-29-30-31-32-34-35-36-37-38-39-40-41-42-43-44

Path 8 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44

e. Graph Matrices

Setelah *flow graph* diberi nomor baru, lalu dibuat tabel *graph matrice* seperti pada gambar 4.5.



Gambar 4.5 *Graph Matrices*

f. Hasil pengujian metode N-Gram

Tabel 4.5 Hasil Pengujian Metode N-Gram

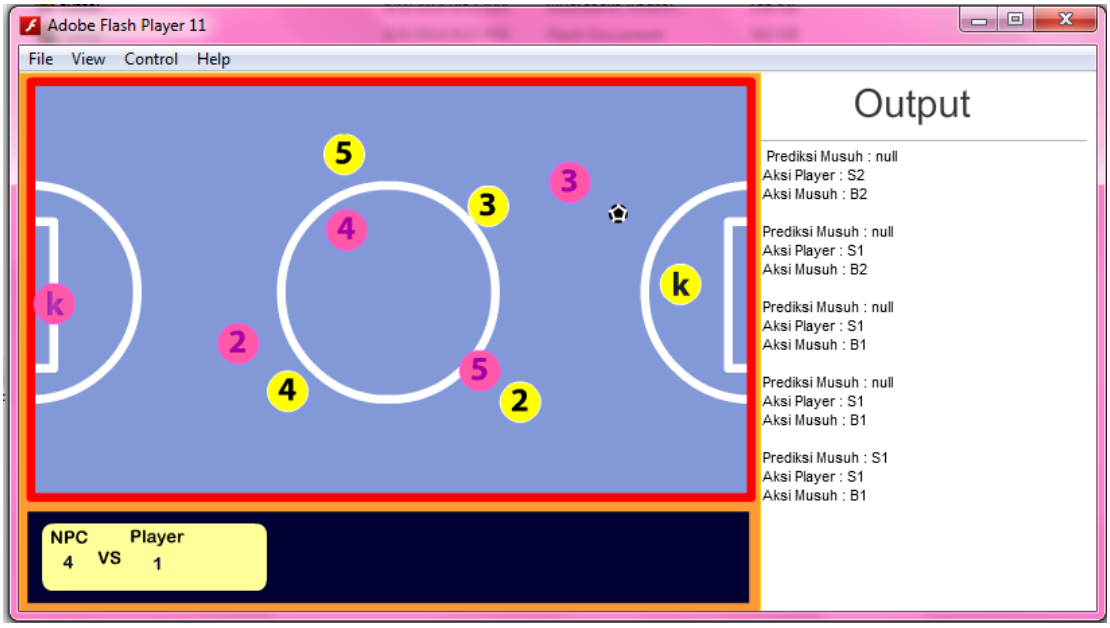
No	Kasus uji	Yang diharapkan	Hasil yang dicapai	Keterangan
1	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44	Melakukan proses memulai prediksi N-gram.	Prediksi N-Gram telah terbentuk di dalam array.	[√] Berhasil [ ] Tidak Berhasil
2	1-2-3-4-5-6-7-8-9-10-11-12-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-	Mengambil data dan menimpannya ke dalam *array*.	Data tersimpan dan masuk ke dalam *array*.	[√] Berhasil [ ] Tidak Berhasil

	41-42-43-44			
3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44	Membentuk pola dan menghitung frekuensi.	Tercipta pola berupa keydata dan menghitung kemunculan pola.	[√] Berhasil [ ] Tidak Berhasil
4	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44	Menentukan *window size* yang selalu bernilai n-1	*Window size* terbentuk bernilai 3.	[√] Berhasil [ ] Tidak Berhasil
5	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44	Menghapus duplikat dan menyimpannya ke *array* baru.	*Array* baru terbentuk *array* lama terhapus.	[√] Berhasil [ ] Tidak Berhasil
6	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44	Mendapatkan pola data atau *keydata* terakhir	*Keydat*a terakhir sudah ditentukan.	[√] Berhasil [ ] Tidak Berhasil
7	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-26-27-28-29-30-31-32-34-35-36-37-38-39-40-41-42-43-44	Menghitung pola data yang sering keluar sesuai *keydata* terakhir.	*Keydata* yang sering keluar sesuai pola terakhir ditemukan.	[√] Berhasil [ ] Tidak Berhasil
8	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44	Kembali ke prediksi akhir atau terciptanya prediksi akhir.	Terciptanya prediksi akhir N-Gram.	[√] Berhasil [ ] Tidak Berhasil

4.2.4 Pengujian Metode

Uji performansi metode N-Gram pada penelitian ini mempunyai 3 parameter formasi prediksi antara lain formasi prediksi serang 1, serang 2 dan serang 3.

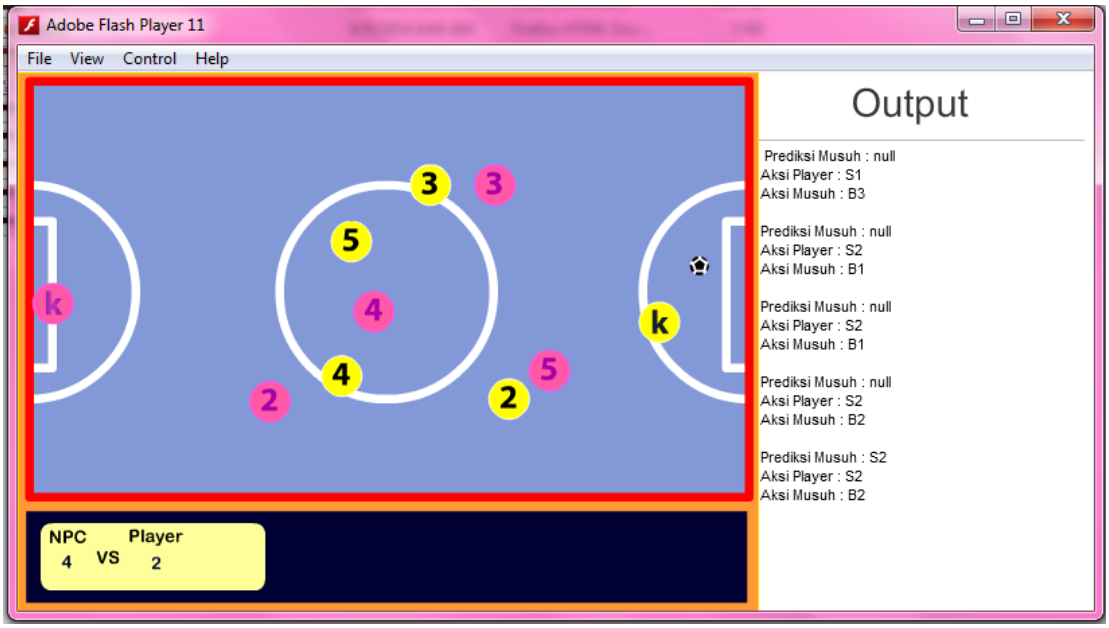
1. Prediksi formasi serang 1



Gambar 4.6 Prediksi formasi serang 1.

Pada gambar 4.6 dapat dilihat bahwa NPC baru bisa memprediksi formasi serang 1 setelah *player* melakukan 4 kali formasi yaitu S2S1S1S1 dan terbentuk frekuensi tertinggi yaitu S1S1S1.

2. Prediksi formasi serang 2



Gambar 4.7 Prediksi formasi serang 2.

Pada gambar 4.7 dapat dilihat bahwa NPC baru bisa memprediksi formasi serang 2 setelah *player* melakukan 4 kali formasi yaitu S1S2S2S2 dan terbentuk frekuensi tertinggi yaitu S2S2S2.

3. Prediksi formasi serang 3



Gambar 4.8 Prediksi formasi serang 3.

Pada gambar 4.8 dapat dilihat bahwa NPC baru bisa memprediksi formasi serang 3 setelah *player* melakukan 4 kali formasi yaitu S1S3S3S3 dan terbentuk frekuensi tertinggi yaitu S3S3S3.

Untuk memastikan metode berjalan dengan baik maka akan dilakukan uji coba dengan *player* melakukan formasi yang sudah ditentukan sebelumnya. formasi yang akan dilakukan *player* adalah sebanyak 20 gerakan. Berikut ini gerakan *player* dan aksi yang dilakukan NPC seperti ditunjukkan pada tabel 4.6.

Tabel 4.6 Hasil Uji Coba Metode N-Gram

No	Aksi Player	Aksi NPC
1	S1	Random
2	S2	Random
3	S1	Random
4	S1	Random
5	S1	Random
6	S1	Prediksi
7	S2	Random
8	S1	Prediksi
9	S1	Prediksi
10	S3	Random
11	S2	Random
12	S1	Random

13	S3	Random
14	S2	Prediksi
15	S1	Prediksi
16	S3	Random
17	S1	Random
18	S3	Random
19	S2	Prediksi
20	S1	Prediksi

Tabel yang diberi warna kuning adalah gerakan player yg terprediksi oleh NPC. Dari tabel dapat dianalisa bahwa ketika *player* melakukan formasi ke-1 sampai ke-5 *NPC* melakukan gerakan secara *random* karena belum tercipta *Key* S1S1S1. Pada formasi *player* ke 6 baru tercipta *Key* sehingga ketika *player* melakukan formasi S1 maka *NPC* dapat memprediksi secara tepat. Ketika pada formasi ke-7 *player* dimasukkan formasi lain *NPC* tidak terprediksi karena *NPC* masih memprediksi yang sama seperti sebelumnya karena frekuensi tertinggi masih S1S1S1. dari percobaan 20 formasi player pertama NPC dapat memprediksi 6 gerakan dengan tepat. Ini dikarenakan *key* yang tercipta masih sedikit. Ketika sudah terbentuk banyak *Key* maka *NPC* lebih banyak memprediksi dengan tepat gerakan lawan seperti ditunjukkan pada tabel 1 yaitu dari 6 prediksi yang tepat 5 diantaranya terdapat pada antara gerakan ke-8 sampai gerakan ke-20 dengan *player* mengeluarkan formasi yang random berurutan.