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.

IV.1 Implementasi

Tujuan implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem sehingga *user* dapat memberi masukan demi berkembangnya sistem yang telah dibangun sebagai simulasi dari aplikasi penerapan algoritma minimax untuk menentukan langkah yang tepat yang dilakukan komputer sehingga komputer menjadi lawan yanng tidak terkalahkan. Perangkat keras dan perangkat lunak yang digunakan untuk melakukan implementasi dan pengujian aplikasi yaitu sebagai berikut :

a. Perangkat keras

Adapun Perangkat keras yang digunakan untuk pembuatan aplikasi adalah sebagai berikut:

1. processor 1,5 Ghz

2. memori 1 GB

4. Layar resolusi 1024X760 pixel

5. LCD 10,1 inc

6. Harddisk 250 GB

7. Mouse

b. Perangkat lunak

Perangkat lunak yang digunakan untuk mengimplementasikan sistem dijelaskan pada tabel IV.1.

Tabel IV.1 Perangkat Lunak yang Digunakan

Komputer	Spesifikasi perangkat lunak
Sistem Operasi	Microsoft Window 7 ultimate 32 bit
Bahasa Pemograman	Borland Delphi 7.0

Sistem operasi pada tabel IV.1 harus sudah terinstal pada komputer, agar aplikasi yang dibangun dapat dijalankan pada komputer tersebut. Jika pengguna ingin membuka *project*-nya maka terlebih dahulu harus menginstalkan Borland Delphi dan *component* yang mendukung.

IV.1.1 Implementasi *Class*

Tabel IV.2 Implementasi *Class*

No	Nama *Class*	Nama File	Jenis
1	TForm		*Class Controll*
1	TFMenuUtama	U_menuutama.pas	*Class Boundary*
2	TFabout	U_about.pas	*Class Boundary*
3	TFgamedirection	U_gamedirection.pas	*Class Boundary*
4	TFormUtama	U_formutama.pas	*Class Boundary*
5	TFpengaturan	U_pengaturan.pas	*Class Boundary*
6	TSetupDlg	U_setup.pas	*Class Boundary*

```
Create Class 'TForm'
```

```
Create Class 'TFMenuUtama'
type
  TFMenuUtama = class(TForm)
    mulaiBtn: TBitBtn;
    keluarBtn: TBitBtn;
    BantuanRGp: TRadioGroup;
    procedure mulaiBtnClick(Sender: TObject);
    procedure keluarBtnClick(Sender: TObject);
    procedure BantuanRGpClick(Sender: TObject);
  end;
```

```
Create Class 'TFabout'
type
  TFabout = class(TForm)
    BitBtn1: TBitBtn;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    Memo1: TMemo;
    Panel1: TPanel;
    Image1: TImage;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
  procedure Flatten(theControl: TWinControl);
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
Create Class 'TFgamedirection'
type
  TFgamedirection = class(TForm)
    BitBtn1: TBitBtn;
    Timer1: TTimer;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Memo1: TMemo;
    Memo2: TMemo;
    Memo3: TMemo;
    Memo4: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    procedure Timer1Timer(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
Create Class 'TFormUtama'
type
  Trecord = record
    nbrtokens:integer;
    baris:integer;
    objek:TList;
  end;
  TFormUtama = class(TForm)
    HumanRGrp: TGroupBox;
    Panel1: TPanel;
    Timer1: TTimer;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Bevel1: TBevel;
    ListBox1: TListBox;
```

```
    solveBtn: TButton;
    AmbilBtn: TButton;
    statistList: TListBox;
    Label14: TLabel;
    StatusBar1: TStatusBar;
    Shape1: TShape;
    Image1: TImage;
    com_image: TImage;
    hum_image: TImage;
    kamu_lbl: TLabel;
    hore_lbl: TLabel;
    UlangSBtn: TSpeedButton;
    mainbaruSBtn: TSpeedButton;
    Label5: TLabel;
    Label4: TLabel;
    Label1: TLabel;
    procedure AmbilBtnClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Shape1MouseDown(Sender: TObject; Button:
  TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Panel1Resize(Sender: TObject);
    procedure mainbaruSBtnClick(Sender: TObject);
    procedure UlangSBtnClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure solveBtnClick(Sender: TObject);
    procedure kembaliBtnClick(Sender: TObject);
   private
    procedure Flatten(theControl: TWinControl);
   public
    NextPlayer:char;
    misere:boolean;
    normal:integer;
    board:array of Trecord;
    tokensize:integer;
    usedrows:integer;
    clickedrow:integer;
    gameover:boolean;
    player1move:boolean;
    clickedcol:integer;
    player1, player2:string;
    blink: boolean;
    ambil: integer;
    procedure computermove;
    procedure restoregame;
    procedure setplayer;
    procedure setplayernames;
  end;
```

```
Create Class 'TFpengaturan'
type
  TFpengaturan = class(TForm)
    RadioGroup1: TRadioGroup;
```

```
    RadioGroup2: TRadioGroup;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
Create Class 'TSetup'
type
  TSetupDlg = class(TForm)
    StatusBar1: TStatusBar;
    Boardgrid: TStringGrid;
    RandomBoardBtn: TButton;
   procedure RandomBoardBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

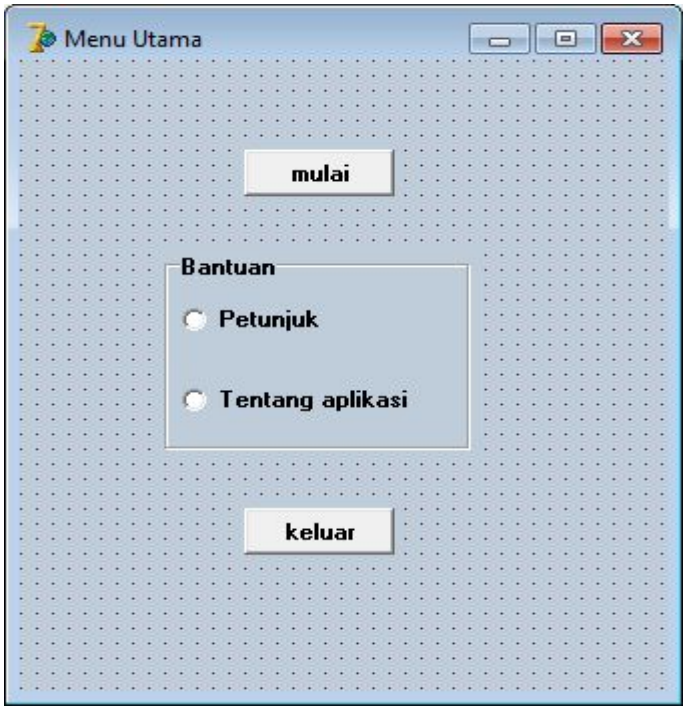
IV.1.2 Implementasi Antarmuka

Implementasi antarmuka untuk perancangan aplikasi tampilan utama yang dibangun dapat dilihat dari gambar IV.1 sampai gambar IV.3.

1. Implementasi Antarmuka Form Menu Utama

   Nama Form : TformMenuUtama

   Nama Program : U_menuutama.pas



Gambar IV.1 Implementasi Antarmuka Form Menu Utama

Di dalam menu utama terdapat beberapa menu, di antaranya yaitu: Menu mulai, bantuan, dan keluar. Ketika menekan tombol mulai maka akan muncul form pengaturan seperti pada gambar IV.2.

Di dalam menu Bantuan terdapat submenu petunjuk, untuk melihat petunjuk permainan, diperlihatkan pada gambar IV.4 , dan submenu tentang aplikasi untuk melihat sekilas informasi tentang aplikasi ini, gambar IV.5.

Terakhir menu keluar untuk keluar aplikasi. Setelah menekan tombol ini, kemudian akan muncul pesan konfirmasi keluar aplikasi. Jika ya, aplikasi akan ditutup, jika tidak, akan kembali ke menu utama. Tampilan pesan konfirmasi dapat dilihat pada gambar IV.6.

2. Implementasi Antarmuka Form Pengaturan.

Nama Form : TFPengaturan

Nama Program : U_pengaturan.pas



Gambar IV.2 Implementasi Antarmuka Form Pengaturan

Selesai mengisi pengaturan, klik ok kemudian akan menampilkan form utama permainan, seperti pada gambar IV.3. klik kembali untuk kembali ke form menu utama.

3. Implementasi Antarmuka Form Utama

Dalam form utama, ada beberapa tombol yang dapat dijalankan. Antara lain: Tombol ambil, rekomendasi, ulang, main baru, dan kembali. Tombol ambil funggsinya untuk mengambil objek. Klik objek pada papan permainan, jumlahnya

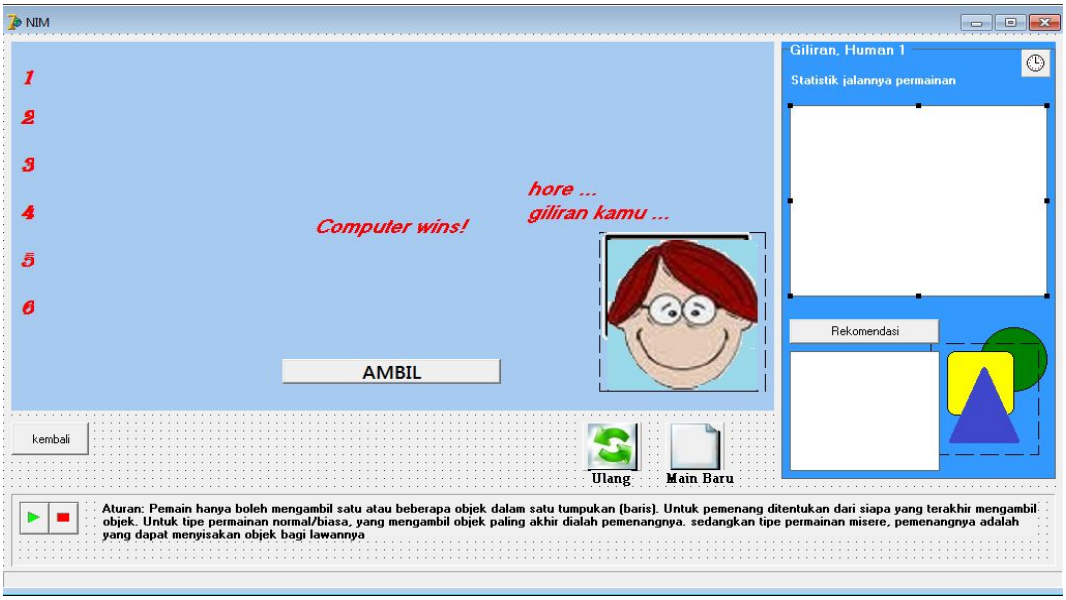
bebas, boleh satu, dua, atau lebih pada tiap tumpukan (baris). Kemudian tekan tombol ambil, maka objek akan hilang sesuai jumlah objek yang kita klik.

Tombol rekomendasi fungsinya untuk memberikan rekomendasi atau saran bagi pemain manusia, berapa jumlah objek yang dapat diambil. Rekomendasi hanya dibatasi sebanyak 5 kali.

Tombol ulang untuk mengulang kembali permainan, bedanya dengan main baru adalah, jika main baru setting papan permainan akan berbeda dengan yang sudah dimainkan. Sedangkan ulang, hanya mengulang permainan dengan setting papan permainan yang sama dengan yang sudah dimainkan terakhir. Tombol kembali untuk kembali ke form pengaturan.

Nama : TFormUtama

Nama Program : U_FormUtama.pas

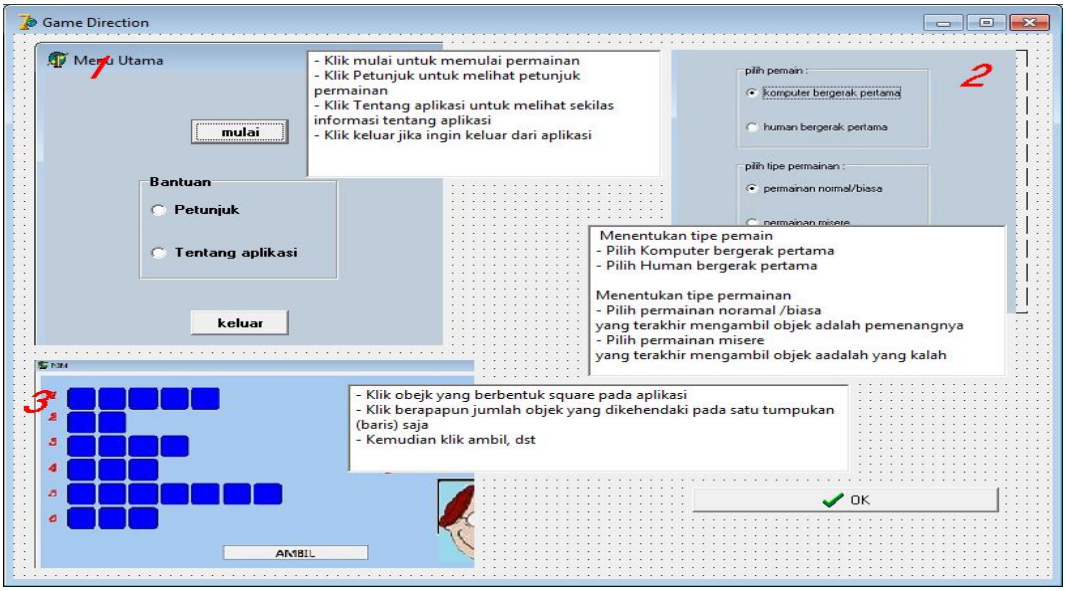


Gambar IV.3 Implementasi Antarmuka Form Utama

4. Implementasi Antarmuka Form Petunjuk

Nama Form : Tfgamedirection

Nama Program : U_gamedirection.pas

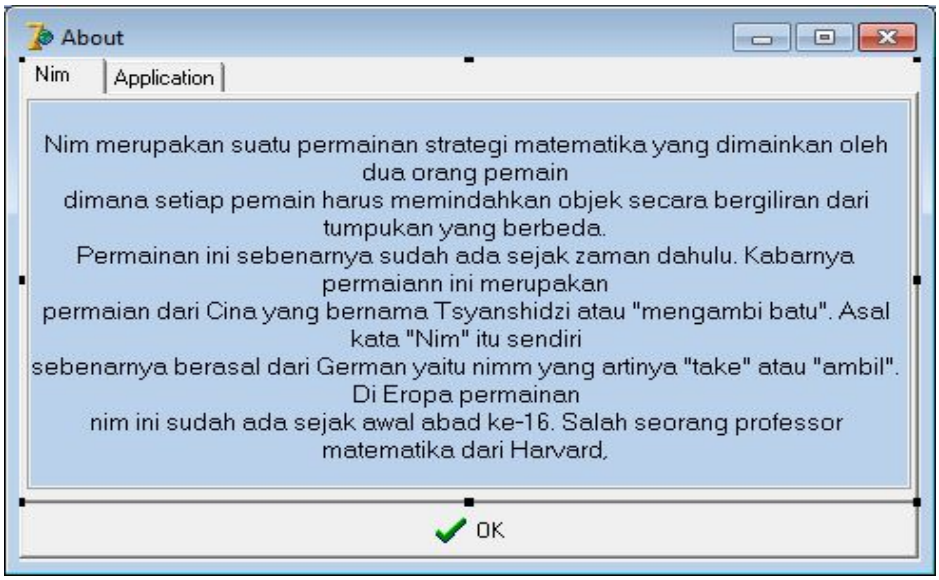


Gambar IV.4 Implementasi Antarmuka Form Petunjuk

5. Implementasi Antarmuka Form Tentang Aplikasi

Nama Form : Tfabout

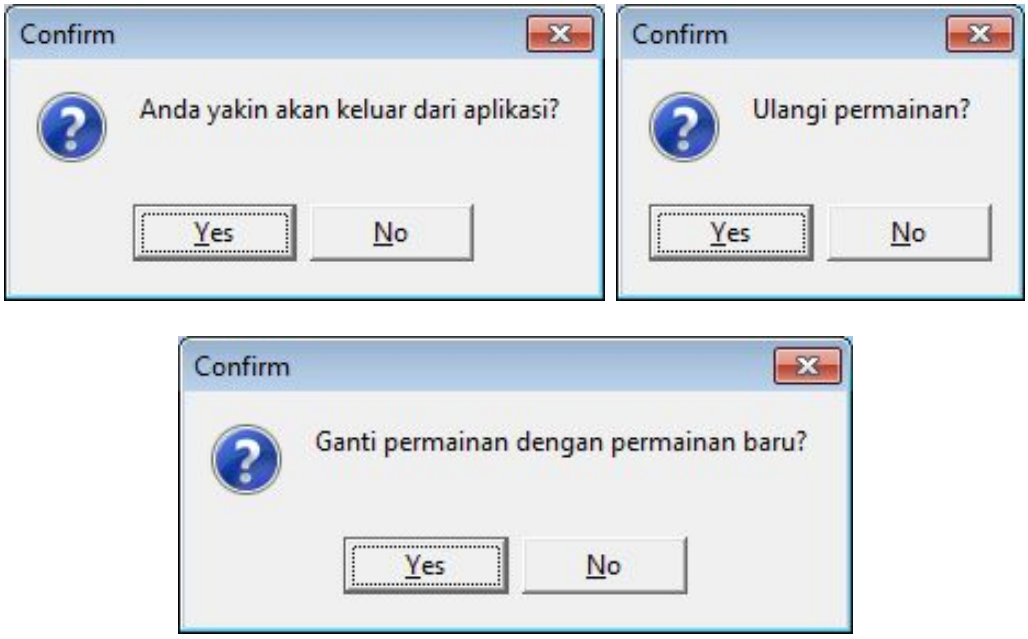
Nama Program : U_about.pas



Gambar IV.5 Implementasi Antarmuka Form Tentang

6. Implementasi Antarmuka Pesan Konfirmasi

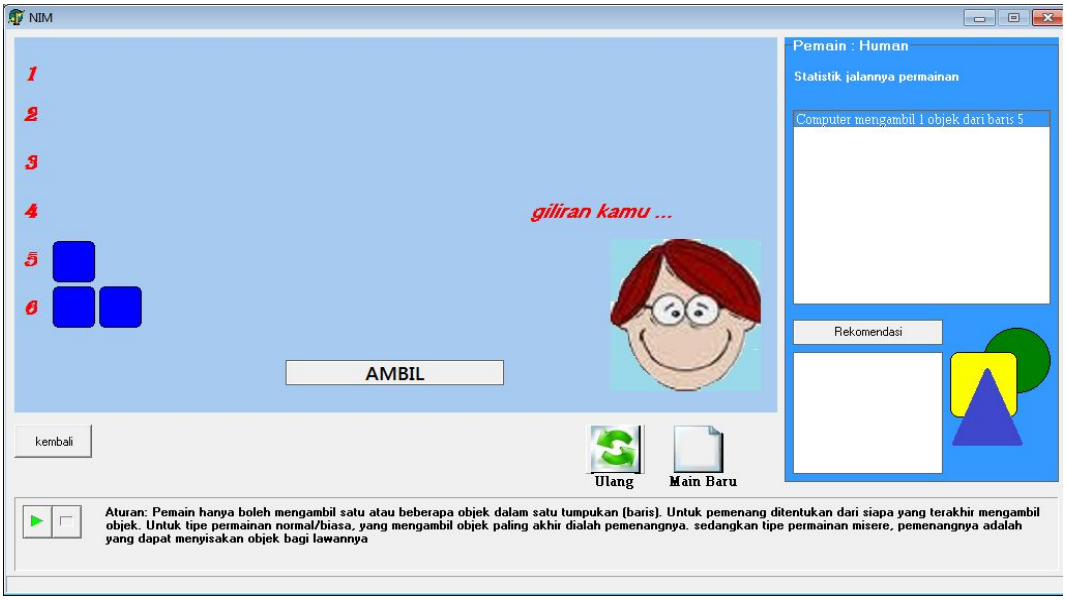
Berdasarkan aplikasi yang sudah dirancang, terdapat beberapa pesan konfirmasi diantaranya konfirmasi untuk mengganti permainan baru, konfirmasi untuk mengulang permainan, dan konfirmasi untuk keluar dari aplikasi. Diperlihatkan pada gambar IV.6



Gambar IV.6 Implementasi Antarmuka Pesan Konfirmasi

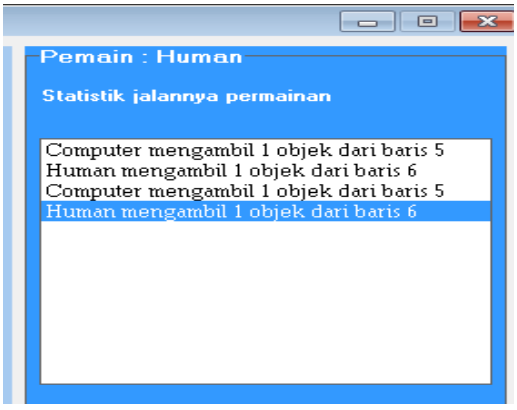
IV.1.3 Implementasi Permainan

Setelah membuka aplikasinya, klik mulai, pilih pengaturan, untuk implemtasi kali ini, untuk tipe pemaiin pilh komputer yang bergerak lebih dahulu dan untuk tipe permainannya, pilih normal/biasa.

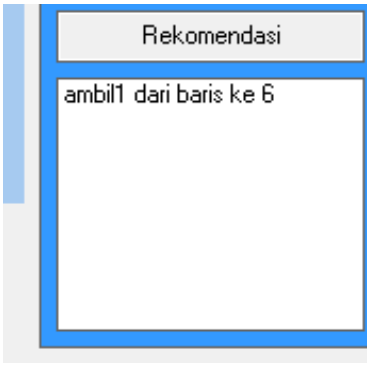


Gambar IV.7 Implementasi permainan

Gambar IV.7 memperlihatkan form permainan yang sedang aktif dengan jumlah masing-masing 2 objek pada tumpukan(baris) ke-5 dan 2 objek pada tumpukan(baris) ke-6. Sesuai dengan pengaturan yang sudah dipilih sebelumnya, komputer bergerak pertama, setiap jalannya permainan dapat dilihat pada statistik jalannya permainan, gambar IV.8. Dan saran dari komputer dapat dilihat pada gambar IV.9.



Gambar IV.8 Implementasi Statistik Jalannya Permainan



Gambar IV.9 Implementasi Rekomendasi Jalan bagi Pemain

IV.2 Pengujian

Pengujian yang digunakan untuk menguji sistem ini dilakukan dengan menggunakan pengujian Alpha dan Beta.

IV.2.1 Pengujian Alpha

Metode yang digunakan dalam pengujian alpha yaitu dengan menggunakan pendekatan metode *blackbox* dan *whitebox.*

Metode *Blackbox*

Pengujian dengan metode *blackbox* yaitu pengujian yang fokus pada persyaratan fungsional dari perangkat lunak yang dibangun. Berikut rencana pengujian menggunakan metode *blackbox.*

Tabel IV.3 Tabel Rencana Pengujian

Item Uji	Detail Pengujiam
Pilih tipe pemain	Komputer bergerak pertama
	Human bergerak pertama
Pilih tipe permainan	Permainan normal/biasa
	Permainan misere

Tabel IV.4 Kasus dan Hasil Pengujian

Kasus dan Hasil Uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Pilih pemain: komputer bergerak pertama	Dapat menjalankan permainan dengan komputer yang bergerak lebih dahulu	Dapat melakukan permainan sesuai dengan yang diharapkan	Diterima
Pilih pemain: human bergerak pertama	Dapat menjalankan permainan dengan pemain manusia yang bergerak lebih dahulu	Dapat melakukan permainan sesuai dengan yang diharapkan	Diterima
Pilih permainan normal/biasa	Dapat menjalankan permainan dengan ketentuan yang terakhir mengambil objek adalah pemenangnya	Dapat melakukan permainan sesuai dengan ketentuan yang diharapkan	Diterima
Pilih permainan misere	Dapat menjalankan permainan dengan ketentuan yang terakhir mengambil objek adalah yang kalah	Dapat melakukan permainan sesuai dengan ketentuan yang diharapkan	Diterima

Metode *Whitebox*

Sedangkan metode *whitebox* yaitu pengujian yang didasarkan pada pengecekan terhadap detil perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur system, pengetahuan mengenai program digunakan untuk mengidentifikasikan kasus uji tambahan.

Berdasarkan rencana pengujian yang telah disusun, maka dapat dilakukan pengujiannya sebagai berikut:

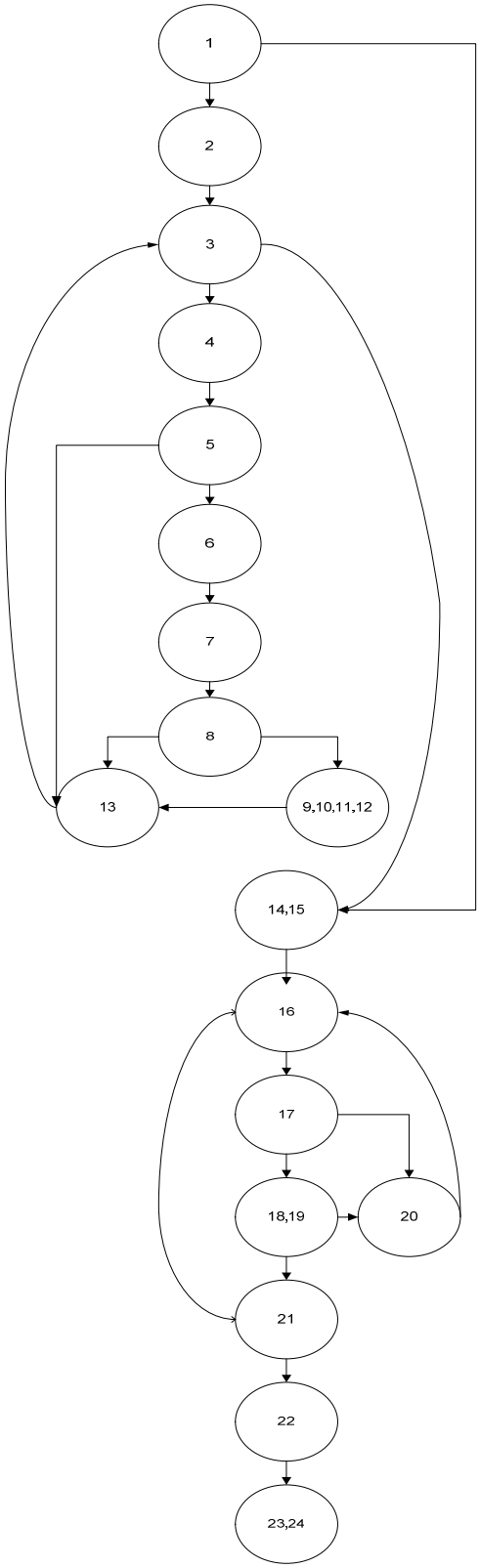
1. Pengujian *Basis Path*

Pengujian basis path adalah pengujian *white box* yang diusulkan pertama kali oleh Tom McCabe. Metode ini memungkinkan untuk mengukur kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menetapkan himpunan basis dari semua jalur eksekusi.

Langkah pergerakan komputer

```
1     : if row<0 then
        begin
2     :     n:=board[0].objek.count;
3     :     for i:=1 to maxbaris-1 do
4     :           n:=n xor board[i]. objek.count;
5     :     if n>0 then
            begin
6     :           for i:=0 to maxbaris -1 do
                  begin
7     :               j:=n xor board[i]. objek.count;
8     :               if j < board[i]. objek.count then
                      begin
9     :                     row:=i;
10    :                     ambil:=board[i]. objek.count - j;
11    :                     break;
12    :               end;
13    :           end;
      :     end
        else
```

```
        begin
14    :     max:=0;
15    :     row:=-1;
16    :     for i:=0 to maxbaris -1 do
            begin
17    :         if board[i]. objek.count>max then
              begin
18    :               row:=i;
19    :               max:=board[i]. objek.count;
20    :         end;
21    :      end;
22    :     ambil:=1;
23    :     end;
24    : end;
```



Gambar IV.9 *Flow graf*

2. Kompleksitas Siklomatis

Adalah metrik perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program, nilai yang didapat akan menentukan jumlah jalur independen dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali.

Adapun jalur independen yang didapat berdasarkan *flow graph* yang sudah dibuat sebelumnya, adalah:

Jalur 1 : 1-2-3-4-5-6-7-8-9,10,11,12-13-3-14,15-16-17-18,19-20-16-21-22-23,24

Jalur 2 : 1-2-3-4-5-6-7-8-13-3-14,15-16-17-18,19-20-16-21-22-23,24

Jalur 3 : 1-2-3-4-5-13-3-14,15-16-17-18,19-20-16-21-22-23,24

Jalur 4 : 1-2-3-14,15-16-17-18,19-20-16-21-22-23,24

Jalur 5 : 1-2-3-14,15-16-17-20-16-21-22-23,24

Jalur 6 : 1-14,15-16-17-18,19-20-16-21-22-23,24

Jalur 7 : 1-14,15-16-17-20-16-21-22-23,24

a. Jumlah region grafik = 7

b. *Cyclomatix complexity* V(G)

V(G) = E – N + 2

= 23 – 18 +2

= 7

ket:

E = jumlah *edge* pada grafik alir

N = jumlah *node* pada grafik alir

c. V(G) = P +1

= 6 + 1= 7

Ket:

P = Simpul predikat, yaitu penggambaran suatu *node* yang memiliki lebih dari satu atau beberapa inputan dan lebih dari satu output.

IV.2.1.1 Kesimpulan Pengujian Alpha

Berdasarkan hasil pengujian dengan kasus sampel uji yang telah dilakukan, penulis memberikan kesimpulan dari proses pengujian bahwa pada proses masih memungkinkan terjadinya kesalahan, tampilan program pun belum sempurna diciptakan dan terdapat kekurangan. Namun secara fungsional perangkat lunak bantu yang dibangun ini sudah dapat menghasilkan output yang diharapkan.

IV.2.2 Pengujian Beta

Pengujian Betha merupakan pengujian yang dilakukan secara objektif dimana diuji secara langsung ke lapangan yaitu *user* yang bersangkutan dengan membuat kusioner mengenai kepuasan *user* dengan kandungan poin syarat *user friendly* untuk selnjutnya dibagikan kepada sebagian *user*. Dari hasil kuesioner tersebut akan dilakukan perhitungan untuk dapat diambil kesimpulannya terhadap penilaian penerapan sistem yang baru.

Kuesioner disebar menggunakan teknik sampling yaitu *Simple Random Sampling* yang disebarkan ke beberapa pengguna. Dari hasil kuesioner tersebut

akan dilakukan perhitungan agar dapat diambil kesimpulnnya terhadap penilaian penerapan sistem yang baru.

Berdasarkan hasil pengujian beta, dicari persentase masing-masing jawaban dengan menggunakan rumus:

$$P \quad \frac{S}{Skor\ ideal} \quad 100\%$$

Keterangan:

P = jumlah jawaban responden tiap soal

S = frekuensi jawaban dikali skor

Skor = nilai Jawaban

Skor ideal = jumlah skor dikali nilai jawaban

Adapun hasil persentasi dari setiap jawaban yang sudah dihitung nilainya adalah sebagai berikut. Dengan jumlah responden sebanyak 14 orang.

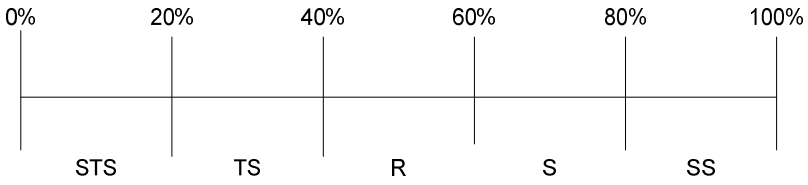
Pertanyaan no 1:

"Apakah tampilan dari aplikasi yang dibuat sudah menarik?"

Tabel IV.5 Hasil pengujian no 1

Kategori Jawaban	Frekuensi Jawaban	*Score*	S
Sangat Setuju	2	5	10
Setuju	10	4	40
Ragu-ragu	2	3	6
Tidak Setuju	0	2	0
Sangat Tidak Setuju	0	1	0
JUMLAH	14		56

$$P \quad \frac{56}{5 \quad 14} \quad 100\% \quad 80\%$$



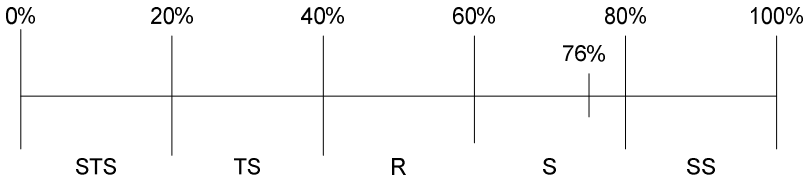
Pertanyaan no 2:

“Apakah struktur dari aplikasi perangkat lunak ini mudah untuk dipelajari?”

Tabel IV.6 Hasil pengujian no 2

Kategori Jawaban	Frekuensi Jawaban	*Score*	S
Sangat Setuju	0	5	0
Setuju	11	4	44
Ragu-ragu	3	3	9
Tidak Setuju	0	2	0
Sangat Tidak Setuju	0	1	0
JUMLAH	14		53

$$P \quad \frac{53}{5 \quad 14} \quad 100\% \quad 76\%$$



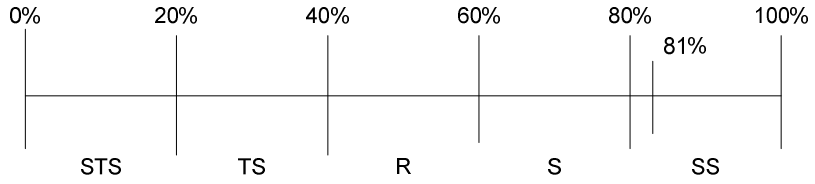
Pertanyaan no 3:

“Apakah aplikasi yang dibuat mudah dioperasikan/ digunakan?”

Tabel IV.7 Hasil pengujian no 3

Kategori Jawaban	Frekuensi Jawaban	*Score*	S
Sangat Setuju	2	5	10
Setuju	11	4	44
Ragu-ragu	1	3	3
Tidak Setuju	0	2	0
Sangat Tidak Setuju	0	1	0
JUMLAH	14		57

$$P \quad \frac{57}{5 \quad 14} \quad 100\% \quad 81\%$$



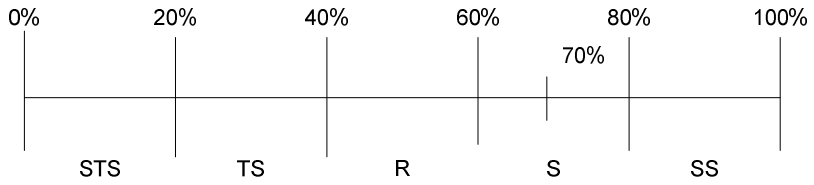
Pertanyaan no 4:

“Apakah dengan diterapkannya metode minimax dalam permainan membuat permainannya semakin menarik?”

Tabel IV.8 Hasil pengujian no 4

Kategori Jawaban	Frekuensi Jawaban	*Score*	S
Sangat Setuju	1	5	5
Setuju	6	4	24
Ragu-ragu	6	3	18
Tidak Setuju	1	2	2
Sangat Tidak Setuju	0	1	0
JUMLAH	14		49

$$P \quad \frac{49}{5 \quad 14} \quad 100\% \quad 70\%$$



Beberapa saran dari responden yang dapat disimpulkan adalah sebagai berikut:

1. Lebih dipersulit lagi tingkat permainannya
2. Agar aplikasinya didesain lebih menarik lagi

IV.2.2.1 Kesimpulan Hasil Pengujian Beta

Berdasarkan dari jawaban yang diberikan oleh responden, bisa ditarik kesimpulan bahwa penerapan algoritma minimax cukup mudah dipahami, aplikasi yang dibuat sesuai dengan tujuan.