BAB 4
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi dan pengujian terhadap *soccer client* yang dikembangkan dan aplikasi yang dibangun untuk menguji algoritma yang diterapkan pada *soccer client*, yaitu *soccer client* Uva Trilearn dan aplikasi yang digunakan untuk menguji algoritma yaitu ABsoccer. Hasil perancangan pada tahap sebelumnya kemudian diimplementasikan kedalam bahasa pemograman. Setelah tahap implementasi selesai, maka dilakukan tahap pengujian terhadap *soccer client* yang dikembangkan dan aplikasi pengujian algoritma.

4.1 Implementasi

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

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk menjalankan simulator *RoboCup Soccer* dengan spesifikasi sebagai berikut:

1. Processor AMD A8-6410 APU 2.0 Ghz
2. RAM 4 GB
3. VGA 4 GB
4. Harddisk 500 GB
5. Monitor
6. Mouse dan Keyboard

4.1.2 Sistem Operasi dan File Library

RoboCup Soccer Simulator dapat dijalankan pada semua sistem operasi yang berbasis Linux, seperti *Fedora, Redhat, Debian, Ubuntu, Blankon,* dan masih banyak lagi. Selain pada sistem operasi yang berbasis linux, *RoboCup Soccer Simulator* juga bisa dijalankan diatas sistem operasi

Windows. Saat ini sudah tersedia beberapa versi *Soccer Server, Soccer Monitor dan Log Player* untuk sistem operasi *Windows*. Pada penelitian ini penulis menggunakan sistem operasi Linux *Ubuntu* versi 12.0.4. Dengan alasan Linux *Ubuntu* dapat dengan mudah digunakan dan juga untuk kebutuhan Library *RoboCup Soccer Simulator* dapat dengan mudah diinstall-kan. Setelah linux *Ubuntu* terinstall, senjautnya adalah menginstall-kan *library-library* yang diperlukan untuk menjalankan *RoboCup Soccer Simulator,* diantaranya adalah :

1. *Build-essential*
2. *Libtool*
3. *Libboost-dev*
4. *Libboost-filesystem-dev*
5. *Flex*
6. *Bison*
7. *Libx11-dev*
8. *Libxpm-dev*
9. *qt4-dev-tools*
10. *librcrc*

Untuk menginstall-kan *library* tersebut dibutuhkan koneksi internet. Setelah terhubung dengan internet, buka *Terminal,* ketikkan perintah *sudo apt-get install build-essential* untuk menginstall library *build-essential.* Perintah yang sama juga dilakukan untuk menginstallkan *library-library* yang lainnya. Setelah semua *library* terinstall, selanjutnya adalah menginstall *RoboCup Soccer Simulator.* Sedangkan untuk aplikasi pengujian dibagun pada system operasi windows dengan menggunakan *tools* visual studio.

4.1.3 Instalasi RoboCup Soccer Simulator

Untuk menginstall *RoboCup Soccer Simulator* dibutuhkan file installan komponen *RoboCup Soccer Simulator.* Adapun file yang dibutuhkan adalah *rcssserver, rcssmonitor, rcsslogplayer.* Pada penelitian ini komponen

RoboCup Soccer Simulator yang digunakan adalah *rcssserver* versi 15.0.1, *rcssmonitor* versi 15.0.0, dan *rcsslogplayer* versi 15.0.0. file-file tersebut dapat di *download* dari website http://sourceforge.net/projects/sserver/. Setelah file komponen *RoboCup Soccer Simulator* di *download*, selanjutnya adalah menginstalkan file-file tersebut melalui terminal. Berikut adalah perintah untuk menginstalkan-nya :

1. Instalasi *rcssserver*:

   Pertama ekstrak terlebih dahulu file *rcssserver*, kemudian buka terminal dan arahkan ke folder *rcssserver* untuk perintah instalasinya seperti berikut ini :

   ```
   $ cd ~/rcssserver-15.0.1
   $ ./configure
   $ make
   $ sudo make install
   ```

2. Instalasi *rcssmonitor*:

   Pertama ekstrak terlebih dahulu file *rcssmonitor*, kemudian buka terminal dan arahkan ke folder *rcssmonitor* untuk perintah instalasinya seperti berikut ini :

   ```
   $ cd ~/rcssmonitor-15.0.0
   $ ./configure
   $ make
   $ sudo make install
   ```

3. Instalasi *soccerwindow*:

   Pertama ekstrak terlebih dahulu file *soccerwindow*, kemudian buka terminal dan arahkan ke folder *soccerwindow* untuk perintah instalasinya seperti berikut ini :

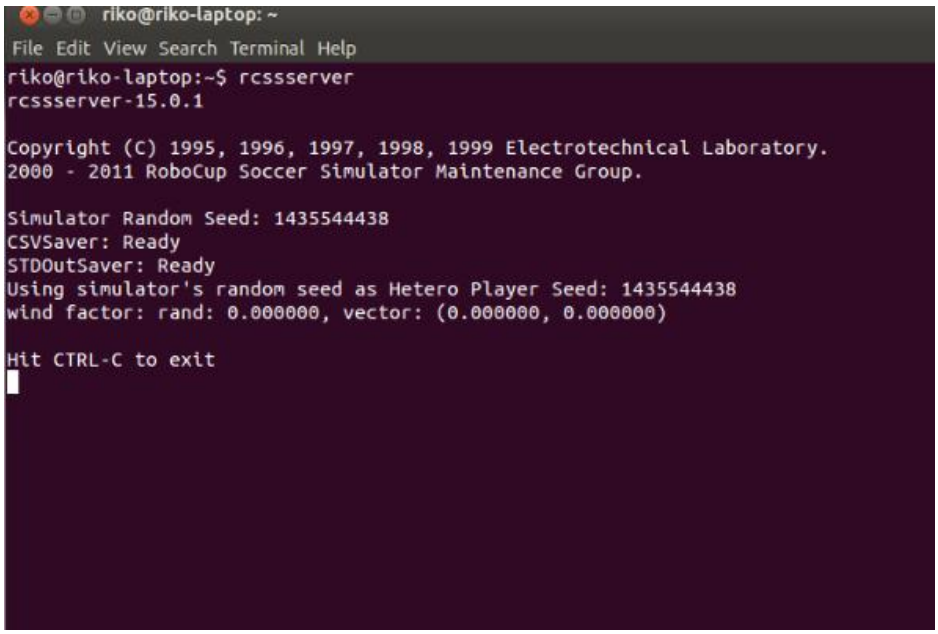
   ```
   $ cd ~/soccerwindow2-4.1.1
   $ ./configure
   $ make
   $ sudo make install
   ```

4. Instalasi *rcsslogplayer*:

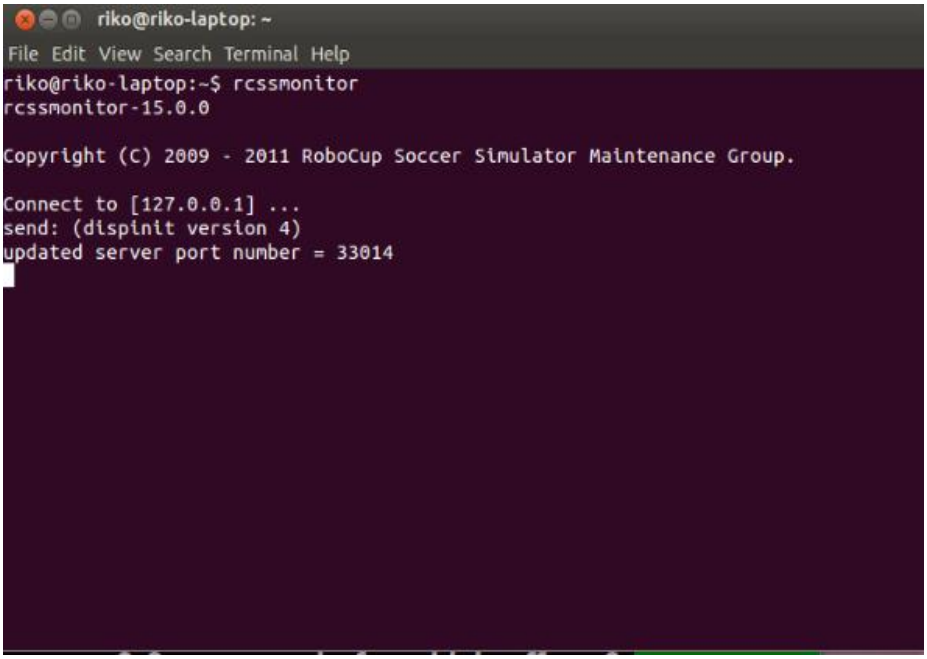
   Pertama ekstrak terlebih dahulu file *rcsslogplayer*, kemudian buka terminal dan arahkan ke folder *rcsslogplayer* untuk perintah instalasinya seperti berikut ini :

```
$ cd ~/rcsslogplayer-15.0.0
$ ./configure
$ make
$ sudo make install
```

Setelah instalasi komponen-komponen yang dibutuhkan untuk menjalankan *soccer simulator* sudah selesai, untuk menjalankan *soccer server*, *soccer monitor* atau *soccerwindow* dengan mengetikkan perintah: $rcssserver, $soccerwindow, $rcssmonitor pada terminal. Untuk lebih jelasnya dapat dilihat pada gambar 4.1 yang menunjukkan *soccer server* yang sedang berjalan, gambar 4.2 menunjukkan *soccer monitor* yang sedang berjalan, gambar 4.3 menunjukkan *soccer monitor* terhubung dengan *soccer server*.



Gambar 4.1 Menjalankan Soccer Server



Gambar 4.2 Menjalankan Soccer Monitor



Gambar 4.3 Soccer Monitor dan Soccer Server terhubung

4.1.4 Simulasi Pertandingan

Untuk menjalankan simulasi pertandingan pertama jalankan terlebih dahulu *soccer server* dan *soccer monitor*. Selanjutnya buka terminal

dan arahkan ke folder tempat penyimpanan *source code* atau *binary* tim yang akan digunakan untuk simulasi pertandingan. Ketikkan perintah **./start.sh** pada terminal dan lakukan perintah yang sama untuk tim yang akan digunakan untuk simulasi pertandingan, setelah kedua tim yang akan melakukkan simulasi telah terlihat pada monitor seperti yang ditujukkan pada gambar 4.4, simulasi dapat dilakukkan dengan menekan tombol *kick off* (ctrl + k) pada *soccer monitor* atau *soccerwindow*. Pertandingan dilakukan dalam komputer yang sama yaitu *local server* (*local host*). Waktu yang dibutuhkan untuk simulasi pertandingan dalam satu babak adalah 3000 *time cycle*. Dalam setiap satu babak pertandingan sepakbola biasanya terjadi pergantian daerah lapangan, namun dalam simulasi pertandingan ini tidak terjadi pergantian lapangan antara kedua tim. Simulasi pertandingan ini diatur oleh wasit otomatis dalam *server*. Setelah pertandingan selesai simulasi dapat dihentikan dengan menekan tombol *quit*. *Soccer server* akan secara otomatis membuat dua *file* secara bersamaan pada saat menekan tombol *quit*. *File* tersebut adalah *file* yang berekstension **.rcg* dan **.rcl*. *File *.rcg* merupakan *file* rekaman simulasi pertandingan yang dapat diputar ulang kembali dengan menggunakan *rcsslogplayer*. Sementara *file *.rcl* merupakan *file* yang berisi *log* aktivitas setiap tim selama simulasi pertandingan.



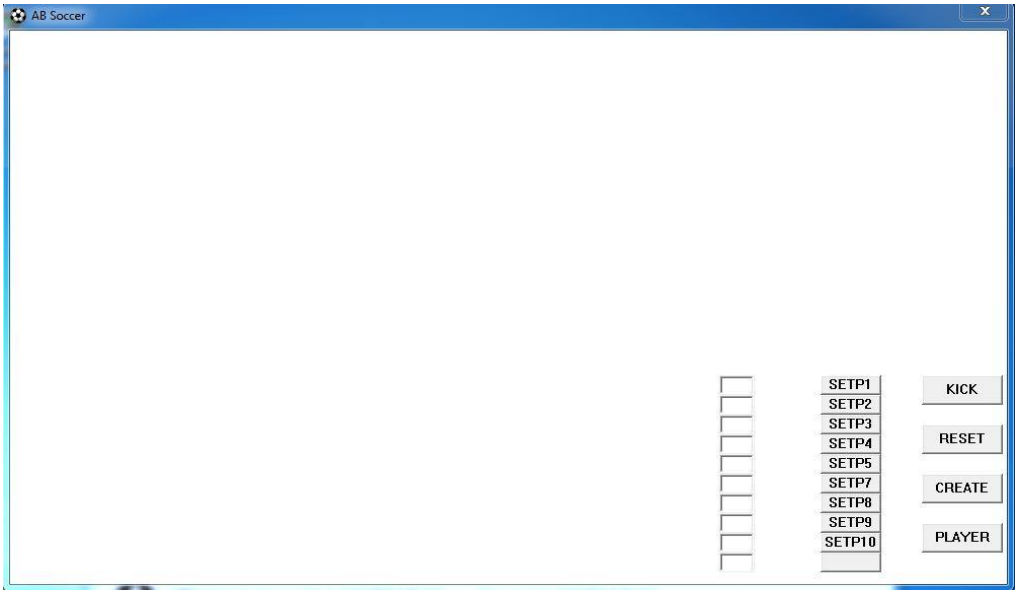
Gambar 4.4 Tampilan simulasi pertandingan di Soccerwindow

4.1.5 Implementasi Antarmuka

Implementasi antarmuka pada aplikasi pengujian algoritma *Alpha Beta* Pruning adalah sebagai berikut:

1. Tampilan Awal.

   Antarmuka tampilan awal merupakan tampilan aplikasi pada saat pertama kali dijalankan. Tampilan awal aplikasi dapat dilihat pada gambar 4.5.



Gambar 4.5 Tampilan Awal Aplikasi

2. Tampilan agent beserta bola
   Antarmuka tampilan *agent* dan bola merupakan tampilan antarmuka setelah pemain menekan tombol create dan player. Tampilan *agent* dan bola dapat dilihat pada gambar 4.6.



Gambar 4.6 Tampilan Agent dan bola

4.1 Pengujian dan Evaluasi

Pada bagian ini penulis akan melakukan percobaan skenario untuk melihat kinerja algoritma yang diterapkan pada Soccer client UvA Trilearn. Sebagai mana

algoritma yang diterapkan bertujuan untuk melakukan pencarian terhadap posisi Teammate untuk diberikan bola atau passing bola terhadap posisi Teammate tersebut.

4.2.1 Pengujian Aplikasi

Pada tahap ini, pegujian dilakukan terhadap fungsionalitas pada aplikasi yang dibangun yaitu ABsoccer. Apakah sudah sesuai dengan kebutuhan yang di inginkan.

4.2.1.1 Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi pengujian algoritma yaitu ABsoccer. Skenario pengujian dari aplikasi dapat dilihat pada tabel 4.1.

Tabel 4.1 Skenario pengujian Aplikasi

No	Komponen yang diuji	Skenario pengujian	Jenis pengujian
1	Create lapangan	Membuat lapangan	*Black Box*
2	Create player	Membuat player	*Black Box*
3	Kick	Menendang bola	*Black Box*
4	Reset	Mengembalikan ke posisi awal	*Black Box*
5	ABpuring	Mencari nilai terbesar	*White Box*

4.2.1.2 Kasus dan Pengujian

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

1. Pengujian Create lapangan

   Pengujian Create lapangan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.2.

Tabel 4.2 Pengujian create lapangan

Kasus dan Hasil Uji			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pemain membuat lapangan	Lapangan tercipta	*Player* dapat membuat lapangan	[√] Diterima [ ] Ditolak

2. Pengujian Player

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

Tabel 4.3 Pengujian Create Player

Kasus dan Hasil Uji			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pemain membuat *agent*	*agent* tercipta	*Pemain* dapat membuat *agent*	[√] Diterima [ ] Ditolak

3. Pengujian Kick

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

Tabel 4.4 Pengujian Kick

Kasus dan Hasil Uji			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pemain memulai percobaan	Bola dioper oleh pemain	*Pemain* dapat mengoper bola	[√] Diterima [ ] Ditolak

4. Pengujian Reset

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

Tabel 4.5 Pengujian reset

Kasus dan Hasil Uji			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pemain mengembalikan bola dan *agent* keposisi awal	Posisi bola dan *agent* pindah keposisi awal	*Pemain* dapat mengembalikan posisi bola dan *agent* ke posisi awal	[√] Diterima [ ] Ditolak

4.2.1.3 Kasus dan Pengujian (*White box*)

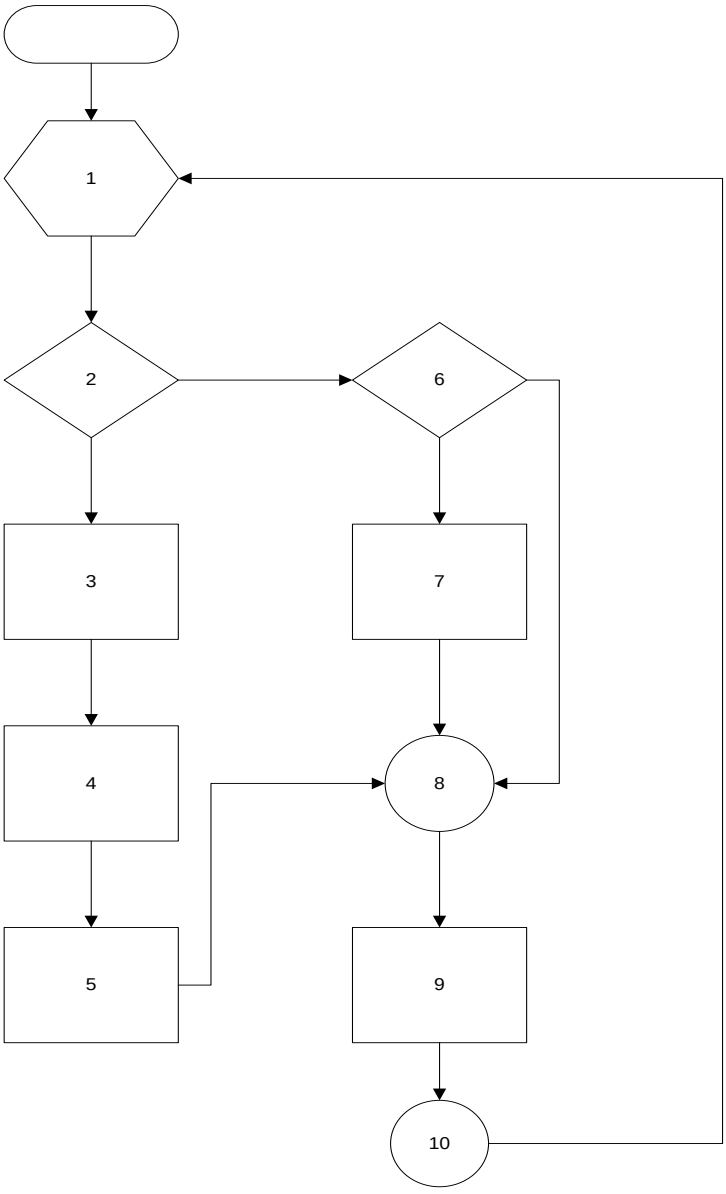
Dibawah ini merupakan kasus *Alpha Beta Pruning* untuk menguji perangkat lunak yang sudah dibangun menggunakan metode *white box*, berdasarkan skenario pengujian aplikasi yang sudah dijabarkan.

1. Mencari nilai terbesar.

   Berikut adalah *pseudocode* pencarian nilai terbesar untuk setiap posisi pemain.

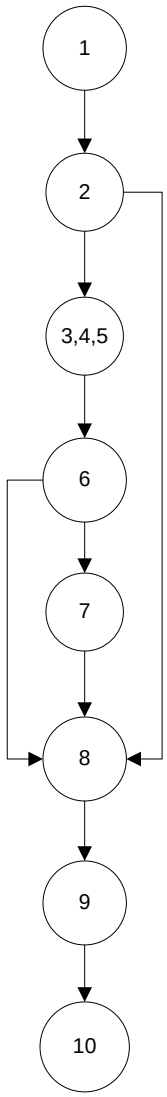
```
1  :  while(first != NULL)
2  :        if(first->value > max)
3  :              max = first->value;
4  :              best_cost = first->player;
5  :              alpha = max;
6  :        if(beta > alpha)
7  :              return best_cost;
8  :  endif
9 :    first = first->next;
10 :  endwhile
```

Berdasarkan *psudeocode procedure* diatas, dibuatlah *flowchart* yang dapat dilihat pada gambar 4.7.



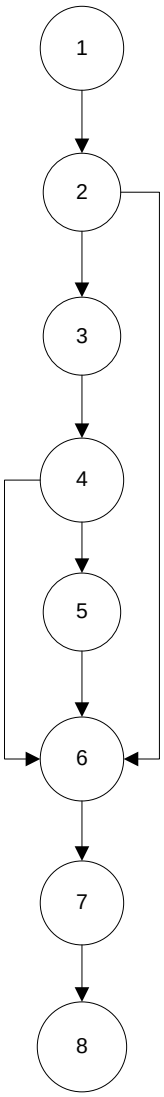
Gambar 4.7 Flowchart prosedur ABmax

Flowgraph yang merupakan hasil dari perubahan *flowchart* gambar 4.7 dapat dilihat pada gambar 4.8.



Gambar 4.8 Flowgraph ABmax

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut.



Gambar 4.9 Penyederhanaan Flowgraph ABmax

Jumlah dari *cyclomatic complexity* atau V(G) dari flowgraph pada gambar 4.9 adalah sebagai berikut :

Edge = *9* , *Node* = 8

V(G) = *Edge* – *Node* + 2

V(G) = 9 – 8 + 2

V(G) = 1 + 2

V(G) = 3

Dari gambar 4.9 didapat *independent path* sebagai berikut :

Path 1 = 1,2,3,4,5,6,7,8

Path 2 = 1,2,3,4,6,7,8

Path 3 = 1,2,6,7,8

Dengan menggunakan rumus *predicate node* didapatkan hasil sebagai berikut:

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

V(G) = 2 + 1

V(G) = 3

Graph Matriks pengujian dapat dilihat pada table 4.6

Tabel 4.6 Graph Matriks Prosedur ABmax

	1	2	3	4	5	6	7	8	9	10	11	12	n(E)-1
1		1											1-1=0
2			1			1							2-1=1
3				1									1-1=0
4					1	1							2-1=1
5						1							1-1=0
6							1						1-1=0
7								1					1-1=0
8													0

Hasil = 2

V(G)= jumlah *Graph Matriks* + 1

V(G)= 2+1

V(G) = 3

Kesimpulan yang didapat berdasarkan hasil pengujian dihasilkan *Cyclomatic Complexity* yang sama yaitu 3, maka dapat disimpulkan bahwa pada prosedur ABmax berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.2 Skenario I (UvA Riko VS UvA Trilearn)

Adapun skenario yang dilakukan adalah dengan melakukan simulasi pertandingan antara tim UvA Riko dengan tim UvA Trilearn. Dimana tim UvA Riko adalah tim yang dikembangkan dengan menggunakan algoritma *Alpha Beta Pruning*. Dimana simulasi pertandingan akan dilakukan sebanyak 5 kali dengan sistem kandang dan tandang, pertandingan tandang adalah tim berada disebelah kiri sedangkan untuk tandang tim berada di sebelah kanan. Satu pertandingan membutuhkan waktu 2 x 3000 time cycle. Time cycle merupakan waktu pada simulator Robocup Soccer, sedangkan dalam waktu nyata satu pertandingan adalah 10 menit.

4.2.2.1 Hasil Percobaan Skenario I

Tabel 4.8 Hasil pertandingan kandang UvA riko VS UvA Trilearn

Tim	Gol pada setiap pertandingan				
	1	2	3	4	5
UvA Riko	3	2	5	2	5
UvA Trilearn	2	2	2	2	1

Tabel 4.9 Statistik pertandingan kandang

Tim	Jumlah Gol	Jumlah Menang	Jumlah Seri	Jumlah Kalah
UvA Riko	17	3	2	0
UvA Trilearn	9	0	2	3

Tabel 4.10 Hasil pertandingan tandang UvA Riko VS UvA Trilearn

Tim	Gol pada setiap pertandingan				
	1	2	3	4	5
UvA Trilearn	0	1	2	1	1
UvA Riko	2	2	2	5	2

Tabel 4.11 Statistik pertandingan tandang

Tim	Jumlah Gol	Jumlah Menang	Jumlah Seri	Jumlah Kalah
UvA Trilearn	5	0	1	4
UvA riko	13	4	1	0

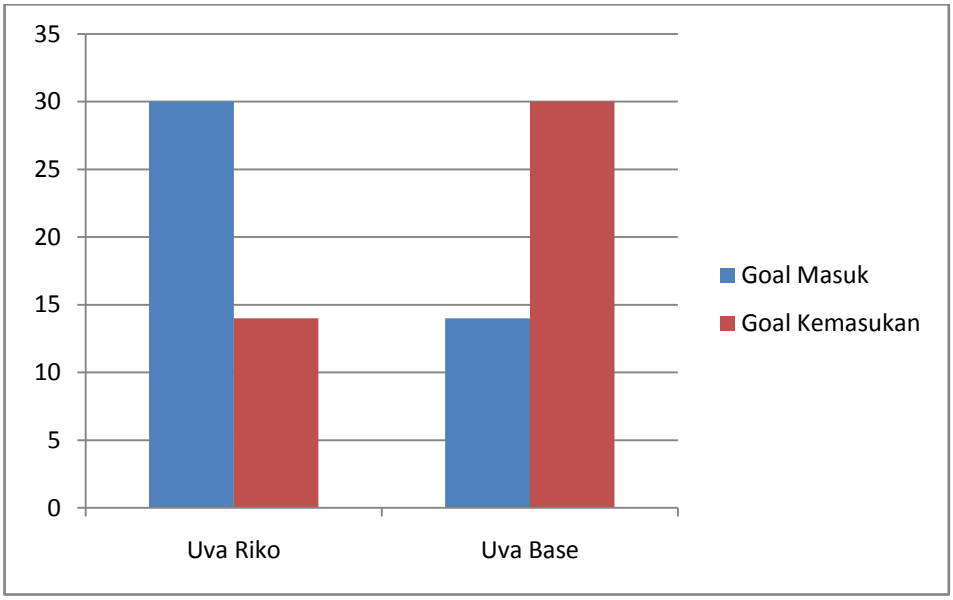
4.2.2.2 Analisa Skenario I

Tabel 4.12 Statistik Pertandingan

No	Tim	Main	M	S	K	Cetak Gol	Gol Kemasukan
1	UvA Riko	10	7	3	0	30	14
2	UvA Trilearn	10	0	3	7	14	30

Keterangan : M = Menang, S = Seri, K = Kalah

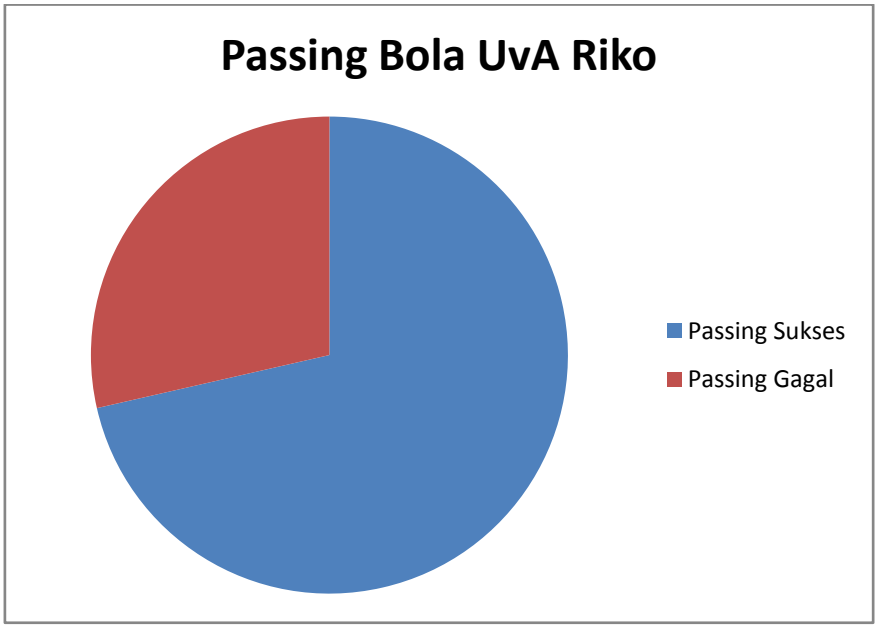
Berdasarkan table 4.12 dapat dilihat bahwa tim UvA riko yang dikembangkan dari 10 pertandingan memenangkan 7 pertandingan dan 3 kali seri dan tidak pernah kalah dari UvA trilearn. UvA riko mencetak 30 gol dari 10 pertandingan dan kemasukan 14 gol, sedangkan UvA Trilearn mencetak 14 gol dan kemasukan 30 gol. Dari hasil tersebut tim yang dikembangkan sudah bisa dikatakan lebih baik dibandingkan dengan UvA Trilean yang dijadikan Base Code. Untuk gambar pertandingannya dapat dilihat pada lampiran A. Berikut ini merupakan grafik pertandingan Uva riko vs Uva base dapat dilihat pada gambar 4.10.



Gambar 4.10 Statistik Uva Riko VS Uva Base

Berikut ini adalah log atau riwayat passing bola atau umpan bola yang dilakukan oleh *agent* Uva riko pada satu pertandingan melawan tim Uva Base. Dari log pertandingan diketahui tim Uva Riko melakukan umpan bola atau passing bola sebanyak 119 kali passing dan passing bola yang sukses dilakukan sebanyak 85 kali, kategori sukses tersebut adalah ketika bola sudah ditendang oleh *agent* pertama dan selanjutnya bola dikuasai oleh *agent* penerima bola atau bola berada pada area lingkaran yang mana titik pusatnya

adalah *agent* penerima dengan radius 5 grid koordinat. Sedangkan passing yang gagal sebanyak 34 kali, dimana kategori gagal adalah setelah bola di tendang oleh *agent* pertama bola berhasil dipotong oleh *agent* lawan.dengan demikian passing bola yang sukses dilakukan adalah sebesar 71.43% dan passing bola yang gagal adalah 28.57%. berikut ini adalah statistik passing bola yang dilakukan *agent* Uva riko dapat dilihat pada gambar 4.11.



Gambar 4.11 Statistik Passing Bola UvA Riko

4.2.3 Skenario II(UvA Riko VS WEBASE)

Adapun skenario yang dilakukan adalah dengan melakukan simulasi pertandingan antara tim UvA Riko dengan tim WEBase. Dimana simulasi pertandingan akan dilakukan sebanyak 5 kali dengan sistem kandang dan tandang, pertandingan tandang adalah tim berada disebelah kiri sedangkan untuk tandang tim berada di sebelah kanan. Satu pertandingan membutuhkan waktu 2 x 3000 time cycle. Time cycle merupakan waktu pada simulator Robocup Soccer, sedangkan dalam waktu nyata satu pertandingan adalah 10 menit.

4.2.3.1 Hasil Percobaan Skenario II

Tabel 4.13 Hasil Pertandingan Kandang UvA Riko VS WEBase

Tim	Gol pada setiap pertandingan				
	1	2	3	4	5
UvA Riko	5	5	5	6	3
WEBase	0	0	1	1	0

Tabel 4.14 Statistik Pertandingan Kandang

Tim	Jumlah Gol	Jumlah Menang	Jumlah Seri	Jumlah Kalah
UvA Riko	24	5	0	0
WEBase	2	0	0	5

Tabel 4.15 Hasil Pertandingan tandang UvA Riko VS WEBase

Tim	Gol pada setiap pertandingan				
	1	2	3	4	5
WEBase	2	3	3	1	1
UvA Riko	1	8	5	4	1

Tabel 4.16 Statistik Pertandingan tandang

Tim	Jumlah Gol	Jumlah Menang	Jumlah Seri	Jumlah Kalah
WEBase	10	1	1	3
UvA Riko	19	3	1	1

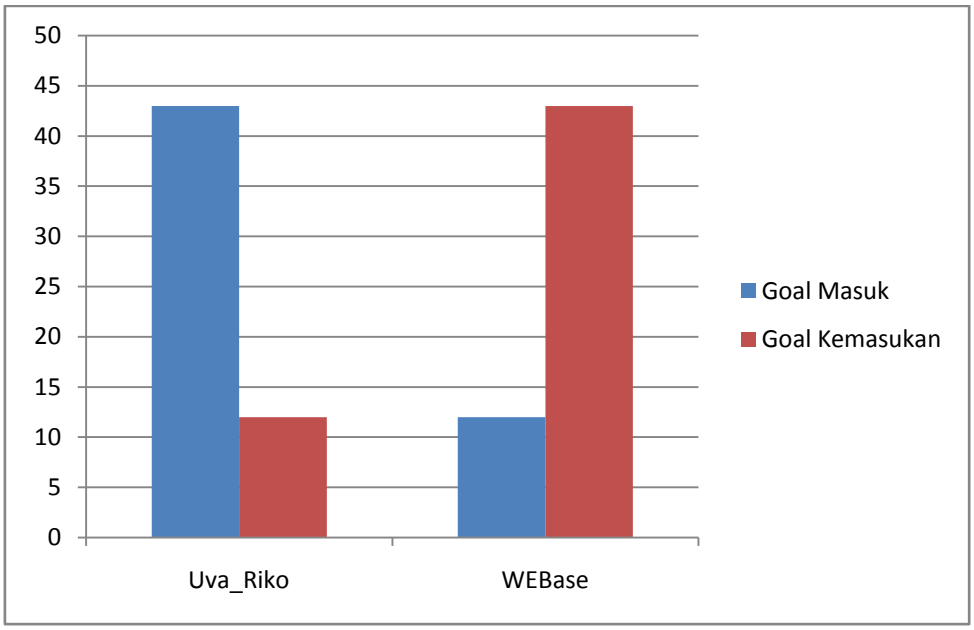
4.2.3.2 Analisa Skenario II

Tabel 4.17 Statistik Pertandingan

No	Tim	Main	M	S	K	Cetak Gol	Gol Kemasukan
1	UvA Riko	10	8	1	1	43	12
2	WEBase	10	1	1	8	12	43

Keterangan : M = Menang, S = Seri, K = Kalah

Berdasarkan table 4.17 dapat dilihat bahwa tim UvA riko dari 10 pertandingan memenangkan 8 pertandingan dan 1 kali seri dan 1 kali kalah. UvA riko mencetak 43 gol dari 10 pertandingan dan kemasukan 12 gol, sedangkan WEBase mencetak 12 gol dan kemasukan 43 gol. Untuk gambar pertandingannya dapat dilihat di lampiran A. Berikut ini merupakan grafik pertandingan Uva riko vs WEBase dapat dilihat pada gambar 4.12.



Gambar 4.13 Statistik Uva riko vs WEBase

4.2.4 Pengujian Alpha Beta Pruning

Pengujian dilakukan dengan membuat sebuah skenario dimana *agent* di tempatkan dalam sebuah posisi, dimana penempatan tersebut bertujuan untuk melihat apakah algoritma yang digunakan memilih posisi *agent* yang memiliki nilai paling tinggi diantara *agent* lainnya. Nilai-nilai untuk setiap posisi *agent* telah di hitung terlebih dahulu, sehingga terdapat beberapa tabel seperti tabel jarak bola, tabel jarak gawang lawan, tabel jarak musuh dan tabel nilai area. Untuk lebih jelasnya skenario posisi *agent* dapat dilihat pada gambar 4.14.



Gambar 4.14 Skenario posisi agent

1. Tabel jarak bola

Tabel 4.18 Nilai 1

P1	P2	P3	P4	P5	P7	P8	P9	P10	P11
-1	1	-1	-1	-1	1	1	-1	1	-1

2. Tabel jarak gawang lawan

Tabel 4.19 Nilai 2

P1	P2	P3	P4	P5	P7	P8	P9	P10	P11
-1	-1	-1	-1	-1	1	-1	1	1	1

3. Tabel jarak musuh

Tabel 4.20 Nilai 3

P1	P2	P3	P4	P5	P7	P8	P9	P10	P11
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

4. Tabel nilai area

Tabel 4.21 Nillai 4

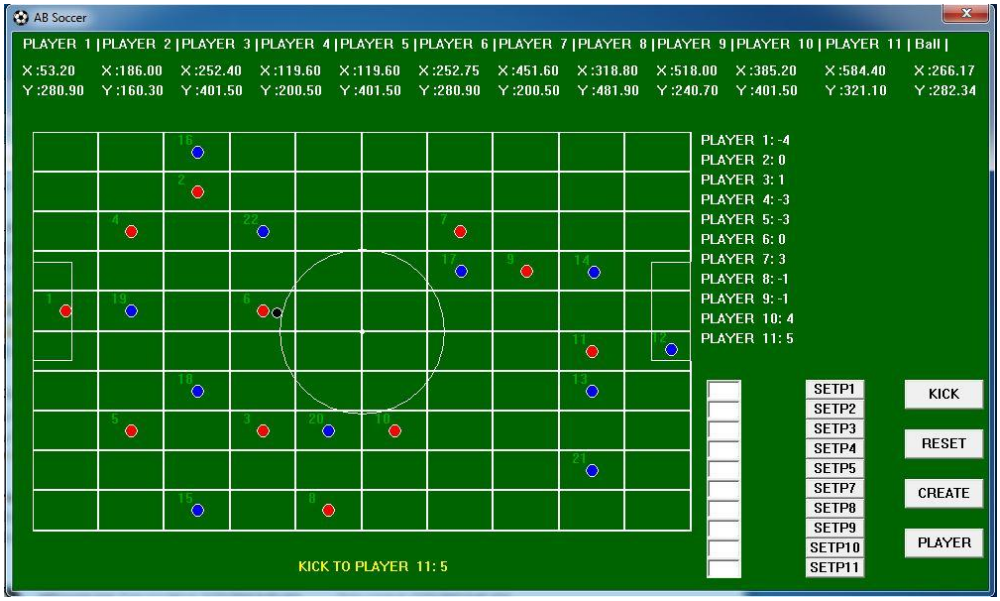
P1	P2	P3	P4	P5	P7	P8	P9	P10	P11
-2	1	1	-2	-2	4	2	5	3	6

5. Total nilai

Tabel 4.22 Total

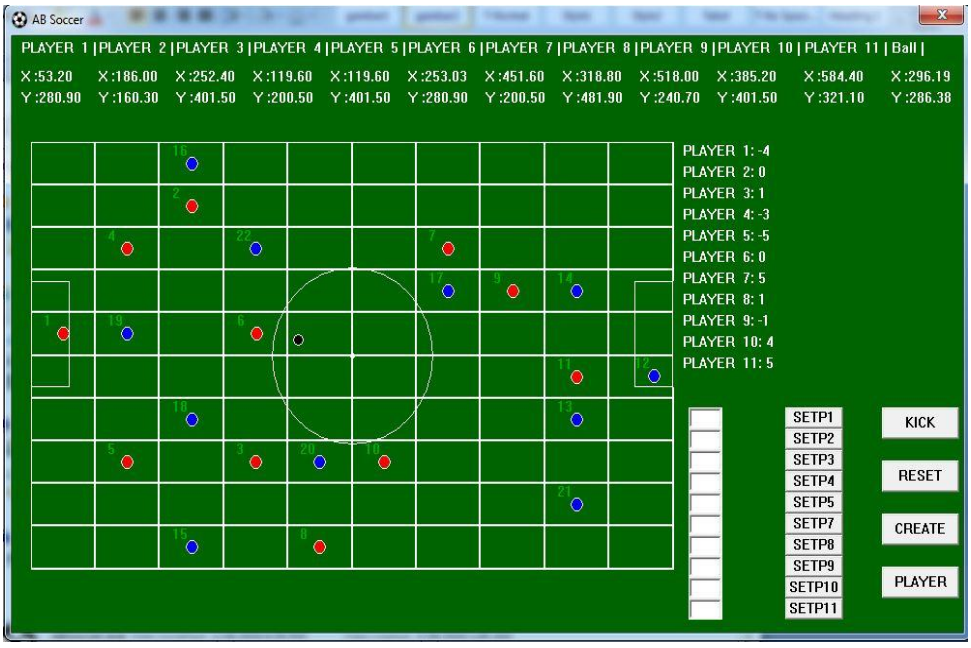
P1	P2	P3	P4	P5	P7	P8	P9	P10	P11
-5	0	-2	-5	-5	3	1	4	4	5

Jika dilihat dari tabel 4.22 maka pemain nomor 6 akan mengoper bola ke pemain nomor 11. Untuk lebih jelasnya bisa dilihat pada gambar 4.15.



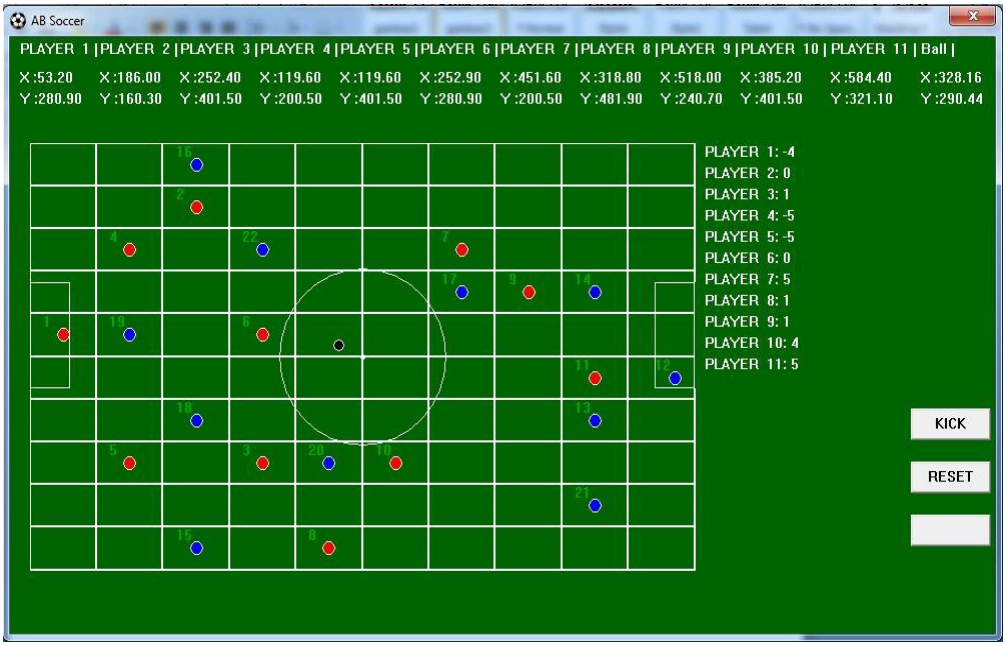
Gambar 4.15 Agent nomor 6 menuju bola

Berikut ini dapat dilihat *agent* 6 menendang bola ke arah *agent* 11. Untuk lebih jelasnya dapat dilihat pada gambar 4.16.



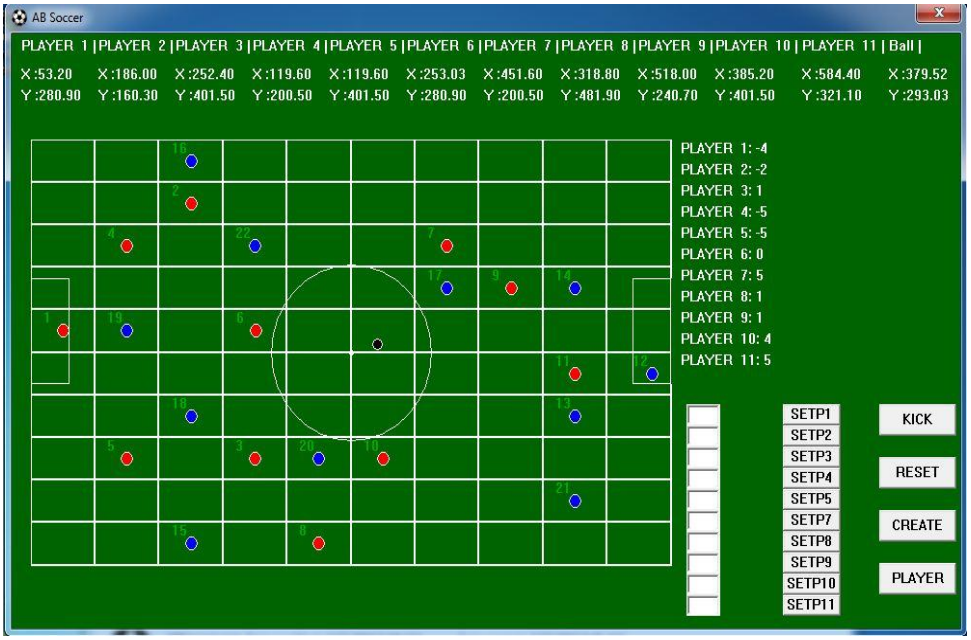
Gambar 4.16 Bola menuju agent 11

Berikut ini merupakan pergerakan bola menuju posisi *agent* 11. Untuk lebih jelasnya dapat dilihat pada gambar 4.17.



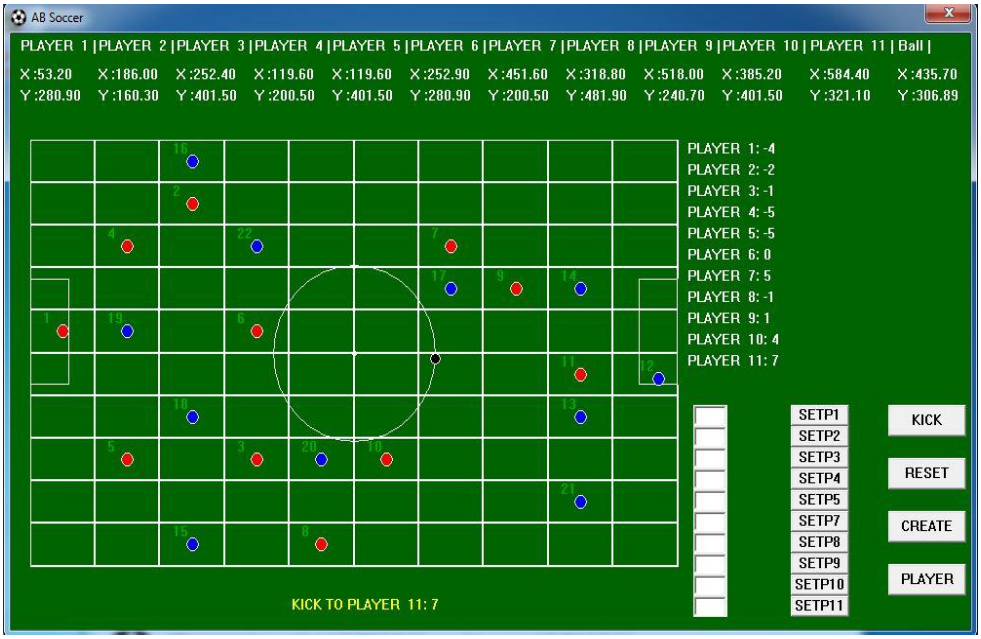
Gambar 4.17 Bola menuju agent 11 lanjutan

Berikut ini merupakan pergerakan bola menuju posisi *agent* 11. Untuk lebih jelasnya dapat dilihat pada gambar 4.18.



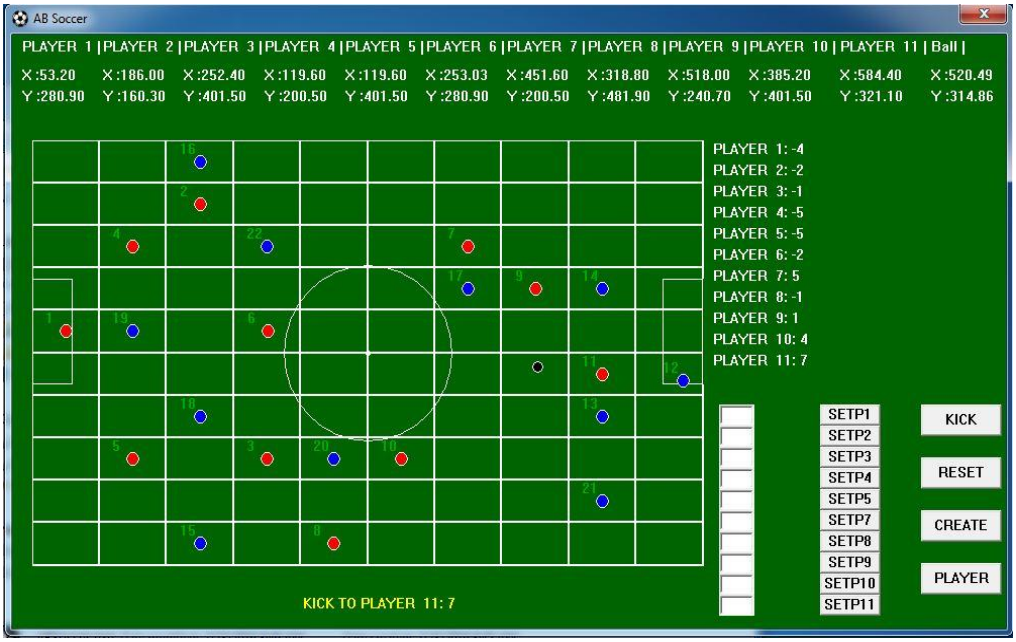
Gambar 4.18 Bola Menuju Agent 11 lanjutan

Berikut ini merupakan pergerakan bola menuju posisi *agent* 11. Untuk lebih jelasnya dapat dilihat pada gambar 4.19.



Gambar 4.19 Bola Menuju Agent 11 Lanjutan

Berikut ini merupakan pergerakan bola menuju posisi *agent* 11. Untuk lebih jelasnya dapat dilihat pada gambar 4.20.



Gambar 4.20 Bola Menuju Agent 11 Lanjutan